aws

Panduan Pengguna

Amazon CloudWatch

Amazon CloudWatch: Panduan Pengguna

Table of Contents

Apa itu Amazon CloudWatch?

Amazon CloudWatch memantau sumber daya Amazon Web Services (AWS) Anda dan aplikasi yang Anda jalankan AWS secara real time. Anda dapat menggunakan CloudWatch untuk mengumpulkan dan melacak metrik, yang merupakan variabel yang dapat Anda ukur untuk sumber daya dan aplikasi Anda.

CloudWatch Halaman beranda secara otomatis menampilkan metrik tentang setiap AWS layanan yang Anda gunakan. Anda dapat membuat dasbor kustom tambahan untuk menampilkan metrik tentang aplikasi khusus Anda, dan menampilkan koleksi metrik-metrik kustom yang Anda pilih.

Anda dapat membuat alarm yang mengawasi metrik dan mengirimkan notifikasi atau secara otomatis melakukan perubahan pada sumber daya yang sedang dipantau ketika ada pelanggaran ambang batas. Misalnya, Anda dapat memantau penggunaan CPU dan pembacaan cakram serta penulisan instans Amazon EC2 Anda dan kemudian menggunakan data tersebut untuk menentukan apakah Anda harus meluncurkan instans tambahan untuk menangani peningkatan beban. Anda juga dapat menggunakan data ini untuk menghentikan instans yang tidak digunakan untuk menghemat uang.

Dengan CloudWatch, Anda mendapatkan visibilitas seluruh sistem ke dalam pemanfaatan sumber daya, kinerja aplikasi, dan kesehatan operasional.

Mengakses CloudWatch

Anda dapat mengakses CloudWatch menggunakan salah satu metode berikut:

- CloudWatch Konsol Amazon - https://console.aws.amazon.com/cloudwatch/
- AWS CLI - Untuk informasi selengkapnya, lihat Menyiapkan denganAWS Command Line Interface Panduan Pengguna. AWS Command Line Interface
- CloudWatch API — Untuk informasi selengkapnya, lihat Referensi Amazon CloudWatch API.
- AWS SDK — Untuk informasi selengkapnya, lihat Alat untuk Amazon Web Services.

AWS Layanan terkait

Layanan berikut digunakan bersama dengan Amazon CloudWatch:

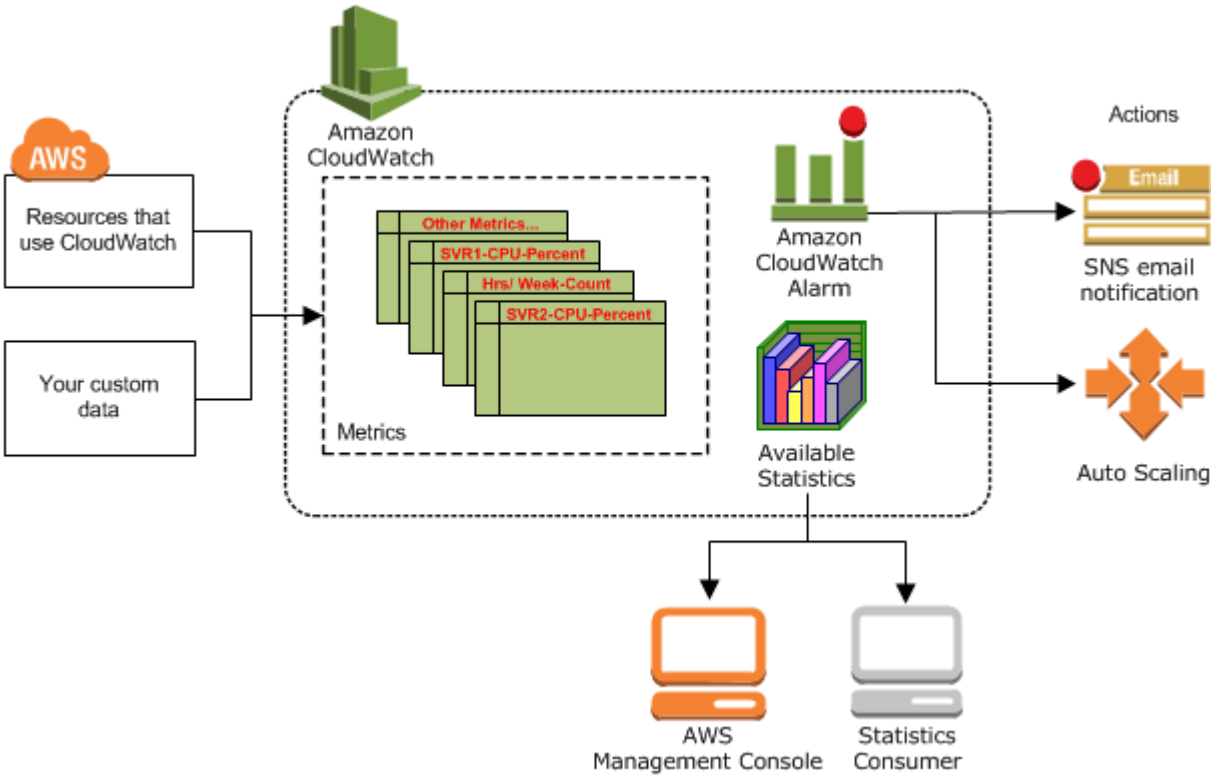
- Amazon Simple Notification Service (Amazon SNS) mengoordinasikan dan mengelola penyampaian atau pengiriman pesan ke titik akhir atau klien yang berlangganan . Anda

menggunakan Amazon SNS CloudWatch untuk mengirim pesan ketika ambang batas alarm telah tercapai. Untuk informasi selengkapnya, lihat Menyiapkan notifikasi Amazon SNS.

- Amazon EC2 Auto Scaling memungkinkan Anda untuk secara otomatis meluncurkan atau mengakhiri instans Amazon EC2 didasarkan pada kebijakan, pemeriksaan status kesehatan, dan jadwal yang ditentukan pengguna. Anda dapat menggunakan CloudWatch alarm dengan Amazon EC2 Auto Scaling untuk menskalakan instans EC2 berdasarkan permintaan. Untuk informasi selengkapnya, silakan lihat Dynamic Scaling di Panduan Pengguna Amazon EC2 Auto Scaling.
- AWS CloudTrailmemungkinkan Anda untuk memantau panggilan yang dilakukan ke Amazon CloudWatch API untuk akun Anda, termasuk panggilan yang dilakukan oleh AWS Management Console, AWS CLI, dan layanan lainnya. Saat CloudTrail logging diaktifkan, CloudWatch tulis file log ke bucket Amazon S3 yang Anda tentukan saat Anda mengonfigurasi. CloudTrail Untuk informasi selengkapnya, lihat Mencatat panggilan CloudWatch API Amazon dengan AWS CloudTrail.
- AWS Identity and Access Management (IAM) adalah layanan web yang membantu Anda mengontrol akses ke AWS sumber daya dengan aman bagi pengguna Anda. Gunakan IAM untuk mengendalikan orang yang dapat menggunakan sumber daya AWS Anda (autentikasi) dan sumber daya apa yang dapat digunakan dengan cara apa (otorisasi). Untuk informasi selengkapnya, lihat Manajemen identitas dan akses untuk Amazon CloudWatch.

Bagaimana Amazon CloudWatch bekerja

Amazon pada dasarnya CloudWatch adalah repositori metrik. AWS Layanan—seperti Amazon EC2—menempatkan metrik ke dalam repositori, dan Anda mengambil statistik berdasarkan metrik tersebut. Jika Anda menempatkan metrik kustom Anda sendiri ke dalam repositori, Anda juga dapat mengambil statistik pada metrik ini.



Anda dapat menggunakan metrik untuk menghitung statistik dan kemudian menyajikan data secara grafis di konsol. CloudWatch Untuk informasi selengkapnya tentang AWS sumber daya lain yang menghasilkan dan mengirim metrik CloudWatch, lihatAWS layanan yang mempublikasikan CloudWatch metrik.

Anda dapat mengonfigurasi tindakan alarm untuk mengakhiri, memulai, atau mengakhiri instans Amazon EC2 ketika kriteria-kriteria tertentu telah terpenuhi. Selain itu, Anda dapat membuat alarm yang memulai tindakan Amazon EC2 Auto Scaling dan Amazon Simple Notification Service (Amazon SNS) atas nama Anda. Untuk informasi selengkapnya tentang membuat CloudWatch alarm, lihat Alarm.

AWS Sumber daya komputasi awan ditempatkan di fasilitas pusat data yang sangat tersedia. Untuk menyediakan skalabilitas dan keandalan tambahan, setiap fasilitas pusat data terletak di wilayah geografis tertentu, dikenal sebagai Wilayah. Setiap Wilayah dirancang untuk terisolasi sepenuhnya dari Wilayah lainnya, untuk mencapai isolasi dan stabilitas kegagalan semaksimal mungkin. Metrik disimpan secara terpisah di Wilayah, tetapi Anda dapat menggunakan fungsionalitas CloudWatch lintas wilayah untuk mengumpulkan statistik dari Wilayah yang berbeda. Untuk informasi selengkapnya, silakan lihat Konsol CloudWatch lintas akun lintas Wilayah dan Wilayah dan Titik Akhir di Referensi Umum Amazon Web Services.

CloudWatch Konsep Amazon

Terminologi dan konsep berikut sangat penting untuk pemahaman dan penggunaan Amazon CloudWatch Anda:

- Namespace
- Metrik
- Dimensi
- Resolusi
- Statistik
- Persentil
- Alarm

Untuk informasi tentang kuota layanan untuk CloudWatch metrik, alarm, permintaan API, dan notifikasi email alarm, lihat CloudWatch kuota layanan.

Namespace

Namespace adalah wadah untuk CloudWatch metrik. Metrik di ruang nama yang berbeda diisolasi satu sama lain, sehingga metrik dari aplikasi yang berbeda tidak digabungkan secara keliru menjadi statistik yang sama.

Tidak ada ruang nama bawaan. Anda harus menentukan namespace untuk setiap titik data yang Anda publikasikan. CloudWatch Anda dapat menentukan nama dari ruang nama ketika membuat metrik. Nama-nama ini harus berisi karakter ASCII yang benar, dan 255 karakter atau kurang. Karakter yang mungkin adalah: karakter alfanumerik (0-9a-za-Z), periode (.), tanda hubung (-), garis bawah (_), garis miring maju (/), hash (#), titik dua (:), dan karakter spasi. Sebuah ruang nama harus berisi setidaknya satu karakter non-spasi.

AWS Ruang nama biasanya menggunakan konvensi penamaan berikut:. `AWS/service` Misalnya, Amazon EC2 menggunakan ruang nama `AWS/EC2`. Untuk daftar AWS ruang nama, lihat. AWS layanan yang mempublikasikan CloudWatch metrik

Metrik

Metrik adalah konsep dasar dalam CloudWatch. Metrik mewakili kumpulan titik data yang diurutkan waktu yang dipublikasikan ke CloudWatch. Pikirkan metrik sebagai variabel untuk memantau, dan

titik data sebagai representasi nilai-nilai variabel tersebut dari waktu ke waktu. Misalnya, penggunaan CPU dari instans EC2 tertentu adalah satu metrik yang disediakan oleh Amazon EC2. Titik data itu sendiri dapat berasal dari aplikasi atau aktivitas bisnis tempat Anda mengumpulkan data.

Secara default, banyak AWS layanan menyediakan metrik tanpa biaya untuk sumber daya (seperti instans Amazon EC2, volume Amazon EBS, dan instans Amazon RDS DB). Untuk biaya, Anda juga dapat mengaktifkan pemantauan terperinci untuk beberapa sumber daya, seperti instans Amazon EC2 Anda, atau menerbitkan metrik aplikasi Anda sendiri. Untuk metrik kustom, Anda dapat menambahkan titik-titk data pada urutan tertentu, dan pada tingkat yang dipilih. Anda dapat mengambil statistik tentang titik data tersebut sebagai kumpulan data deret waktu yang diurutkan.

Metrik hanya ada di Wilayah tempat data dibuat. Metrik tidak dapat dihapus, tetapi akan kedaluwarsa secara otomatis setelah 15 bulan jika tidak ada data baru yang diterbitkan kepada mereka. Titik data yang lebih lama dari 15 bulan kedaluwarsa secara bergulir; ketika titik data yang baru masuk, maka data yang lebih lama dari 15 bulan akan hilang.

Metrik didefinisikan secara unik berdasarkan nama, ruang nama, dan dimensi nol atau lebih. Setiap titik data dalam metrik memiliki stempel waktu, dan (secara opsional) satuan pengukuran. Anda dapat mengambil statistik dari CloudWatch metrik apa pun.

Untuk informasi selengkapnya, silakan lihat Lihat metrik yang tersedia dan Menerbitkan metrik kustom.

Stempel Waktu

Setiap titik data metrik harus dikaitkan dengan stempel waktu. Stempel waktu dapat sampai dua minggu di masa lalu dan sampai dua jam di masa mendatang. Jika Anda tidak memberikan cap waktu, CloudWatch buat stempel waktu untuk Anda berdasarkan waktu titik data diterima.

Stempel waktu adalah objek `dateTime`, dengan tanggal lengkap ditambah jam, menit, dan detik (misalnya, 2016-10-31T23:59:59Z). Untuk informasi selengkapnya, silakan lihat dateTime. Meskipun tidak diperlukan, kami menyarankan agar Anda menggunakan Waktu Universal Terkoordinasi (UTC). Ketika Anda mengambil statistik dari CloudWatch, semua waktu dalam UTC.

CloudWatch alarm memeriksa metrik berdasarkan waktu saat ini di UTC. Metrik khusus yang dikirim ke CloudWatch dengan stempel waktu selain waktu UTC saat ini dapat menyebabkan alarm untuk menampilkan status Data Tidak Cukup atau mengakibatkan alarm tertunda.

Retensi metrik

CloudWatch mempertahankan data metrik sebagai berikut:

- Titik data dengan periode kurang dari 60 detik tersedia selama 3 jam. Titik data ini adalah metrik kustom resolusi tinggi.
- Titik data dengan periode 60 detik (1 menit) tersedia selama 15 hari
- Titik data dengan periode 300 detik (5 menit) tersedia selama 63 hari
- Titik data dengan periode 3.600 detik (1 jam) tersedia selama 455 hari (15 bulan)

Titik data yang awalnya diterbitkan dengan periode lebih singkat dikumpulkan bersama untuk penyimpanan jangka panjang. Sebagai contoh, jika Anda mengumpulkan data menggunakan periode 1 menit, data tetap tersedia selama 15 hari dengan resolusi 1 menit. Setelah 15 hari, data ini masih tersedia, tetapi dikumpulkan dan dapat diambil hanya dengan resolusi 5 menit. Setelah 63 hari, data akan dikumpulkan lebih lanjut dan tersedia dengan resolusi 1 jam.

> **Note**
>
> Metrik yang tidak memiliki titik data baru dalam dua minggu terakhir tidak muncul di konsol. Mereka juga tidak muncul ketika Anda mengetik nama metrik atau nama dimensi mereka di kotak pencarian dalam tab Semua metrik di konsol, dan tidak dikembalikan di dalam hasil dari perintah list-metrics. Cara terbaik untuk mengambil metrik ini adalah dengan get-metric-statisticsperintah get-metric-dataatau di. AWS CLI

Dimensi

Dimensi adalah pasangan nama/nilai yang merupakan bagian dari identitas metrik. Anda dapat menetapkan hingga 30 dimensi ke sebuah metrik.

Setiap metrik memiliki karakteristik tertentu yang menjelaskannya, dan Anda dapat berpikir dimensi sebagai kategori untuk karakteristik tersebut. Dimensi membantu Anda merancang struktur untuk rencana statistik. Karena dimensi adalah bagian dari pengidentifikasi unik untuk metrik, setiap kali Anda menambahkan pasangan nama/nilai unik ke salah satu metrik, Anda membuat variasi baru dari metrik tersebut.

AWS layanan yang mengirim data untuk CloudWatch melampirkan dimensi ke setiap metrik. Anda dapat menggunakan dimensi untuk memfilter hasil yang CloudWatch kembali. Misalnya, Anda dapat memperoleh statistik untuk instans EC2 tertentu dengan menetapkan `InstanceId` dimensi ketika mencari metrik.

Untuk metrik yang dihasilkan oleh AWS layanan tertentu, seperti Amazon EC2 CloudWatch , dapat mengumpulkan data di seluruh dimensi. Misalnya, jika Anda mencari metrik di `AWS/EC2` namespace tetapi tidak menentukan dimensi apa pun, CloudWatch agregat semua data untuk metrik yang ditentukan untuk membuat statistik yang Anda minta. CloudWatch tidak menggabungkan seluruh dimensi untuk metrik kustom Anda.

Kombinasi dimensi

CloudWatch memperlakukan setiap kombinasi dimensi yang unik sebagai metrik terpisah, bahkan jika metrik memiliki nama metrik yang sama. Anda hanya dapat mengambil statistik menggunakan kombinasi dimensi yang diterbitkan secara khusus. Ketika Anda mengambil statistik, tentukan nilai yang sama untuk ruang nama, nama metrik, dan parameter dimensi yang digunakan ketika metrik tersebut dibuat. Anda juga dapat menentukan waktu mulai dan akhir untuk digunakan CloudWatch untuk agregasi.

Misalnya, Anda menerbitkan empat metrik berbeda yang dinamai ServerStats di DataCenterMetric namespace dengan properti berikut:

```
Dimensions: Server=Prod, Domain=Frankfurt, Unit: Count, Timestamp:
 2016-10-31T12:30:00Z, Value: 105
Dimensions: Server=Beta, Domain=Frankfurt, Unit: Count, Timestamp:
 2016-10-31T12:31:00Z, Value: 115
Dimensions: Server=Prod, Domain=Rio,       Unit: Count, Timestamp:
 2016-10-31T12:32:00Z, Value: 95
Dimensions: Server=Beta, Domain=Rio,       Unit: Count, Timestamp:
 2016-10-31T12:33:00Z, Value: 97
```

Jika Anda hanya menerbitkan empat metrik tersebut, Anda dapat mengambil statistik untuk kombinasi dimensi berikut:

- `Server=Prod,Domain=Frankfurt`
- `Server=Prod,Domain=Rio`
- `Server=Beta,Domain=Frankfurt`
- `Server=Beta,Domain=Rio`

Anda tidak dapat mengambil statistik untuk dimensi berikut atau jika Anda tidak menentukan dimensi. (Pengecualiannya adalah dengan menggunakan perhitungan metrik fungsi SEARCH, yang dapat

mengambil statistik untuk beberapa metrik. Untuk informasi selengkapnya, silakan lihat Gunakan ekspresi pencarian pada grafik.)

- `Server=Prod`
- `Server=Beta`
- `Domain=Frankfurt`
- `Domain=Rio`

Resolusi

Setiap metrik adalah salah satu dari berikut:

- Resolusi standar, dengan data yang memiliki granularitas satu menit
- Resolusi tinggi, dengan data pada granularitas satu detik

Metrik yang dihasilkan oleh AWS layanan adalah resolusi standar secara default. Ketika menerbitkan sebuah metrik kustom, Anda dapat menetapkannya sebagai resolusi standar atau resolusi tinggi. Saat Anda menerbitkan metrik resolusi tinggi, CloudWatch simpan dengan resolusi 1 detik, dan Anda dapat membaca dan mengambilnya dengan jangka waktu 1 detik, 5 detik, 10 detik, 30 detik, atau kelipatan 60 detik.

Metrik resolusi tinggi dapat memberi wawasan yang lebih cepat tentang aktivitas submenit aplikasi Anda. Ingatlah bahwa setiap panggilan `PutMetricData` untuk sebuah metrik kustom yang dikenakan biaya, sehingga memanggil `PutMetricData` lebih sering pada sebuah metrik resolusi tinggi akan dapat mengakibatkan biaya yang lebih tinggi. Untuk informasi selengkapnya tentang CloudWatch harga, lihat CloudWatch Harga Amazon.

Jika Anda mengatur sebuah alarm pada metrik resolusi tinggi, maka Anda dapat menentukan alarm resolusi tinggi dengan periode 10 detik atau 30 detik, atau Anda dapat mengatur sebuah alarm biasa dengan periode lebih dari 60 detik. Terdapat biaya yang lebih tinggi untuk alarm resolusi tinggi dengan periode 10 atau 30 detik.

Statistik

Statistik adalah agregasi data metrik selama periode waktu tertentu. CloudWatch menyediakan statistik berdasarkan titik data metrik yang disediakan oleh data kustom Anda atau disediakan oleh

AWS layanan lain untuk CloudWatch. Agregasi dilakukan menggunakan namespace, nama metrik, dimensi, dan unit titik data dari ukuran, dalam periode waktu yang Anda tentukan.

Untuk definisi rinci dari statistik yang didukung oleh CloudWatch, lihatCloudWatch definisi statistik.

Unit

Setiap statistik memiliki satuan pengukuran. Contoh satuan termasuk `Bytes`, `Seconds`, `Count`, dan `Percent`. Untuk daftar lengkap unit yang CloudWatch mendukung, lihat tipe MetricDatumdata di Referensi Amazon CloudWatch API.

Anda dapat menentukan sebuah satuan ketika membuat metrik kustom. Jika Anda tidak menentukan unit, CloudWatch gunakan `None` sebagai unit. Satuan membantu memberikan makna konseptual pada data Anda. Meskipun CloudWatch tidak ada signifikansi pada unit secara internal, aplikasi lain dapat memperoleh informasi semantik berdasarkan unit.

Titik data metrik yang menentukan satuan pengukuran dikumpulkan secara terpisah. Ketika Anda mendapatkan statistik tanpa menentukan unit, CloudWatch agregat semua titik data dari unit yang sama bersama-sama. Jika Anda memiliki dua metrik identik dengan satuan yang berbeda, dua aliran data yang terpisah akan dikembalikan, satu untuk setiap satuan.

Periode

Periode adalah lamanya waktu yang terkait dengan CloudWatch statistik Amazon tertentu. Setiap statistik mewakili pengumpulan data metrik yang dikumpulkan selama periode waktu tertentu. Periode ditentukan dalam jumlah detik, dan nilai yang valid untuk periode adalah kelipatan 1, 5, 10, 30, atau 60. Misalnya, untuk menentukan periode enam menit, gunakan 360 sebagai nilai periode. Anda dapat menyesuaikan cara mengumpulkan data dengan membuat variasi durasi periode. Nilai default suatu periode adalah 60 detik. Sebuah periode bisa sesingkat satu detik, dan harus kelipatan 60 jika lebih besar dari nilai default 60 detik.

Hanya metrik kustom yang Anda tentukan dengan resolusi penyimpanan selama periode sub-menit dukungan 1 detik. Meskipun pilihan untuk mengatur periode di bawah 60 selalu tersedia di konsol, Anda harus memilih periode yang selaras dengan cara menyimpan metrik. Untuk informasi selengkapnya tentang metrik yang mendukung periode submenit, silakan lihat Metrik resolusi tinggi.

Ketika Anda mengambil statistik, Anda dapat menentukan periode, waktu mulai, dan waktu selesai. Parameter ini menentukan panjang keseluruhan waktu yang terkait dengan statistik. Nilai bawaan untuk waktu mulai dan waktu selesai akan memberi Anda statistik selama satu jam terakhir. Nilai

yang Anda tentukan untuk waktu mulai dan waktu akhir menentukan berapa banyak periode yang CloudWatch dikembalikan. Misalnya, mengambil statistik menggunakan nilai bawaan untuk periode, waktu mulai, dan waktu akhir mengembalikan kumpulan rangkaian statistik untuk setiap menit dari jam sebelumnya. Jika Anda lebih memilih statistik yang dikumpulkan dalam blok sepuluh menit, tentukan periode 600. Untuk statistik yang dikumpulkan selama satu jam penuh, tentukan periode 3.600.

Ketika statistik dikumpulkan selama periode waktu tertentu, statistik diberi stempel waktu yang sesuai dengan awal periode. Misalnya, data yang dikumpulkan dari pukul 19.00 hingga 20.00 diberi stempel pada pukul 19.00. Selain itu, data yang dikumpulkan antara pukul 19:00 dan 20:00 mulai terlihat pada pukul 19:00, kemudian nilai data agregat tersebut dapat berubah saat CloudWatch mengumpulkan lebih banyak sampel selama periode tersebut.

Periode juga penting untuk CloudWatch alarm. Saat Anda membuat alarm untuk memantau metrik tertentu, Anda meminta CloudWatch untuk membandingkan metrik tersebut dengan nilai ambang batas yang Anda tentukan. Anda memiliki kontrol yang luas atas bagaimana CloudWatch membuat perbandingan itu. Anda tidak hanya dapat menentukan periode ketika perbandingan tersebut dibuat, tetapi Anda juga dapat menentukan seberapa banyak periode evaluasi yang digunakan untuk sampai pada kesimpulan. Misalnya, jika Anda menentukan tiga periode evaluasi, CloudWatch bandingkan jendela tiga titik data. CloudWatch hanya memberi tahu Anda jika titik data tertua melanggar dan yang lainnya melanggar atau hilang.

Agregasi

Amazon CloudWatch mengumpulkan statistik sesuai dengan panjang periode yang Anda tentukan saat mengambil statistik. Anda dapat mempublikasikan titik data sebanyak yang Anda inginkan dengan stempel waktu yang sama atau serupa. CloudWatch agregat mereka sesuai dengan panjang periode yang ditentukan. CloudWatch tidak secara otomatis menggabungkan data di seluruh Wilayah, tetapi Anda dapat menggunakan matematika metrik untuk menggabungkan metrik dari Wilayah yang berbeda.

Anda dapat mempublikasikan titik data untuk metrik yang tidak hanya berbagi stempel waktu yang sama, tetapi juga namespace dan dimensi yang sama. CloudWatch mengembalikan statistik agregat untuk titik-titik data tersebut. Anda juga dapat memublikasikan beberapa titik data untuk metrik yang sama atau berbeda, dengan stempel waktu kapan pun.

Untuk set data yang besar, Anda dapat memasukkan rangkaian data yang sudah digabungkan sebelumnya yang dipanggil sebuah set statistik. Dengan set statistik, Anda memberikan CloudWatch Min, Max, Sum, dan SampleCount untuk sejumlah titik data. Hal ini umumnya digunakan ketika Anda

perlu mengumpulkan data beberapa kali dalam satu menit. Misalnya, anggap Anda memiliki metrik untuk latensi permintaan sebuah halaman web. Tidak masuk akal untuk menerbitkan data dengan setiap halaman web yang dikunjungi. Kami menyarankan Anda mengumpulkan latensi semua klik ke halaman web itu, menggabungkannya sekali dalam satu menit, dan mengirim statistik yang disetel ke. CloudWatch

Amazon CloudWatch tidak membedakan sumber metrik. Jika Anda mempublikasikan metrik dengan namespace dan dimensi yang sama dari sumber yang berbeda, CloudWatch perlakukan ini sebagai metrik tunggal. Hal ini dapat berguna untuk metrik layanan dalam sebuah sistem yang terdistribusi dan terskala. Misalnya, semua host dalam aplikasi server web dapat mempublikasikan metrik identik yang mewakili latensi permintaan yang mereka proses. CloudWatchmemperlakukan ini sebagai metrik tunggal, memungkinkan Anda untuk mendapatkan statistik untuk minimum, maksimum, rata-rata, dan jumlah semua permintaan di seluruh aplikasi Anda.

Persentil

Persentil menunjukkan posisi relatif dari nilai dalam sebuah set data. Misalnya, persentil ke-95 berarti bahwa 95 persen data lebih rendah dari nilai ini dan 5 persen data lebih tinggi dari nilai ini. Persentil membantu Anda mendapatkan pemahaman yang lebih baik tentang distribusi data metrik Anda.

Persentil sering kali digunakan untuk mengisolasi anomali. Dalam distribusi normal, 95 persen data berada dalam dua standar deviasi dari rata-rata dan 99,7 persen data berada dalam tiga standar deviasi dari rata-rata. Setiap data yang berada di luar tiga standar deviasi sering kali dianggap sebagai anomali karena data sangat berbeda dari nilai rata-rata. Misalnya, bayangkan Anda memantau pemanfaatan CPU dari instans EC2 untuk memastikan bahwa pelanggan Anda memiliki pengalaman yang baik. Jika Anda memantau rata-rata, hal ini dapat menyembunyikan anomali. Jika Anda memantau secara maksimal, satu anomali saja bisa membuat hasilnya tidak sesuai. Dengan persentil, Anda dapat memantau pemanfaatan CPU ke-95 untuk memeriksa instans dengan beban yang luar biasa berat.

Beberapa CloudWatch metrik mendukung persentil sebagai statistik. Untuk metrik ini, Anda dapat memantau sistem dan aplikasi menggunakan persentil seperti ketika menggunakan CloudWatch statistik (Rata-rata, Minimum, Maksimum, dan Jumlah). Misalnya, ketika Anda membuat alarm, Anda dapat menggunakan persentil sebagai fungsi statistik. Anda dapat menentukan persentil dengan hingga sepuluh tempat desimal (misalnya, p95.0123456789).

Statistik persentil tersedia untuk metrik kustom selama Anda menerbitkan titik data mentah yang tidak terangkum untuk metrik kustom Anda. Statistik persentil tidak tersedia untuk metrik ketika nilai metrik merupakan angka negatif.

CloudWatch membutuhkan titik data mentah untuk menghitung persentil. Jika Anda menerbitkan data menggunakan set statistik, Anda hanya dapat mengambil statistik persentil untuk data ini jika salah satu kondisi berikut benar:

- SampleCount Nilai himpunan statistik adalah 1 dan Min, Max, dan Jumlah semuanya sama.
- Min dan Max sama, dan Jumlah sama dengan Min dikalikan dengan. SampleCount

AWS Layanan berikut mencakup metrik yang mendukung statistik persentil.

- API Gateway
- Penyeimbang Beban Aplikasi
- Amazon EC2
- Penyeimbang Beban Elastis
- Kinesis
- Amazon RDS

CloudWatch juga mendukung rata-rata yang dipangkas dan statistik kinerja lainnya, yang dapat memiliki penggunaan yang sama sebagai persentil. Untuk informasi selengkapnya, lihat CloudWatch definisi statistik.

Alarm

Anda dapat menggunakan alarm untuk secara otomatis memulai tindakan atas nama Anda. Alarm mengawasi metrik tunggal selama periode waktu tertentu, dan melakukan satu atau beberapa tindakan tertentu, didasarkan pada nilai metrik relatif terhadap ambang batas dari waktu ke waktu. Tindakan ini adalah notifikasi yang dikirim ke topik Amazon SNS atau kebijakan Penskalaan Otomatis. Anda juga dapat menambahkan alarm ke dasbor.

Alarm memanggil tindakan untuk perubahan status berkelanjutan saja. CloudWatch alarm tidak memanggil tindakan hanya karena mereka berada dalam keadaan tertentu. Status harus diubah dan dipertahankan selama jangka waktu tertentu.

Ketika membuat alarm, pilih periode pemantauan alarm yang lebih besar atau sama dengan resolusi metrik. Misalnya, pemantauan dasar untuk Amazon EC2 menyediakan metrik untuk instans Anda setiap 5 menit. Ketika mengatur alarm pada metrik pemantauan dasar, pilih periode selama paling sedikit 300 detik (5 menit). Pemantauan terperinci untuk Amazon EC2 menyediakan metrik untuk

instans Anda dengan resolusi 1 menit. Ketika mengatur alarm pada metrik pemantauan terperinci, pilih periode selama paling sedikit 60 detik (1 menit).

Jika Anda mengatur sebuah alarm pada metrik resolusi tinggi, maka Anda dapat menentukan alarm resolusi tinggi dengan periode 10 detik atau 30 detik, atau Anda dapat mengatur sebuah alarm biasa dengan periode lebih dari 60 detik. Ada beban yang lebih tinggi untuk alarm-alarm dengan resolusi tinggi. Untuk informasi selengkapnya tentang metrik resolusi tinggi, silakan lihat Menerbitkan metrik kustom.

Untuk informasi selengkapnya, silakan lihat Menggunakan CloudWatch alarm Amazon dan Buat sebuah alarm dari metrik pada grafik.

Penagihan dan biaya

Untuk informasi selengkapnya tentang CloudWatch harga, lihat CloudWatch Harga Amazon.

Untuk informasi yang dapat membantu Anda menganalisis tagihan Anda dan mungkin mengoptimalkan dan mengurangi biaya, silakan lihat CloudWatch penagihan dan biaya.

CloudWatch Sumber daya Amazon

Sumber daya terkait berikut dapat membantu Anda ketika bekerja dengan layanan ini.

Sumber daya	Deskripsi
Amazon CloudWatch FAQ	Pertanyaan Umum mencakup pertanyaan paling atas yang telah ditanyakan pengembang tentang produk ini.
AWS Pusat Pengembang	Titik awal utama untuk menemukan dokumentasi, contoh kode, catatan rilis, dan informasi lain untuk membantu Anda membangun aplikasi inovatif AWS.
AWS Management Console	Konsol ini memungkinkan Anda untuk melakukan sebagian besar fungsi Amazon CloudWatch dan berbagai AWS penawaran lainnya tanpa pemrograman.
Forum CloudWatch Diskusi Amazon	Forum berbasis komunitas bagi pengembang untuk mendiskusikan pertanyaan teknis yang terkait dengan Amazon. CloudWatch

Sumber daya	Deskripsi
AWS Support	Hub untuk membuat dan mengelola AWS Support kasus Anda. Juga termasuk tautan ke sumber daya bermanfaat lainnya, seperti forum, FAQ teknis, status kesehatan layanan, dan Trusted AWS Advisor.
Informasi CloudWatch produk Amazon	Halaman web utama untuk informasi tentang Amazon CloudWatch.
Hubungi Kami	Titik kontak pusat untuk pertanyaan tentang AWS penagihan, akun, acara, penyalahgunaan, dll.

Menyiapkan

Untuk menggunakan Amazon CloudWatch Anda memerlukan sebuah akun AWS. Akun AWS Anda akan memungkinkan Anda untuk menggunakan layanan-layanan (misalnya, Amazon EC2) untuk menghasilkan metrik-metrik yang dapat Anda tampilkan di konsol CloudWatch, antarmuka berbasis-web titik-dan-klik. Selain itu, Anda juga dapat melakukan instalasi dan mengonfigurasi antarmuka baris perintah (CLI) AWS.

Daftar Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk secara online.

   Anda akan diminta untuk menerima panggilan telepon dan memasukkan kode verifikasi pada keypad telepon sebagai bagian dari prosedur pendaftaran.

   Saat Anda mendaftar Akun AWS, Pengguna root akun AWS dibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya dalam akun. Sebagai praktik terbaik keamanan, tetapkan akses administratif ke pengguna administratif, dan hanya gunakan pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS mengirimi Anda email konfirmasi setelah proses pendaftaran selesai. Anda dapat melihat aktivitas akun saat ini dan mengelola akun dengan mengunjungi https://aws.amazon.com/ dan memilih Akun Saya.

Membuat pengguna administratif

Setelah mendaftar Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat sebuah pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Mengamankan Pengguna root akun AWS Anda

1. Masuk ke AWS Management Console sebagai pemilik akun dengan memilih Pengguna root dan memasukkan alamat email Akun AWS Anda. Pada halaman berikutnya, masukkan kata sandi Anda.

   Untuk bantuan masuk menggunakan pengguna root, silakan lihat Masuk sebagai pengguna root dalam Panduan Pengguna AWS Sign-In.
2. Mengaktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

   Untuk petunjuk, silakan lihat Mengaktifkan perangkat MFA virtual untuk pengguna root Akun AWS Anda (konsol) dalam Panduan Pengguna IAM.

Membuat pengguna administratif

1. Aktifkan Pusat Identitas IAM.

   Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center.
2. Di Pusat Identitas IAM, berikan akses administratif ke sebuah pengguna administratif.

   Untuk mendapatkan tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, silakan lihat Mengonfigurasi akses pengguna dengan Direktori Pusat Identitas IAM default di Panduan Pengguna AWS IAM Identity Center.

Masuk sebagai pengguna administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email saat Anda membuat pengguna Pusat Identitas IAM.

  Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, silakan lihat Masuk ke portal akses AWS dalam Panduan Pengguna AWS Sign-In.

Masuk ke Konsol Amazon CloudWatch

Cara masuk ke Konsol Amazon CloudWatch

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.

2. Jika diperlukan, Anda bisa menggunakan bilah navigasi untuk mengubah wilayah ke Wilayah tempat Anda memiliki sumber daya AWS.
3. Meskipun ini pertama kalinya Anda menggunakan konsol CloudWatch, Metrik Anda sudah dapat melaporkan metrik, karena Anda telah menggunakan sebuah produk AWS yang secara otomatis mendorong metrik ke Amazon CloudWatch secara gratis. Layanan-layanan lainnya mengharuskan Anda untuk mengaktifkan metrik.

   Jika Anda tidak memiliki alarm, bagian Alarm Anda akan memiliki sebuah tombol Buat Alarm.

Menyiapkan AWS CLI

Anda dapat menggunakan AWS CLI atau Amazon CloudWatch CLI untuk melaksanakan perintah-perintah CloudWatch. Perlu diperhatikan bahwa AWS CLI menggantikan CloudWatch CLI; kami menyertakan fitur-fitur CloudWatch baru hanya di AWS CLI.

Untuk informasi tentang cara melakukan instalasi dan mengonfigurasi AWS CLI, silakan lihat Menyiapkan dengan AWS Antarmuka Baris Perintah di AWS Command Line Interface Panduan Pengguna.

Untuk informasi tentang cara melakukan instalasi dan mengonfigurasi Amazon CloudWatch CLI, silakan lihat Mengatur Antarmuka Baris Perintah di Referensi Amazon CloudWatch CLI.

Memulai dengan Amazon CloudWatch

Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.

Halaman beranda CloudWatch ikhtisar muncul.

Ikhtisar tersebut akan menampilkan item-item berikut ini, disegarkan kembali secara otomatis.

- Alarm berdasarkan AWS layanan menampilkan daftar AWS layanan yang Anda gunakan di akun Anda, bersama dengan status alarm di layanan tersebut. Di samping itu, dua atau empat alarm di akun Anda ditampilkan. Jumlahnya tergantung pada berapa banyak AWS layanan yang Anda gunakan. Alarm-alarm yang ditampilkan tersebut adalah alarm-alarm yang berada dalam status ALARM atau yang terakhir kali diubah.

  Area atas ini membantu Anda menilai kesehatan AWS layanan Anda dengan cepat, dengan melihat status alarm di setiap layanan dan alarm yang baru-baru ini berubah keadaan. Hal ini akan membantu Anda dalam memantau dan mendiagnosis masalah-masalah dengan cepat.
- Di bawah area ini adalah dasbor bawaan, jika ada. Dasbor default adalah dasbor khusus yang telah Anda buat dan beri nama CloudWatch-Default. Ini adalah cara mudah bagi Anda untuk menambahkan metrik tentang layanan atau aplikasi kustom Anda sendiri ke halaman ikhtisar, atau untuk memajukan metrik kunci tambahan dari AWS layanan yang paling ingin Anda pantau.

> **Note**
>
> Dasbor otomatis di CloudWatch halaman beranda hanya menampilkan informasi dari akun saat ini, meskipun akun tersebut adalah akun pemantauan yang disiapkan untuk pengamatan CloudWatch lintas akun. Untuk mendapatkan informasi tentang cara membuat dasbor lintas akun kustom, silakan lihat CloudWatch dasbor observabilitas lintas akun.

Dari ikhtisar ini, Anda dapat melihat dasbor metrik lintas layanan dari beberapa AWS layanan, atau memfokuskan tampilan Anda ke grup sumber daya tertentu atau layanan tertentu AWS . Hal ini akan memungkinkan Anda untuk mempersempit tampilan ke sekelompok bagian dari sumber daya tertentu yang menjadi minat Anda. Untuk informasi selengkapnya, silakan lihat bagian-bagian berikut ini.

Lihat dasbor pra-bangun otomatis untuk satu layanan

Untuk melihat dasbor pra-bangun otomatis untuk satu layanan

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.

   Halaman beranda akan ditampilkan.
2. Di panel navigasi kiri, pilih Dasbor.
3. Pilih tab Dasbor otomatis, lalu pilih layanan yang ingin Anda lihat.
4. Untuk beralih melihat alarm untuk layanan ini, pilih kotak centang untuk Dalam alarm, Data tidak mencukupi, atau OK di dekat bagian atas layar tempat nama layanan saat ini ditampilkan.
5. Ketika Anda melihat metrik, Anda dapat berfokus pada satu metrik tertentu dengan beberapa cara:

   a. Untuk melihat detail selengkapnya tentang metrik-metrik tersebut dalam grafik, arahkan kursor ke grafik, dan pilih ikon tindakan, Lihat dalam metrik.

      Grafik tersebut kemudian akan muncul di sebuah tab baru yang menampilkan metrik-metrik yang relevan yang tercantum di bawah grafik tersebut. Anda dapat menyesuaikan tampilan grafik ini, juga mengubah metrik-metrik dan sumber daya yang ditampilkan, statistik, periode, dan faktor lainnya yang ditampilkan untuk mendapatkan pemahaman lebih baik tentang situasi saat ini.

b. Anda dapat melihat peristiwa log dari rentang waktu yang ditunjukkan dalam grafik tersebut. Hal ini akan dapat membantu Anda menemukan peristiwa yang terjadi dalam infrastruktur Anda yang menyebabkan perubahan terhadap metrik-metrik yang tidak terduga.

   Untuk melihat peristiwa log, silakan Anda arahkan kursor ke grafik, dan pilih ikon tindakan, Lihat dalam log.

   Tampilan CloudWatch Log muncul di tab baru, menampilkan daftar grup log Anda. Untuk melihat peristiwa log di salah satu grup log ini yang terjadi selama rentang waktu yang ditunjukkan dalam grafik asli, silakan pilih grup log tersebut.

6. Ketika menampilkan alarm, Anda dapat berfokus pada alarm tertentu dengan beberapa cara:

   - Untuk melihat detail selengkapnya tentang sebuah alarm, silakan Anda arahkan kursor ke alarm, dan pilih ikon tindakan, Lihat dalam alarm.

     Tampilan alarm akan muncul pada tab yang baru dan menampilkan daftar alarm Anda, beserta detail tentang alarm yang sudah Anda pilih. Untuk melihat riwayat alarm ini, silakan pilih tab Riwayat.

7. Alarm akan selalu disegarkan satu kali per menit. Untuk menyegarkan tampilan, Anda pilih ikon segarkan (dua panah lengkung) yang ada di bagian kanan atas layar. Untuk mengubah tingkat penyegaran otomatis untuk item-item yang ada di layar selain alarm, pilihlah panah ke bawah yang ada di samping ikon segarkan dan kemudian pilih tingkat penyegaran. Anda juga dapat memilih untuk menonaktifkan penyegaran otomatis ini.
8. Untuk mengubah rentang waktu yang ditampilkan di semua grafik dan alarm yang saat ini ditampilkan, di samping Rentang waktu pada bagian atas layar, pilihlah rentang waktunya. Untuk memilih pilihan rentang waktu lebih banyak dari yang ditampilkan secara bawaan, pilih kustom.
9. Untuk kembali ke dasbor lintas layanan, silakan pilih Ikhtisar pada daftar yang ada di bagian atas layar yang saat ini menunjukkan layanan yang sedang Anda fokuskan.

   Atau, dari tampilan apa pun, Anda dapat memilih CloudWatchdi bagian atas layar untuk menghapus semua filter dan kembali ke halaman ikhtisar.

Lihat dasbor lintas layanan yang sudah dibuat sebelumnya

Anda dapat beralih ke layar dasbor Cross-service dan berinteraksi dengan dasbor untuk semua AWS layanan yang Anda gunakan. CloudWatch Konsol menampilkan dasbor Anda dalam urutan abjad dan menampilkan satu atau dua metrik utama di setiap dasbor.

> **Note**
>
> Jika Anda menggunakan lima AWS layanan atau lebih, CloudWatch Konsol tidak akan menampilkan dasbor Cross-service di layar Ikhtisar.

Cara membuka dasbor lintas layanan Cross-service

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.

   Anda diarahkan ke layar Ikhtisar.

2. Dari layar Ikhtisar, Anda harus memilih menu geser-turun yang bertuliskan Ikhtisar, lalu pilih Dasbor lintas layanan.

   Anda akan diarahkan ke layar dasbor Lintas layanan.

3. (Opsional) Jika Anda menggunakan antarmuka asli, Anda harus menggulir ke bagian Dasbor lintas layanan, lalu pilih Tampilkan dasbor Cross-service.

   Anda akan diarahkan ke layar dasbor Lintas layanan.

4. Anda dapat berfokus pada sebuah layanan tertentu dengan dua cara:

   a. Untuk melihat metrik utama lainnya pada sebuah layanan, Anda bisa memilih nama metrik tersebut dari daftar yang ada di bagian atas layar, tempat Dasbor lintas layanan ditampilkan saat ini. Atau, Anda juga dapat memilih Lihat Dasbor layanan yang ada di samping nama layanan.

      Sebuah dasbor otomatis untuk layanan tersebut akan ditampilkan, yang menunjukkan metrik-metrik lainnya untuk layanan tersebut. Selain itu, untuk beberapa layanan, bagian bawah dasbor layanan akan menampilkan sumber daya yang terkait dengan layanan tersebut. Anda dapat memilih salah satu sumber daya itu untuk konsol layanan tersebut dan berfokus lebih jauh pada sumber daya.

   b. Untuk melihat semua alarm yang terkait dengan sebuah layanan, Anda harus memilih tombol yang ada di sebelah kanan layar di samping nama layanan. Teks pada tombol ini menunjukkan seberapa banyak alarm yang telah Anda buat dalam layanan ini, dan apakah ada di antara alarm itu yang sedang dalam status ALARM.

Ketika alarm-alarm tersebut ditampilkan, beberapa alarm yang memiliki pengaturan serupa (seperti dimensi, ambang batas, atau periode) akan dapat ditampilkan dalam satu grafik tunggal.

Kemudian, Anda juga dapat melihat detail tentang sebuah alarm dan melihat riwayat dari alarm tersebut. Untuk melakukan hal itu, Anda harus mengarahkan pada grafik alarm, dan kemudian memilih ikon tindakan, Tampilan dalam alarm.

Tampilan alarm akan muncul pada tab browser yang baru, menampilkan daftar alarm Anda, beserta detail tentang alarm yang sudah Anda pilih. Untuk melihat riwayat alarm ini, silakan pilih tab Riwayat.

5. Anda dapat berfokus pada sumber daya yang ada di grup sumber daya tertentu. Untuk melakukan hal itu, Anda harus memilih grup sumber daya dari daftar yang ada di bagian atas halaman tempat Semua sumber daya ditampilkan.

   Untuk informasi selengkapnya, lihat Lihat dasbor pra-bangun untuk grup sumber daya.

6. Untuk mengubah rentang waktu yang ditampilkan di semua grafik dan alarm yang saat ini ditampilkan, Anda harus memilih rentang yang Anda inginkan yang ada di samping Rentang waktu pada bagian atas layar. Pilih kustom untuk memilih pilihan rentang waktu lebih banyak dari yang ditampilkan secara bawaan.
7. Alarm akan selalu disegarkan satu menit sekali. Untuk menyegarkan tampilan, Anda pilih ikon segarkan (dua panah lengkung) yang ada di bagian kanan atas layar. Untuk mengubah tingkat penyegaran otomatis untuk item-item yang ada di layar selain alarm, pilihlah panah ke bawah yang ada di samping ikon segarkan dan kemudian pilih tingkat penyegaran yang Anda inginkan. Anda juga dapat memilih untuk menonaktifkan penyegaran otomatis.

Hapus layanan dari dasbor lintas layanan

Anda dapat mencegah metrik dari sebuah layanan muncul pada dasbor lintas layanan. Hal ini akan membantu Anda dalam memfokuskan dasbor lintas layanan pada layanan yang paling ingin Anda pantau.

Jika Anda menghapus sebuah layanan dari dasbor lintas layanan, alarm untuk layanan tersebut akan tetap muncul di tampilan alarm Anda.

Cara menghapus metrik dari sebuah layanan dari dasbor lintas layanan

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.

   Halaman beranda akan muncul.
2. Pada bagian atas halaman, pada Ikhtisar, pilihlah layanan yang ingin Anda hapus.

   Tampilan akan berubah untuk menunjukkan metrik hanya dari layanan tersebut.
3. Pilih Tindakan, dan kemudian hapus centang pada kotak yang ada di samping Tampilkan di dasbor lintas layanan.

Lihat dasbor pra-bangun untuk grup sumber daya

Anda dapat memfokuskan tampilan Anda sehingga ia hanya menampilkan metrik-metrik dan alarm-alarm dari satu grup sumber daya. Dengan menggunakan grup sumber daya, Anda akan dapat menggunakan tag untuk mengorganisir proyek-proyek, berfokus pada bagian dari arsitektur, atau membedakan antara lingkungan produksi dan pengembangan Anda. Mereka juga memungkinkan Anda untuk fokus pada masing-masing kelompok sumber daya ini pada CloudWatch ikhtisar. Untuk informasi selengkapnya tentang ini, silakan lihat Apa Itu AWS Resource Groups?.

Ketika Anda berfokus pada sebuah grup sumber daya, layar tampilan hanya akan menampilkan layanan yang sumber dayanya telah ditandai sebagai bagian dari grup sumber daya ini. Area alarm terbaru hanya akan menunjukkan alarm yang dikaitkan dengan sumber daya yang menjadi bagian dari grup sumber daya. Selain itu, jika Anda telah membuat dasbor dengan nama CloudWatch-Default- ResourceGroupName, itu ditampilkan di area dasbor Default.

Anda dapat menelusuri lebih jauh dengan berfokus pada AWS layanan tunggal dan grup sumber daya pada saat yang bersamaan. Prosedur berikut hanya menjelaskan bagaimana fokus pada kelompok sumber daya.

Untuk berfokus pada grup sebuah sumber daya tunggal

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada bagian atas halaman, tempat Semua sumber daya ditampilkan, pilih sebuah grup sumber daya.
3. Untuk melihat metrik-metrik lainnya yang terkait dengan grup sumber daya ini, di dekat bagian bawah layar, pilih Lihat dasbor lintas layanan.

Dasbor lintas layanan akan ditampilkan dan hanya akan menunjukkan layanan yang terkait dengan grup sumber daya ini. Untuk masing-masing layanan, satu atau dua metrik utama akan ditampilkan.

4. Untuk mengubah rentang waktu yang ditampilkan di semua grafik dan alarm yang saat ini ditampilkan, di samping Rentang waktu pada bagian atas layar, Anda harus memilih rentang waktunya. Untuk memilih pilihan rentang waktu lebih banyak dari yang ditampilkan secara bawaan, pilih kustom.
5. Alarm akan selalu disegarkan satu kali per menit. Untuk menyegarkan tampilan, Anda pilih ikon segarkan (dua panah lengkung) yang ada di bagian kanan atas layar. Untuk mengubah tingkat penyegaran otomatis untuk item-item yang ada di layar selain alarm, pilihlah panah ke bawah yang ada di samping ikon segarkan dan kemudian pilih tingkat penyegaran. Anda juga dapat memilih untuk menonaktifkan penyegaran otomatis ini.
6. Untuk kembali menampilkan informasi tentang semua sumber daya yang ada di akun Anda, di dekat bagian atas layar tempat nama grup sumber daya saat ini ditampilkan, pilihlah Semua sumber daya.

Lihat dasbor lintas layanan yang sudah dibuat sebelumnya

Anda dapat beralih ke layar dasbor Cross-service dan berinteraksi dengan dasbor untuk semua AWS layanan yang Anda gunakan. CloudWatch Konsol menampilkan dasbor Anda dalam urutan abjad dan menampilkan satu atau dua metrik utama dari setiap layanan.

> **Note**
>
> Jika Anda menggunakan lima AWS layanan atau lebih, CloudWatch Konsol tidak akan menampilkan dasbor Cross-service di layar Ikhtisar.

Cara membuka dasbor lintas layanan Cross-service

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.

   Anda diarahkan ke layar Ikhtisar.
2. Dari layar Ikhtisar, Anda harus memilih menu geser-turun yang bertuliskan Ikhtisar, lalu pilih Dasbor lintas layanan.

Anda akan diarahkan ke layar dasbor Lintas layanan.

3. (Opsional) Jika Anda menggunakan antarmuka asli, Anda harus menggulir ke bagian Dasbor lintas layanan, lalu pilih Tampilkan dasbor Cross-service.

   Anda akan diarahkan ke layar dasbor Lintas layanan.

4. Anda dapat berfokus pada sebuah layanan tertentu dengan dua cara:

   a. Untuk melihat metrik utama lainnya pada sebuah layanan, Anda bisa memilih nama metrik tersebut dari daftar yang ada di bagian atas layar, tempat Dasbor lintas layanan ditampilkan saat ini. Atau, Anda juga dapat memilih Lihat Dasbor layanan yang ada di samping nama layanan.

      Sebuah dasbor otomatis untuk layanan tersebut akan ditampilkan, yang menunjukkan metrik-metrik lainnya untuk layanan tersebut. Selain itu, untuk beberapa layanan, bagian bawah dasbor layanan akan menampilkan sumber daya yang terkait dengan layanan tersebut. Anda dapat memilih salah satu sumber daya itu untuk konsol layanan tersebut dan berfokus lebih jauh pada sumber daya.

   b. Untuk melihat semua alarm yang terkait dengan sebuah layanan, Anda harus memilih tombol yang ada di sebelah kanan layar di samping nama layanan. Teks pada tombol ini menunjukkan seberapa banyak alarm yang telah Anda buat dalam layanan ini, dan apakah ada di antara alarm itu yang sedang dalam status ALARM.

      Ketika alarm-alarm tersebut ditampilkan, beberapa alarm yang memiliki pengaturan serupa (seperti dimensi, ambang batas, atau periode) akan dapat ditampilkan dalam satu grafik tunggal.

      Kemudian, Anda juga dapat melihat detail tentang sebuah alarm dan melihat riwayat dari alarm tersebut. Untuk melakukan hal itu, Anda harus mengarahkan pada grafik alarm, dan kemudian memilih ikon tindakan, Tampilan dalam alarm.

      Tampilan alarm akan muncul pada tab browser yang baru, menampilkan daftar alarm Anda, beserta detail tentang alarm yang sudah Anda pilih. Untuk melihat riwayat alarm ini, silakan pilih tab Riwayat.

5. Anda dapat berfokus pada sumber daya yang ada di grup sumber daya tertentu. Untuk melakukan hal itu, Anda harus memilih grup sumber daya dari daftar yang ada di bagian atas halaman tempat Semua sumber daya ditampilkan.

Untuk informasi selengkapnya, lihat Lihat dasbor pra-bangun untuk grup sumber daya.

6. Untuk mengubah rentang waktu yang ditampilkan di semua grafik dan alarm yang saat ini ditampilkan, Anda harus memilih rentang yang Anda inginkan yang ada di samping Rentang waktu pada bagian atas layar. Pilih kustom untuk memilih pilihan rentang waktu lebih banyak dari yang ditampilkan secara bawaan.
7. Alarm akan selalu disegarkan satu menit sekali. Untuk menyegarkan tampilan, Anda pilih ikon segarkan (dua panah lengkung) yang ada di bagian kanan atas layar. Untuk mengubah tingkat penyegaran otomatis untuk item-item yang ada di layar selain alarm, pilihlah panah ke bawah yang ada di samping ikon segarkan dan kemudian pilih tingkat penyegaran yang Anda inginkan. Anda juga dapat memilih untuk menonaktifkan penyegaran otomatis ini.

Menghapus sebuah layanan agar tidak muncul di dasbor lintas layanan

Anda dapat mencegah metrik dari sebuah layanan muncul pada dasbor lintas layanan. Hal ini akan membantu Anda dalam memfokuskan dasbor lintas layanan pada layanan yang paling ingin Anda pantau.

Jika Anda menghapus sebuah layanan dari dasbor lintas layanan, alarm untuk layanan tersebut akan tetap muncul di tampilan alarm Anda.

Cara menghapus metrik dari sebuah layanan dari dasbor lintas layanan

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.

   Halaman beranda akan muncul.
2. Pada bagian atas halaman, pada Ikhtisar, pilihlah layanan yang ingin Anda hapus.

   Tampilan akan berubah untuk menunjukkan metrik hanya dari layanan tersebut.
3. Pilih Tindakan, dan kemudian hapus centang pada kotak yang ada di samping Tampilkan di dasbor lintas layanan.

Lihat dasbor pra-bangun untuk satu layanan AWS

Di CloudWatch halaman beranda, Anda dapat memfokuskan tampilan ke satu AWS layanan. Anda dapat menelusuri lebih jauh dengan berfokus pada AWS layanan tunggal dan grup sumber daya

pada saat yang bersamaan. Prosedur berikut hanya menunjukkan bagaimana fokus pada AWS layanan.

Cara berfokus pada satu layanan tunggal

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.

   Halaman beranda akan ditampilkan.
2. Untuk Ikhtisar, di mana Ikhtisar saat ini ditampilkan di menu tarik-turun, pilih Dasbor layanan.
3. Pilih layanan yang ingin Anda fokuskan.

   Tampilan kemudian akan berubah dan menampilkan grafik dari metrik-metrik utama dari layanan yang telah Anda pilih.
4. Untuk beralih melihat alarm untuk layanan ini, pilih kotak centang untuk Dalam alarm, Data tidak mencukupi, atau OK di dekat bagian atas layar tempat nama layanan saat ini ditampilkan.
5. Ketika Anda melihat metrik, Anda dapat berfokus pada satu metrik tertentu dengan beberapa cara:

   a. Untuk melihat detail selengkapnya tentang metrik-metrik tersebut dalam grafik, arahkan kursor ke grafik, dan pilih ikon tindakan, Lihat dalam metrik.

      Grafik tersebut kemudian akan muncul di sebuah tab baru yang menampilkan metrik-metrik yang relevan yang tercantum di bawah grafik tersebut. Anda dapat menyesuaikan tampilan grafik ini, juga mengubah metrik-metrik dan sumber daya yang ditampilkan, statistik, periode, dan faktor lainnya yang ditampilkan untuk mendapatkan pemahaman lebih baik tentang situasi saat ini.

   b. Anda dapat melihat peristiwa log dari rentang waktu yang ditunjukkan dalam grafik tersebut. Hal ini akan dapat membantu Anda menemukan peristiwa yang terjadi dalam infrastruktur Anda yang menyebabkan perubahan terhadap metrik-metrik yang tidak terduga.

      Untuk melihat peristiwa log, silakan Anda arahkan kursor ke grafik, dan pilih ikon tindakan, Lihat dalam log.

      Tampilan CloudWatch Log muncul di tab baru, menampilkan daftar grup log Anda. Untuk melihat peristiwa log di salah satu grup log ini yang terjadi selama rentang waktu yang ditunjukkan dalam grafik asli, silakan pilih grup log tersebut.
6. Ketika menampilkan alarm, Anda dapat berfokus pada alarm tertentu dengan beberapa cara:

- Untuk melihat detail selengkapnya tentang sebuah alarm, silakan Anda arahkan kursor ke alarm, dan pilih ikon tindakan, Lihat dalam alarm.

  Tampilan alarm akan muncul pada tab yang baru dan menampilkan daftar alarm Anda, beserta detail tentang alarm yang sudah Anda pilih. Untuk melihat riwayat alarm ini, silakan pilih tab Riwayat.

7. Alarm akan selalu disegarkan satu kali per menit. Untuk menyegarkan tampilan, Anda pilih ikon segarkan (dua panah lengkung) yang ada di bagian kanan atas layar. Untuk mengubah tingkat penyegaran otomatis untuk item-item yang ada di layar selain alarm, pilihlah panah ke bawah yang ada di samping ikon segarkan dan kemudian pilih tingkat penyegaran. Anda juga dapat memilih untuk menonaktifkan penyegaran otomatis ini.
8. Untuk mengubah rentang waktu yang ditampilkan di semua grafik dan alarm yang saat ini ditampilkan, di samping Rentang waktu pada bagian atas layar, pilihlah rentang waktunya. Untuk memilih pilihan rentang waktu lebih banyak dari yang ditampilkan secara bawaan, pilih kustom.
9. Untuk kembali ke dasbor lintas layanan, silakan pilih Ikhtisar pada daftar yang ada di bagian atas layar yang saat ini menunjukkan layanan yang sedang Anda fokuskan.

   Atau, dari tampilan apa pun, Anda dapat memilih CloudWatchdi bagian atas layar untuk menghapus semua filter dan kembali ke halaman ikhtisar.

Lihat dasbor pra-bangun untuk grup sumber daya

Anda dapat memfokuskan tampilan Anda sehingga ia hanya menampilkan metrik-metrik dan alarm-alarm dari satu grup sumber daya. Dengan menggunakan grup sumber daya, Anda akan dapat menggunakan tag untuk mengorganisir proyek-proyek, berfokus pada bagian dari arsitektur, atau membedakan antara lingkungan produksi dan pengembangan Anda. Mereka juga memungkinkan Anda untuk fokus pada masing-masing kelompok sumber daya ini pada CloudWatch ikhtisar. Untuk informasi selengkapnya tentang ini, silakan lihat Apa Itu AWS Resource Groups?.

Ketika Anda berfokus pada sebuah grup sumber daya, layar tampilan hanya akan menampilkan layanan yang sumber dayanya telah ditandai sebagai bagian dari grup sumber daya ini. Area alarm terbaru hanya akan menunjukkan alarm yang dikaitkan dengan sumber daya yang menjadi bagian dari grup sumber daya. Selain itu, jika Anda telah membuat dasbor dengan nama CloudWatch-Default- ResourceGroupName, itu ditampilkan di area dasbor Default.

Anda dapat menelusuri lebih jauh dengan berfokus pada AWS layanan tunggal dan grup sumber daya pada saat yang bersamaan. Prosedur berikut ini akan menunjukkan cara berfokus pada sebuah grup sumber daya.

Untuk berfokus pada grup sebuah sumber daya tunggal

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada bagian atas halaman, tempat Semua sumber daya ditampilkan, pilih sebuah grup sumber daya.
3. Untuk melihat metrik-metrik lainnya yang terkait dengan grup sumber daya ini, di dekat bagian bawah layar, pilih Lihat dasbor lintas layanan.

   Dasbor lintas layanan akan ditampilkan dan hanya akan menunjukkan layanan yang terkait dengan grup sumber daya ini. Untuk masing-masing layanan, satu atau dua metrik utama akan ditampilkan.
4. Untuk mengubah rentang waktu yang ditampilkan di semua grafik dan alarm yang saat ini ditampilkan, di samping Rentang waktu pada bagian atas layar, Anda harus memilih rentang waktunya. Untuk memilih pilihan rentang waktu lebih banyak dari yang ditampilkan secara bawaan, pilih kustom.
5. Alarm akan selalu disegarkan satu kali per menit. Untuk menyegarkan tampilan, Anda pilih ikon segarkan (dua panah lengkung) yang ada di bagian kanan atas layar. Untuk mengubah tingkat penyegaran otomatis untuk item-item yang ada di layar selain alarm, pilihlah panah ke bawah yang ada di samping ikon segarkan dan kemudian pilih tingkat penyegaran. Anda juga dapat memilih untuk menonaktifkan penyegaran otomatis ini.
6. Untuk kembali menampilkan informasi tentang semua sumber daya yang ada di akun Anda, di dekat bagian atas layar tempat nama grup sumber daya saat ini ditampilkan, pilihlah Semua sumber daya.

CloudWatch penagihan dan biaya

Bagian ini menjelaskan bagaimana CloudWatch fitur Amazon menghasilkan biaya. Ini juga menyediakan metode yang dapat membantu Anda menganalisis, mengoptimalkan, dan mengurangi CloudWatch biaya. Di seluruh bagian ini, kami terkadang merujuk pada harga saat mendeskripsikan CloudWatch fitur. Untuk informasi tentang harga, lihat CloudWatch harga Amazon.

Topik

Analisis data CloudWatch biaya dan penggunaan dengan Cost Explorer

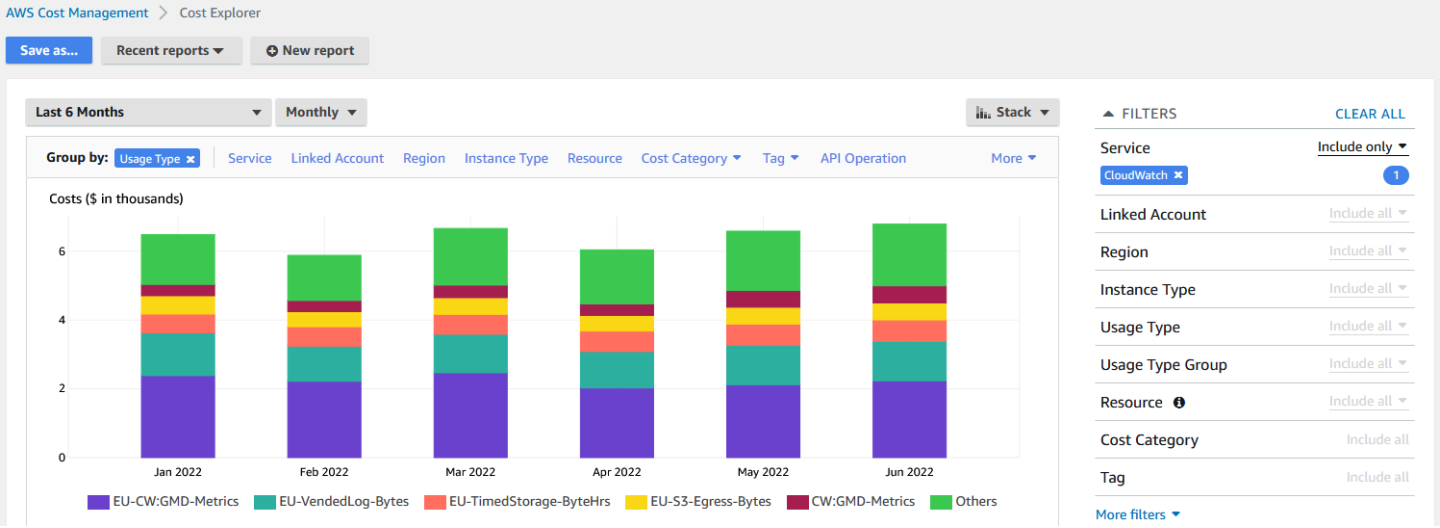
Dengan AWS Cost Explorer, Anda dapat memvisualisasikan dan menganalisis data biaya dan penggunaan Layanan AWS dari waktu ke waktu, termasuk CloudWatch. Untuk informasi selengkapnya, lihat Memulai dengan AWS Cost Explorer.

Prosedur berikut menjelaskan cara menggunakan Cost Explorer untuk memvisualisasikan dan menganalisis data CloudWatch biaya dan penggunaan.

Untuk memvisualisasikan dan menganalisis CloudWatch data biaya dan penggunaan

1. Masuk ke konsol Cost Explorer di https://console.aws.amazon.com/cost-management/home#/custom.
2. Di bawah FILTER, untuk Layanan, pilih CloudWatch.
3. Untuk Grup berdasar, pilih Jenis Penggunaan. Anda juga dapat mengelompokkan hasil yang Anda dapatkan berdasarkan kategori lain, seperti berikut ini:
   - Operasi API – Lihat operasi API manakah yang menimbulkan biaya paling banyak.
   - Wilayah – Lihat Wilayah manakah yang menimbulkan biaya paling banyak.

Gambar berikut menunjukkan contoh biaya yang CloudWatch fitur yang dihasilkan selama enam bulan.



Untuk melihat CloudWatch fitur mana yang menghasilkan biaya paling banyak, lihat nilainya`UsageType`. Misalnya, `EU-CW:GMD-Metrics` mewakili biaya yang dihasilkan oleh permintaan API CloudWatch massal.

> **Note**
>
> String untuk `UsageType` menunjukkan fitur dan Wilayah tertentu. Misalnya, bagian pertama dari `EU-CW:GMD-Metrics` (EU) cocok dengan Wilayah Eropa (Irlandia), dan bagian kedua dari `EU-CW:GMD-Metrics` (`GMD-Metrics`) cocok dengan permintaan API CloudWatch massal.
>
> Seluruh string untuk `UsageType` dapat diformat sebagai berikut: `<Region>-CW:<Feature>` atau`<Region>-<Feature>`.
>
> Untuk meningkatkan keterbacaan, string untuk `UsageType` di tabel di seluruh dokumen ini telah disingkat menjadi awalan pada string. Sebagai contoh, `EU-CW:GMD-Metrics` disingkat menjadi`GMD-Metrics`.

Tabel berikut mencakup nama setiap CloudWatch fitur, daftar nama setiap subfitur, dan daftar string untuk. `UsageType`

CloudWatch fitur	CloudWatch subfitur	**`UsageType`**
CloudWatch metrik	Metrik-metrik kustom	`MetricMonitorUsage`
	Pemantauan terperinci	`MetricMonitorUsage`
	Metrik tersemat	`MetricMonitorUsage`
CloudWatch Permintaan API	Permintaan API	`Requests`
	Dalam jumlah besar (Ambil)	`GMD-Metrics`
	Wawasan Kontributor	`GIRR-Metrics`
	Snapshot citra bitmap	`GMWI-Metrics`
CloudWatch aliran metrik	Aliran metrik	`MetricStreamUsage`
CloudWatch dasbor	Dasbor dengan 50 metrik atau kurang	`DashboardsUsageHour-Basic`
	Dasbor dengan lebih dari 50 metrik	`DashboardsUsageHour`
CloudWatch alarm	Standar (alarm metrik)	`AlarmMonitorUsage`
	Resolusi tinggi (alarm metrik)	`HighResAlarmMonitorUsage`
	Alarm kueri Wawasan Metrik	`MetricInsightAlarmUsage`

CloudWatch fitur	CloudWatch subfitur	**`UsageType`**
	Gabungan (alarm agregat)	`CompositeAlarmMonitorUsage`
CloudWatch Sinyal Aplikasi	Sinyal Aplikasi	`Application-Signals`
CloudWatch log kustom	Kumpulkan (pemasukan)	`DataProcessing-Bytes`
	Simpan (arsip)	`TimedStorage-ByteHrs`
	Analisis (kueri)	`DataScanned-Bytes`
CloudWatch Log Akses Jarang	Kumpulkan (pemasukan)	`DataProcessingIA-Bytes`
CloudWatch log yang dijual	Pengiriman (Amazon CloudWatch Log)	`VendedLog-Bytes`
	Pengiriman (CloudWatch Log Log Akses Jarang)	`VendedLogIA-Bytes`
	Pengiriman (Layanan Penyimpanan Sederhana Amazon)	`S3-Egress-ComprBytes` `S3-Egress-Bytes`
	Pengiriman (Amazon Data Firehose)	`FH-Egress-Bytes`
Wawasan Kontributor	CloudWatch Log (Aturan)	`ContributorInsightRules`

CloudWatch fitur	CloudWatch subfitur	**`UsageType`**
	CloudWatch Log (Acara)	`ContributorInsightEvents`
	Amazon DynamoDB (Aturan)	`ContributorRulesManaged`
	Kejadian DynamoDB)	`ContributorEventsManaged`
Canary (Sintetik)	Jalankan .	`Canary-runs`
Nyata	Kejadian	`Evidently-event`
	Unit Analisis	`Evidently-eau`
RUM	Kejadian	`RUM-event`

Analisis data CloudWatch biaya dan penggunaan dengan AWS Cost and Usage Report s dan Athena

Cara lain untuk menganalisis data CloudWatch biaya dan penggunaan adalah dengan menggunakan AWS Cost and Usage Report s dengan Amazon Athena. AWS Cost and Usage Report s berisi seperangkat data biaya dan penggunaan yang komprehensif. Anda dapat membuat laporan yang melacak biaya dan penggunaan Anda, dan Anda juga dapat menerbitkan laporan ini ke bucket S3 pilihan Anda. Anda juga dapat mengunduh dan menghapus laporan-laporan Anda dari bucket S3 Anda. Untuk informasi lebih lanjut, lihat Apa itu AWS Cost and Usage Report s? di Panduan Pengguna AWS Cost and Usage Report s.

> **Note**
>
> Tidak ada biaya untuk menggunakan AWS Cost and Usage Report s. Anda hanya membayar biaya penyimpanan ketika memublikasikan laporan Anda ke Amazon Simple Storage Service (Amazon S3). Untuk informasi selengkapnya, silakan lihat Kuota dan Batasan di Panduan Pengguna AWS Cost and Usage Report.

Athena adalah layanan kueri yang dapat Anda gunakan dengan AWS Cost and Usage Report s untuk menganalisis data biaya dan penggunaan. Anda dapat menggunakan kueri laporan Anda di bucket S3 Anda tanpa perlu mengunduhnya terlebih dahulu. Untuk informasi selengkapnya, silakan lihat Apa yang dimaksud dengan Amazon Athena? di Panduan Pengguna Amazon Athena. Untuk informasi selengkapnya, silakan lihat Apa yang dimaksud dengan Amazon Athena? di Panduan Pengguna Amazon Athena. Untuk informasi selengkapnya tenngenai harga, silakan lihat harga Amazon Athena.

Prosedur berikut menjelaskan proses untuk mengaktifkan AWS Cost and Usage Report dan mengintegrasikan layanan dengan Athena. Prosedur ini berisi dua contoh kueri yang dapat Anda gunakan untuk menganalisis data CloudWatch biaya dan penggunaan.

> **Note**
>
> Anda dapat menggunakan salah satu contoh kueri di dokumen ini. Semua contoh kueri dalam dokumen ini sesuai dengan basis data bernama laporanbiayadanpenggunaan, dan menunjukkan hasil untuk bulan April dan tahun 2022. Anda dapat mengubah informasi ini. Namun demikian, sebelum Anda menjalankan kueri, pastikan bahwa nama basis data Anda cocok dengan nama basis data dalam kueri.

Untuk menganalisis data biaya dan penggunaan dengan AWS Cost and Usage Report s dan Athena

1. Aktifkan AWS Cost and Usage Report s. Untuk informasi selengkapnya, silakan lihat Membuat laporan biaya dan penggunaan di Panduan Pengguna AWS Cost and Usage Report.

> **Tip**
>
> Ketika membuat laporan Anda, pastikan untuk memilih Sertakan ID sumber daya. Jika tidak menyertakannya, laporan Anda tidak akan menyertakan kolom`line_item_resource_id`. Baris ini membantu Anda mengidentifikasi biaya lebih lanjut ketika menganalisis data biaya dan penggunaan.

2. Integrasikan AWS Cost and Usage Report s dengan Athena. Untuk informasi selengkapnya, lihat Menyiapkan Athena menggunakan AWS CloudFormation templat di Panduan Pengguna AWS Cost and Usage Report s.
3. Gunakan kueri laporan biaya dan penggunaan Anda.

Contoh: Kueri Athena

Anda dapat menggunakan kueri berikut untuk menunjukkan CloudWatch fitur mana yang menghasilkan biaya paling banyak untuk bulan tertentu.

```
SELECT
CASE
-- Metrics
WHEN line_item_usage_type LIKE '%%MetricMonitorUsage%%' THEN 'Metrics (Custom, Detailed
 monitoring management portal EMF)'
WHEN line_item_usage_type LIKE '%%Requests%%' THEN 'Metrics (API Requests)'
WHEN line_item_usage_type LIKE '%%GMD-Metrics%%' THEN 'Metrics (Bulk API Requests)'
WHEN line_item_usage_type LIKE '%%MetricStreamUsage%%' THEN 'Metric Streams'
-- Dashboard
WHEN line_item_usage_type LIKE '%%DashboardsUsageHour%%' THEN 'Dashboards'
-- Alarms
WHEN line_item_usage_type LIKE '%%AlarmMonitorUsage%%' THEN 'Alarms (Standard)'
WHEN line_item_usage_type LIKE '%%HighResAlarmMonitorUsage%%' THEN 'Alarms (High
 Resolution)'
WHEN line_item_usage_type LIKE '%%MetricInsightAlarmUsage%%' THEN 'Alarms (Metrics
 Insights)'
WHEN line_item_usage_type LIKE '%%CompositeAlarmMonitorUsage%%' THEN 'Alarms
 (Composite)'
-- Logs
WHEN line_item_usage_type LIKE '%%DataProcessing-Bytes%%' THEN 'Logs (Collect - Data
 Ingestion)'
-- Logs
```

```
WHEN line_item_usage_type LIKE '%%DataProcessingIA-Bytes%%' THEN 'Infrequent Access
 Logs (Collect - Data Ingestion)'
WHEN line_item_usage_type LIKE '%%TimedStorage-ByteHrs%%' THEN 'Logs (Storage -
 Archival)'
WHEN line_item_usage_type LIKE '%%DataScanned-Bytes%%' THEN 'Logs (Analyze - Logs
 Insights queries)'
-- Vended Logs
WHEN line_item_usage_type LIKE '%%VendedLog-Bytes%%' THEN 'Vended Logs (Delivered to
 CW)'
WHEN line_item_usage_type LIKE '%%VendedLogIA-Bytes%%' THEN 'Vended Infrequent Access
 Logs (Delivered to CW)'
WHEN line_item_usage_type LIKE '%%FH-Egress-Bytes%%' THEN 'Vended Logs (Delivered to
 Kinesis FH)'
WHEN (line_item_usage_type LIKE '%%S3-Egress-Bytes%%') OR (line_item_usage_type LIKE '%
%S3-Egress-
ComprBytes%%') THEN 'Vended Logs (Delivered to S3)'
-- Other
WHEN line_item_usage_type LIKE '%%Application-Signals%%' THEN 'Application Signals'
WHEN line_item_usage_type LIKE '%%Canary-runs%%' THEN 'Synthetics'
WHEN line_item_usage_type LIKE '%%Evidently%%' THEN 'Evidently'
WHEN line_item_usage_type LIKE '%%RUM-event%%' THEN 'RUM'
ELSE 'Others'
END AS UsageType,
-- REGEXP_EXTRACT(line_item_resource_id,'^(?:.+?:){5}(.+)$',1) as ResourceID,
-- SUM(CAST(line_item_usage_amount AS double)) AS UsageQuantity,
SUM(CAST(line_item_unblended_cost AS decimal(16,8))) AS TotalSpend
FROM
costandusagereport
WHERE
product_product_name = 'AmazonCloudWatch'
AND year='2022'
AND month='4'
AND line_item_line_item_type NOT IN
 ('Tax','Credit','Refund','EdpDiscount','Fee','RIFee')
-- AND line_item_usage_account_id = '123456789012' - If you want to filter on a
 specific account, you can
remove this comment at the beginning of the line and specify an AWS account.
GROUP BY
1
ORDER BY
TotalSpend DESC,
UsageType;
```

Contoh: Kueri Athena

Anda dapat menggunakan kueri berikut untuk menunjukkan hasil untuk `UsageType` dan `Operation`. Ini menunjukkan kepada Anda bagaimana CloudWatch fitur menghasilkan biaya. Hasil tersebut juga menunjukkan nilai untuk `UsageQuantity` dan`TotalSpend`, sehingga Anda dapat melihat total biaya penggunaan Anda.

> **Tip**
>
> Untuk informasi selengkapnya tentang`UsageType`, tambahkan baris berikut ke kueri ini:
> `line_item_line_item_description`
> Baris ini membuat kolom yang disebut Keterangan.

```
SELECT
bill_payer_account_id as Payer,
line_item_usage_account_id as LinkedAccount,
line_item_usage_type AS UsageType,
line_item_operation AS Operation,
line_item_resource_id AS ResourceID,
SUM(CAST(line_item_usage_amount AS double)) AS UsageQuantity,
SUM(CAST(line_item_unblended_cost AS decimal(16,8))) AS TotalSpend
FROM
costandusagereport
WHERE
product_product_name = 'AmazonCloudWatch'
AND year='2022'
AND month='4'
AND line_item_line_item_type NOT IN
 ('Tax','Credit','Refund','EdpDiscount','Fee','RIFee')
GROUP BY
bill_payer_account_id,
line_item_usage_account_id,
line_item_usage_type,
line_item_resource_id,
line_item_operation
```

Praktik terbaik untuk mengoptimalkan dan menurunkan biaya

CloudWatch metrik

Banyak Layanan AWS, seperti Amazon Elastic Compute Cloud (Amazon EC2), Amazon S3, dan Amazon Data CloudWatch Firehose, secara otomatis mengirim metrik tanpa biaya. Namun demikian, metrik yang dikelompokkan dalam kategori berikut dapat menimbulkan biaya tambahan:

- Metrik kustom, pemantauan terperinci, dan metrik tersemat
- Permintaan API
- Aliran metrik

Untuk informasi selengkapnya, lihat Menggunakan CloudWatch metrik Amazon.

Metrik kustom, pemantauan terperinci, dan metrik tersemat

Metrik-metrik kustom

Anda dapat membuat metrik kustom untuk mengatur titik data dalam urutan dan pada kecepatan berapa pun.

Semua metrik kustom diprorata per jam. Mereka diukur hanya ketika mereka dikirim ke CloudWatch. Untuk informasi tentang bagaimana metrik diberi harga, lihat Harga Amazon CloudWatch .

Tabel berikut mencantumkan nama subfitur yang relevan untuk CloudWatch metrik. Tabel ini mencakup string untuk `UsageType` dan`Operation`, yang dapat membantu Anda menganalisis dan mengidentifikasi biaya yang terkait dengan metrik.

> **Note**
>
> Untuk mendapatkan hal lebih detail tentang metrik yang tercantum dalam tabel berikut saat Anda menanyakan data biaya dan penggunaan dengan Athena, cocokkan string `Operation` dengan hasil yang ditunjukkan `line_item_operation`.

CloudWatchsubfitur	**`UsageType`**	**`Operation`**	Tujuan

CloudWatchsubfitur	**UsageType**	**Operation**	Tujuan
Metrik-metrik kustom	`MetricMonitorUsage`	`MetricStorage`	Metrik-metrik kustom
Pemantauan terperinci	`MetricMonitorUsage`	`MetricStorage:AWS/{Service}`	Pemantauan terperinci
Metrik tersemat	`MetricMonitorUsage`	`MetricStorage:AWS/Logs-EMF`	Metrik tersemat log
Filter log	`MetricMonitorUsage`	`MetricStorage:AWS/CloudWatchLogs`	Filter metrik grup log

Pemantauan terperinci

CloudWatch memiliki dua jenis pemantauan:

- Pemantauan dasar

  Pemantauan dasar gratis dan diaktifkan secara otomatis diaktifkan untuk semua Layanan AWS yang mendukung fitur tersebut.
- Pemantauan terperinci

  Pemantauan terperinci menimbulkan biaya dan menambahkan peningkatan yang berbeda tergantung pada. Layanan AWS Untuk setiap Layanan AWS yang mendukung pemantauan terperinci, Anda dapat memilih apakah akan akan mengaktifkannya untuk layanan itu. Untuk informasi selengkapnya, silakan lihat Pemantauan dasar dan terperinci.

> **Note**
>
> Layanan AWS Dukungan lain pemantauan terperinci dan mungkin merujuk ke fitur ini menggunakan nama yang berbeda. Sebagai contoh, pemantauan terperinci untuk Amazon S3 disebut sebagai metrik permintaan.

Mirip dengan metrik khusus, pemantauan terperinci diprorata per jam dan diukur hanya ketika data dikirim ke. CloudWatch Pemantauan terperinci menghasilkan biaya berdasarkan jumlah metrik yang dikirim ke CloudWatch. Untuk menurunkan biaya, cukup aktifkan pemantauan terperinci bila diperlukan. Untuk informasi tentang harga pemantauan terperinci, lihat CloudWatch Harga Amazon.

Contoh: Kueri Athena

Anda dapat menggunakan kueri berikut untuk menunjukkan instans EC2 manakah yang memiliki pemantauan terperinci yang diaktifkan.

```
SELECT
bill_payer_account_id as Payer,
line_item_usage_account_id as LinkedAccount,
line_item_usage_type AS UsageType,
line_item_operation AS Operation,
line_item_resource_id AS ResourceID,
SUM(CAST(line_item_usage_amount AS double)) AS UsageQuantity,
SUM(CAST(line_item_unblended_cost AS decimal(16,8))) AS TotalSpend
FROM
costandusagereport
WHERE
product_product_name = 'AmazonCloudWatch'
AND year='2022'
AND month='4'
AND line_item_operation='MetricStorage:AWS/EC2'
AND line_item_line_item_type NOT IN
 ('Tax','Credit','Refund','EdpDiscount','Fee','RIFee')
GROUP BY
bill_payer_account_id,
line_item_usage_account_id,
line_item_usage_type,
line_item_resource_id,
line_item_operation,
line_item_line_item_description
ORDER BY line_item_operation
```

Metrik tersemat

Dengan format metrik yang CloudWatch disematkan, Anda dapat menyerap data aplikasi sebagai data log, sehingga Anda dapat menghasilkan metrik yang dapat ditindaklanjuti. Untuk informasi selengkapnya, lihat Menelan log kardinalitas tinggi dan menghasilkan metrik dengan format metrik yang disematkan. CloudWatch

Metrik tersemat menghasilkan biaya berdasarkan jumlah log yang dimasukkan, jumlah log yang diarsipkan, dan jumlah metrik kustom yang dihasilkan.

Tabel berikut mencantumkan nama subfitur yang relevan untuk format metrik CloudWatch tertanam. Tabel ini mencantumkan string untuk `UsageType` dan`Operation`, yang dapat membantu Anda menganalisis dan mengidentifikasi biaya.

CloudWatch subfitur	**`UsageType`**	**`Operation`**	Tujuan
Metrik-metrik kustom	`MetricMonitorUsage`	`MetricStorage:AWS/Logs-EMF`	Metrik tersemat log
Masukan log	`DataProcessing-Bytes`	`PutLogEvents`	Mengunggah batch kejadian log ke grup log atau log stream tertentu
Arsip log	`TimedStorage-ByteHrs`	`HourlyStorageMetering`	Menyimpan log per jam dan log per byte di CloudWatch Log

Untuk menganalisis biaya, gunakan AWS Cost and Usage Report s dengan Athena sehingga Anda dapat mengidentifikasi metrik mana yang menghasilkan biaya dan menentukan bagaimana biaya dihasilkan.

Untuk memaksimalkan biaya yang dihasilkan oleh format metrik CloudWatch tertanam, hindari membuat metrik berdasarkan dimensi kardinalitas tinggi. Dengan cara ini, CloudWatch tidak membuat metrik khusus untuk setiap kombinasi dimensi unik. Untuk informasi selengkapnya, silakan lihat Dimensi.

Jika Anda menggunakan CloudWatch Container Insights untuk memanfaatkan format metrik yang disematkan, Anda dapat menggunakan AWS Distro untuk Open Telemetry sebagai alternatif untuk memaksimalkan biaya terkait metrik. Dengan Wawasan Kontainer, Anda dapat mengumpulkan, menyatukan, dan meringkas metrik dan log dari aplikasi dan mikrolayanan terkontainer. Saat Anda mengaktifkan Wawasan Kontainer, CloudWatch agen akan mengirimkan log Anda CloudWatch, sehingga dapat menggunakan log untuk menghasilkan metrik yang disematkan. Namun, CloudWatch

agen hanya mengirimkan sejumlah metrik tetap ke CloudWatch, dan Anda dikenakan biaya untuk semua metrik yang tersedia, termasuk yang tidak Anda gunakan. Dengan AWS Distro untuk Open Telemetry, Anda dapat mengonfigurasi dan menyesuaikan metrik dan dimensi mana yang dikirim. CloudWatch Tindakan ini membantu Anda mengurangi volume data dan biaya yang dihasilkan Wawasan Kontainer. Untuk informasi selengkapnya, silakan lihat sumber daya berikut:

- Menggunakan Wawasan Kontainer
- AWS Distro untuk Telemetri Terbuka

Permintaan API

CloudWatch memiliki jenis permintaan API berikut:

- Permintaan API
- Dalam jumlah besar (Ambil)
- Wawasan Kontributor
- Snapshot citra bitmap

Permintaan API menimbulkan biaya berdasarkan jenis permintaan dan jumlah metrik yang diminta.

Tabel berikut ini mencakup daftar jenis perminataan API untuk `UsageType` dan`Operation`, yang dapat membantu Anda menganalisis dan mengidentifikasi biaya yang terkait dengan metrik.

Jenis permintaan API	**`UsageType`**	**`Operation`**	Tujuan
Permintaan API	`Requests`	`GetMetric Statistics`	Mengambil statistik untuk metrik-metrik yang ditentukan
	`Requests`	`ListMetrics`	Daftar metrik-metrik yang ditentukan
	`Requests`	`PutMetricData`	Menerbitkan titik data metrik ke CloudWatch

Jenis permintaan API	**UsageType**	**Operation**	Tujuan
	Requests	GetDashboard	Menampilkan detail pada dasbor yang ditentukan
	Requests	ListDashboards	Buat daftar dasbor di akun Anda
	Requests	PutDashboard	Membuat atau memperbarui sebuah dasbor
	Requests	DeleteDashboards	Menghapus semua dasbor yang ditentukan
Dalam jumlah besar (Ambil)	GMD-Metrics	GetMetricData	Mengambil nilai CloudWatch metrik
Wawasan Kontributor	GIRR-Metrics	GetInsightRuleReport	Mengembalikan data deret waktu yang dikumpulkan oleh aturan Wawasan Kontributor
Snapshot citra bitmap	GMWI-Metrics	GetMetricWidgetImage	Mengambil snapshot dari satu atau beberapa CloudWatch metrik sebagai gambar bitmap

Untuk menganalisis biaya, gunakan Cost Explorer, dan kelompokkan hasil Anda berdasarkan Operasi API.

Biaya untuk permintaan API bervariasi, dan Anda dikenakan biaya jika melebihi jumlah panggilan API yang diberikan kepada Anda di bawah batas Tingkat AWS Gratis.

> **Note**
>
> `GetMetricData`adan `GetMetricWidgetImage` tidak termasuk dalam batas Tingkat AWS Gratis. Untuk informasi selengkapnya, lihat Menggunakan Tingkat AWS Gratis di Panduan AWS Billing Pengguna.

Permintaan API yang biasanya menyebabkan biaya adalah permintaan `Put` dan `Get`.

`PutMetricData`

`PutMetricData`dikenai biaya setiap kali permintaan ini dipanggil dan dapat menimbulkan biaya yang signifikan tergantung pada kasus penggunaan. Untuk informasi selengkapnya, lihat PutMetricDatadi Referensi Amazon CloudWatch API.

Untuk memaksimalkan biaya yang ditimbulkan oleh `PutMetricData`, kumpulkan lebih banyak data ke dalam panggilan API Anda. Bergantung pada kasus penggunaan Anda, pertimbangkan untuk menggunakan CloudWatch Log atau format metrik yang CloudWatch disematkan untuk menyuntikkan data metrik. Untuk informasi selengkapnya, lihat sumber daya berikut:

- Apa itu Amazon CloudWatch Logs? di Panduan Pengguna CloudWatch Log Amazon
- Menelan log kardinalitas tinggi dan menghasilkan metrik dengan format metrik tertanam CloudWatch
- Menurunkan biaya dan fokus pada pelanggan kami dengan metrik kustom CloudWatch tertanam Amazon

`GetMetricData`

`GetMetricData` juga dapat menimbulkanan biaya yang signifikan. Kasus penggunaan umum yang mendorong biaya melibatkan alat pemantauan pihak ketiga yang menarik data untuk menghasilkan insights. Untuk informasi selengkapnya, lihat GetMetricDatadi Referensi Amazon CloudWatch API.

Untuk menurunkan biaya yang ditimbulkan oleh`GetMetricData`, pertimbangkan hanya mengambil data yang dipantau dan digunakan, atau pertimbangkan untuk mengambil data lebih jarang. Tergantung pada kasus penggunaan Anda, Anda dapat mempertimbangkan penggunaan aliran metrik sebagai ganti `GetMetricData`, sehingga Anda dapat mendorong data secara waktu nyata untuk pihak ketiga dengan biaya lebih rendah. Untuk informasi selengkapnya, silakan lihat sumber daya berikut:

- Menggunakan pengaliran metrik
- CloudWatch Aliran Metrik - Kirim AWS Metrik ke Mitra dan Aplikasi Anda secara Real Time

GetMetricStatistics

Tergantung pada kasus penggunaan Anda, Anda dapat mempertimbangkan untuk menggunakan `GetMetricStatistics` alih-alih `GetMetricData`. Dengan `GetMetricData`, Anda dapat mengambil data dengan cepat dan dalam skala besar. Namun, `GetMetricStatistics` termasuk dalam batas Tingkat AWS Gratis hingga satu juta permintaan API, yang dapat membantu Anda mengurangi biaya jika Anda tidak perlu mengambil sebanyak mungkin metrik dan titik data per panggilan. Untuk informasi selengkapnya, lihat sumber daya berikut:

- GetMetricStatisticsdi Referensi CloudWatch API Amazon
- Haruskah saya menggunakan GetMetricData atau GetMetricStatistics?

> **Note**
>
> Pemanggil eksternal melakukan panggilan API. Saat ini, satu-satunya cara untuk mengidentifikasi penelepon ini adalah dengan membuka permintaan dukungan teknis kepada CloudWatch tim dan meminta informasi tentang mereka. Untuk informasi tentang membuat permintaan dukungan teknis, lihat Bagaimana cara mendapatkan dukungan teknis AWS? .

CloudWatch aliran metrik

Dengan aliran CloudWatch metrik, Anda dapat mengirim metrik secara terus menerus ke AWS tujuan dan tujuan penyedia layanan pihak ketiga.

Aliran metrik menimbulkan biaya berdasar jumlah pembaruan metrik. Pembaruan metrik selalu mencantumkan nilai untuk statistik berikut:

- `Minimum`
- `Maximum`
- `Sample Count`
- `Sum`

Untuk informasi selengkapnya, silakan lihat Statistik yang dapat dialirkan.

Untuk menganalisis biaya yang dihasilkan oleh aliran CloudWatch metrik, gunakan AWS Cost and Usage Report s dengan Athena. Dengan cara ini, Anda dapat mengidentifikasi aliran metrik manakah yang menimbulkan biaya dan menentukan cara biaya ditimbukan.

Contoh: Kueri Athena

Anda dapat menggunakan kueri berikut untuk melacak aliran metrik manakah yang menimbulkan biaya berdasarkan Amazon Resource Name (ARN).

```
SELECT
SPLIT_PART(line_item_resource_id,'/',2) AS "Stream Name",
line_item_resource_id as ARN,
SUM(CAST(line_item_unblended_cost AS decimal(16,2))) AS TotalSpend
FROM
costandusagereport
WHERE
product_product_name = 'AmazonCloudWatch'
AND year='2022'
AND month='4'
AND line_item_line_item_type NOT IN
 ('Tax','Credit','Refund','EdpDiscount','Fee','RIFee')
-- AND line_item_usage_account_id = '123456789012' - If you want to filter on a
 specific account, you can
remove this comment at the beginning of the line and specify an AWS account.
AND line_item_usage_type LIKE '%%MetricStreamUsage%%'
GROUP BY line_item_resource_id
ORDER BY TotalSpend DESC
```

Untuk mengurangi biaya yang dihasilkan oleh aliran CloudWatch metrik, streaming hanya metrik yang membawa nilai bisnis Anda. Anda juga dapat menghentikan atau melakukan jeda aliran metrik apa pun yang tidak Anda gunakan.

CloudWatch alarm

Dengan CloudWatch alarm, Anda dapat membuat alarm berdasarkan metrik tunggal, alarm berdasarkan kueri Wawasan Metrik, dan alarm komposit yang menonton alarm lainnya.

> **Note**
>
> Biaya untuk alarm metrik dan gabungan diprorata berdasar jam. Anda mengeluarkan biaya untuk alarm Anda hanya saat alarm Anda diatur. Untuk mengoptimalkan biaya, Anda pastikan untuk tidak meninggalkan alarm yang salah konfigurasi atau bernilai rendah. Untuk membantu ini, Anda dapat mengotomatiskan pembersihan CloudWatch alarm yang tidak lagi Anda perlukan. Untuk informasi selengkapnya, lihat Mengotomatiskan Pembersihan CloudWatch Alarm Amazon pada Skala

Alarm metrik

Alarm metrik memiliki pengaturan resolusi sebagai berikut:

- Standar (dievaluasi setiap 60 detik)
- Resolusi tinggi (dievaluasi setiap 10 detik)

Saat Anda menentukan metrik, biaya Anda didasarkan pada pengaturan resolusi alarm Anda dan jumlah metrik yang merujuk alarm Anda. Sebagai contoh, sebuah alarm metrik yang mereferensikan satu metrik menimbulkan satu biaya alarm-metrik per jam. Untuk informasi selengkapnya, lihat Menggunakan CloudWatch alarm Amazon.

Jika Anda membuat alarm metrik yang berisi ekspresi matematika metrik, yang memberi rujukan beberapa metrik, Anda akan dikenai biaya untuk setiap metrik alarm yang dirujukkan dalam ekspresi matematika metrik. Untuk informasi tentang cara membuat alarm metrik yang berisi ekspresi matematika metrik, lihat Membuat CloudWatch alarm berdasarkan ekspresi matematika metrik.

Jika Anda membuat alarm deteksi anomali, di mana alarm Anda menganalisis data metrik masa lalu untuk membuat perkiraan model nilai, Anda dikenai biaya untuk setiap metrik alarm yang dirujukkan dalam alarm Anda ditambah dua metrik tambahan, satu untuk metrik pita atas dan bawah yang dibuat oleh model deteksi anomali. Untuk informasi tentang cara membuat alarm deteksi anomali, lihat Membuat CloudWatch alarm berdasarkan deteksi anomali.

Alarm-alarm kueri Wawasan Metrik

Alarm kueri Wawasan Metrik adalah jenis alarm metrik tertentu, hanya tersedia dengan resolusi standar (dievaluasi setiap 60 detik).

Ketika Anda membuat alarm kueri Wawasan Metrik, biaya Anda didasarkan pada jumlah metrik yang dianalisis oleh kueri yang dirujuk alarm Anda. Sebagai contoh, alarm kueri Wawasan Metrik yang merujuk kueri yang filternya cocok dengan sepuluh metrik menimbulkan sepuluh metrik yang dianalisis biaya per jam. Untuk informasi selengkapnya, lihat contoh harga di Amazon CloudWatch Pricing.

Jika Anda membuat alarm yang berisi baik kueri Wawasan Metrik dan ekspresi matematika metrik, alarm tersebut dilaporkan sebagai alarm kueri Wawasan Metrik. Jika alarm Anda berisi ekspresi matematika metrik yang memberik rujukan metrik lain selain metrik yang dianalisis oleh kueri Wawasan Metrik, Anda akan dikenai biaya tambahan untuk setiap metrik alarm yang dirujukkan dalam ekspresi matematika metrik. Untuk informasi tentang cara membuat alarm metrik yang berisi ekspresi matematika metrik, lihat Membuat CloudWatch alarm berdasarkan ekspresi matematika metrik.

Alarm gabungan

Alarm gabungan berisi ekspresi aturan yang menentukan bagaimana ekspresi tersebut harus mengevaluasi status alarm lain untuk menentukan statusnya sendiri. Alarm gabungan dikenai biaya standar per jam, terlepas dari berapa banyak alarm lain yang mereka evaluasi. Alarm yang menggabungkan referensi alarm dalam ekspresi aturan dikenai biaya terpisah. Untuk informasi selengkapnya, silakan lihat Membuat sebuah alarm gabungan.

Jenis penggunaan alarm

Tabel berikut mencantumkan nama subfitur yang relevan untuk CloudWatch alarm. Tabel ini mencakup string untuk `UsageType`, yang dapat membantu Anda menganalisis dan mengidentifikasi biaya yang terkait dengan metrik.

CloudWatchsubfitur	**`UsageType`**
Alarm metrik standar	`AlarmMonitorUsage`
Alarm metrik resolusi tinggi	`HighResAlarmMonitorUsage`
Alarm kueri Wawasan Metrik	`MetricInsightAlarmUsage`
Alarm gabungan	`CompositeAlarmMonitorUsage`

Mengurangi biaya alarm

Untuk mengoptimalkan biaya yang dihasilkan oleh alarm matematika metrik yang menggabungkan empat metrik atau lebih, Anda dapat mengumpulkan data sebelum data dikirim. CloudWatch Dengan cara ini, Anda dapat membuat alarm untuk metrik tunggal, alih-alih alarm yang mengumpulkan data untuk beberapa metrik. Untuk informasi selengkapnya, silakan lihat Menublikasikan metrik kustom.

Untuk mengoptimalkan biaya yang ditimbulkan oleh alarm kueri Wawasan Metrik, Anda dapat memastikan bahwa filter yang digunakan untuk kueri hanya cocok dengan metrik yang ingin Anda pantau.

Cara terbaik untuk menurunkan biaya yaitu dengan menghapus semua alarm yang tidak perlu atau tidak digunakan. Misalnya, Anda dapat menghapus alarm yang mengevaluasi metrik yang dipancarkan oleh AWS sumber daya yang tidak ada lagi.

Contoh: Periksa alarm dalam status ***INSUFFICIENT_DATA*** dengan ***DescribeAlarms***

Jika Anda menghapus sumber daya, tetapi tidak alarm metrik yang dipancarkan sumber daya, alarm tersebut masih ada dan biasanya akan masuk ke status `INSUFFICIENT_DATA`. Untuk memeriksa alarm yang berada dalam `INSUFFICIENT_DATA` status, gunakan perintah berikut AWS Command Line Interface (AWS CLI).

```
$ aws cloudwatch describe-alarms –state-value INSUFFICIENT_DATA
```

Cara lain untuk menurunkan biaya mencakup berikut ini:

- Pastikan Anda membuat alarm-alarm untuk metrik yang benar.
- Pastikan Anda tidak mengaktifkan alarm apa pun di Wilayah tempat di mana Anda tidak bekerja.
- Ingatlah bahwa, meskipun alarm gabungan mengurangi kebisingan, alarm tersebut menimbulkan biaya tambahan.
- Ketika memutuskan apakah akan membuat alarm standar atau alarm resolusi tinggi, pertimbangkan kasus penggunaan Anda dan nilai yang dibawa setiap jenis alarm.

CloudWatch Log

Amazon CloudWatch Logs memiliki jenis log berikut:

- Log kustom (log yang Anda buat untuk aplikasi Anda)
- Log yang dijual (log yang lain Layanan AWS, seperti Amazon Virtual Private Cloud (Amazon VPC) dan Amazon Route 53, buat atas nama Anda)

Untuk informasi selengkapnya tentang log vended, lihat Mengaktifkan logging dari AWS layanan tertentu di Panduan Pengguna Amazon CloudWatch Logs.

Log kustom dan dipasok menimbulkan biaya berdasarkan jumlah log yang dikumpulkan, disimpan, dan dianalisis. Secara terpisah, log penjual menghasilkan biaya pengiriman ke Amazon S3 dan Firehose.

Tabel berikut mencantumkan nama fitur CloudWatch Log dan nama subfitur yang relevan. Tabel ini mencakup string untuk `UsageType` dan`Operation`, yang dapat membantu Anda menganalisis dan mengidentifikasi biaya yang terkait dengan log.

CloudWatch Fitur log	CloudWatch Subfitur log	**`UsageType`**	**`Operation`**	Tujuan
Log kustom	Kumpulkan (pemasukan)	`DataProcessing-Bytes`	`PutLogEvents`	Mengunggah sekumpulan log ke aliran log stream tertentu
	Simpan (arsip)	`TimedStorage-ByteHrs`	`HourlyStorageMetering`	Menyimpan log per jam dan log per byte di CloudWatch Log
	Analisis (Kueri Wawasan Log)	`DataScanned-Bytes`	`StartQuery`	Data log dipindai oleh kueri CloudWatch Logs Insights
Log yang dipasok	Pengiriman (CloudWatch Log)	`VendedLog-Bytes`	`PutLogEvents`	Mengunggah sekumpulan log ke aliran log stream tertentu
	Pengiriman (Amazon S3)	`S3-Egress-ComprBytes`	`LogDelivery`	Mengirim log vended (CloudWatch, Amazon S3, atau Firehose)

CloudWatch Fitur log	CloudWatch Subfitur log	**UsageType**	**Operation**	Tujuan
		S3-Egress-Bytes		
	Pengiriman (Firehose)	FH-Egress-Bytes	LogDelivery	Mengirim log vended (CloudWatch, Amazon S3, atau Firehose)

Untuk menganalisis biaya, gunakan AWS Cost and Usage Report s dengan Athena, sehingga Anda dapat mengidentifikasi log mana yang menghasilkan biaya dan menentukan bagaimana biaya dihasilkan.

Contoh: Kueri Athena

Anda dapat menggunakan kueri berikut untuk melacak log mana yang menimbulkan biaya berdasarkan ID sumber daya.

```
SELECT
bill_payer_account_id as Payer,
line_item_usage_account_id as LinkedAccount,
line_item_resource_id AS ResourceID,
line_item_usage_type AS UsageType,
SUM(CAST(line_item_unblended_cost AS decimal(16,8))) AS TotalSpend,
SUM(CAST(line_item_usage_amount AS double)) AS UsageQuantity
FROM
costandusagereport
WHERE
product_product_name = 'AmazonCloudWatch'
AND year='2022'
AND month='4'
AND line_item_operation IN
 ('PutLogEvents','HourlyStorageMetering','StartQuery','LogDelivery')
AND line_item_line_item_type NOT IN
 ('Tax','Credit','Refund','EdpDiscount','Fee','RIFee')
GROUP BY
bill_payer_account_id,
```

```
line_item_usage_account_id,
line_item_usage_type,
line_item_resource_id,
line_item_operation
ORDER BY
TotalSpend DESC
```

Untuk memaksimalkan biaya yang dihasilkan oleh CloudWatch Log, pertimbangkan hal berikut:

- Hanya log kejadian peristiwa yang membawa nilai bisnis Anda. Log ini membantu Anda mengeluarkan lebih sedikit biaya untuk masuk.
- Ubah pengaturan penyimpanan log Anda, sehingga Anda akan menghasilkan lebih sedikit biaya untuk penyimpanan. Untuk informasi selengkapnya, lihat Mengubah penyimpanan data CloudWatch log di Log di Panduan Pengguna CloudWatch Log Amazon.
- Jalankan kueri yang disimpan secara otomatis oleh Wawasan CloudWatch Log dalam riwayat Anda. Dengan cara ini, Anda mengeluarkan lebih sedikit biaya untuk analisis. Untuk informasi selengkapnya, lihat Melihat kueri atau riwayat kueri yang sedang berjalan di Panduan Pengguna CloudWatch Log Amazon.
- Gunakan CloudWatch agen untuk mengumpulkan log sistem dan aplikasi dan mengirimkannya ke CloudWatch. Dengan cara ini, Anda hanya dapat mengumpulkan kejadian log yang memenuhi kriteria Anda. Untuk informasi selengkapnya, lihat CloudWatch Agen Amazon menambahkan Dukungan untuk Ekspresi Filter Log.

Untuk mengurangi biaya log vended, pertimbangkan kasus penggunaan Anda, lalu tentukan apakah log Anda harus dikirim ke CloudWatch atau Amazon S3. Untuk informasi selengkapnya, lihat Log yang dikirim ke Amazon S3 di Panduan Pengguna Amazon CloudWatch Logs.

> **Tip**
>
> Jika Anda ingin menggunakan filter metrik, filter langganan, Wawasan CloudWatch Log, dan Wawasan Kontributor, kirim log vended ke. CloudWatch
> Alternatifnya, jika Anda bekerja dengan Log Aliran VPC dan menggunakannya untuk tujuan audit dan kepatuhan, kirim log yang dipasok ke Amazon S3.
> Untuk informasi tentang cara melacak tagihan yang dihasilkan dengan menerbitkan Log Aliran VPC ke bucket S3, lihat Menggunakan AWS Cost and Usage Report s dan tag alokasi biaya untuk memahami konsumsi data VPC Flow Logs di Amazon S3.

Untuk informasi tambahan tentang cara memaksimalkan biaya yang dihasilkan oleh CloudWatch Log, lihat Grup log mana yang menyebabkan kenaikan mendadak pada tagihan CloudWatch Log saya? .

Menggunakan CloudWatch dasbor Amazon

CloudWatch Dasbor Amazon adalah halaman beranda yang dapat disesuaikan di CloudWatch konsol yang dapat Anda gunakan untuk memantau sumber daya Anda dalam satu tampilan, bahkan sumber daya yang tersebar di berbagai Wilayah. Anda dapat menggunakan CloudWatch dasbor untuk membuat tampilan metrik dan alarm yang disesuaikan untuk sumber daya Anda. AWS

Dengan dasbor tersebut, Anda dapat membuat hal-hal berikut ini:

- Tampilan tunggal untuk metrik dan alarm terpilih untuk membantu Anda menilai kondisi kesehatan sumber daya dan aplikasi Anda di satu Wilayah atau lebih. Anda dapat memilih warna yang digunakan untuk setiap metrik pada masing-masing grafik, sehingga Anda dapat dengan mudah melacak metrik yang sama di beberapa grafik.
- Buku pedoman operasi yang memberikan panduan tentang cara merespons kejadian-kejadian tertentu bagi para anggota tim selama peristiwa operasi.
- Tampilan umum dari pengukuran sumber daya dan aplikasi penting yang dapat dibagikan oleh para anggota tim untuk alur komunikasi yang lebih cepat selama peristiwa operasi.

Jika Anda memiliki beberapa AWS akun, Anda dapat mengatur pengamatan CloudWatch lintas akun dan kemudian membuat dasbor lintas akun kaya di akun pemantauan Anda. Dasbor ini dapat menyertakan grafik metrik dari akun sumber dan widget Wawasan CloudWatch Log dengan kueri grup log dari akun sumber. Selain itu, alarm yang Anda buat di akun pemantauan dapat mengawasi metrik-metrik yang ada di akun sumber. Untuk informasi selengkapnya, lihat CloudWatch observabilitas lintas akun.

Anda dapat membuat dasbor dari konsol atau menggunakan operasi AWS CLI atau `PutDashboard` API. Anda dapat menambahkan dasbor ke daftar favorit, di mana Anda dapat mengakses tidak hanya dasbor-dasbor favorit Anda, tetapi juga dasbor yang baru saja Anda kunjungi. Untuk informasi selengkapnya, silakan lihat Menambahkan dasbor ke daftar favorit Anda.

Untuk mengakses CloudWatch dasbor, Anda memerlukan salah satu dari yang berikut:

- Kebijakan `AdministratorAccess`
- Kebijakan `CloudWatchFullAccess`
- Kebijakan kustom yang mencakup satu atau beberapa izin spesifik tersebut:
  - `cloudwatch:GetDashboard` dan `cloudwatch:ListDashboards` untuk dapat menampilkan dasbor

- `cloudwatch:PutDashboard` untuk dapat membuat atau melakukan modifikasi pada dasbor
- `cloudwatch:DeleteDashboards` untuk dapat menghapus dasbor

Daftar Isi

Membuat CloudWatch dasbor

Untuk memulai, buat CloudWatch dasbor. Anda dapat membuat beberapa dasbor, dan Anda dapat menambahkan dasbor-dasbor itu ke daftar favorit. Anda tidak dibatasi dengan jumlah dasbor yang dapat Anda miliki di Akun AWSAnda. Semua dasbor bersifat global. Dasbor-dasbor itu tidak bersifat spesifik Wilayah.

Prosedur berikut menunjukkan cara membuat dasbor dari CloudWatch konsol. Anda dapat menggunakan operasi API `PutDashboard` untuk membuat sebuah dasbor dari antarmuka baris perintah. Operasi API tersebut berisi string JSON yang menentukan konten dasbor Anda. Untuk informasi selengkapnya tentang membuat dasbor dengan operasi `PutDashboard` API, lihat PutDashboarddi Referensi Amazon CloudWatch API.

> **Tip**
>
> Jika Anda sedang membuat sebuah dasbor baru dengan operasi API `PutDashboard`, Anda dapat menggunakan string JSON dari dasbor yang sudah ada.

Cara membuat sebuah dasbor dari konsol

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, kemudian Buat dasbor.
3. Di kotak dialog Buat dasbor baru, masukkan nama untuk dasbor Anda, dan kemudian pilih Buat dasbor.

   Jika Anda menggunakan nama CloudWatch-Default atau CloudWatch -Default - ***ResourceGroupName***, dasbor akan muncul di ikhtisar CloudWatch halaman beranda di Dashboard Default. Untuk informasi selengkapnya, lihat Memulai dengan Amazon CloudWatch.
4. Di kotak dialog Tambahkan ke dasbor ini, selesaikan salah satu langkah berikut:
   - Untuk menambahkan sebuah grafik ke dasbor, pilih Line atau Area ditumpuk, lalu pilih Konfigurasikan. Dalam kotak dialog Tambahkan grafik metrik, pilih metrik(-metrik) untuk grafik, kemudian pilih Buat widget. Jika ada metrik yang tidak muncul di kotak dialog karena metrik tersebut tidak menerbitkan data lebih dari 14 hari, maka Anda dapat menambahkan metrik tersebut secara manual. Untuk informasi selengkapnya, lihat Grafik metrik secara manual di dasbor CloudWatch .
   - Untuk menambahkan nomor yang menampilkan sebuah metrik ke dasbor, pilih Nomor, lalu pilih Konfigurasikan. Dalam kotak dialog Tambahkan grafik metrik, pilih metrik(-metrik) untuk grafik, kemudian pilih Buat widget.
   - Untuk menambahkan sebuah blok teks ke dasbor, pilih Teks, lalu pilih Konfigurasikan. Di kotak dialog Widget teks baru, untuk Markdown, format teks Anda dengan menggunakan Markdown, lalu pilih Buat widget.
5. (Opsional) Pilih Tambahkan widget, kemudian ulangi langkah 4 untuk menambahkan widget lain ke dasbor. Anda dapat mengulangi langkah ini beberapa kali.

   Untuk masing-masing grafik yang ada di dasbor, ada ikon informasi di kanan atas. Pilih ikon ini untuk melihat deskripsi metrik yang ada dalam grafik tersebut.
6. Pilih Simpan dasbor.

CloudWatch dasbor observabilitas lintas akun

Jika Anda memiliki beberapa AWS akun, Anda dapat mengatur pengamatan CloudWatch lintas akun dan kemudian membuat dasbor lintas akun kaya di akun pemantauan Anda. Anda dapat dengan

mulus mencari, memvisualisasikan, dan menganalisis metrik, log, dan jejak Anda tanpa terhalang batas-batas akun.

Untuk informasi selengkapnya tentang menyiapkan observabilitas CloudWatch lintas akun, lihat. CloudWatch observabilitas lintas akun

Dengan pengamatan CloudWatch lintas akun, Anda dapat melakukan hal berikut di dasbor di akun pemantauan:

- Mencari, menampilkan, dan membuat grafik metrik yang berada di akun sumber. Sebuah grafik tunggal dapat menyertakan metrik-metrik dari beberapa akun.
- Membuat alarm di akun pemantauan yang akan mengawasi metrik-metrik di akun sumber.
- Lihat peristiwa log dari grup log yang terletak di akun sumber, dan jalankan kueri CloudWatch Logs Insights dari grup log di akun sumber. Satu kueri CloudWatch Log Insights dalam akun pemantauan dapat melakukan kueri beberapa grup log di beberapa akun sumber sekaligus.
- Menampilkan simpul dari akun sumber dalam sebuah peta jejak di X-Ray. Anda kemudian dapat melakukan penyaringan peta pada akun sumber tertentu.

Saat Anda masuk ke akun pemantauan, lencana akun Pemantauan berwarna biru muncul di kanan atas setiap halaman yang mendukung fungsionalitas pengamatan CloudWatch lintas akun.

Dasbor lintas akun lintas Wilayah

Anda dapat membuat dasbor lintas wilayah lintas akun, yang merangkum CloudWatch data Anda dari beberapa AWS akun dan beberapa Wilayah menjadi satu dasbor. Dari dasbor tingkat tinggi ini Anda dapat melihat tampilan dari seluruh aplikasi, dan juga menelusuri lebih dalam lagi dasbor-dasbor yang lebih khusus tanpa harus masuk dan keluar akun atau berganti Wilayah.

Anda dapat membuat dasbor lintas wilayah lintas akun di dan secara AWS Management Console terprogram.

Prasyarat

Sebelum Anda dapat membuat sebuah dasbor lintas akun lintas Wilayah, Anda harus mengaktifkan paling sedikit satu akun berbagi dan paling sedikit satu akun pemantauan. Selain itu, untuk dapat menggunakan CloudWatch konsol guna membuat dasbor lintas silang, Anda harus mengaktifkan konsol untuk fungsi lintas akun. Untuk informasi selengkapnya, lihat Konsol CloudWatch lintas akun lintas Wilayah.

Membuat dan menggunakan dasbor lintas akun lintas Wilayah dengan AWS Management Console

Anda dapat menggunakan AWS Management Console untuk membuat dasbor lintas wilayah lintas akun.

Cara membuat dasbor lintas akun lintas Wilayah

1. Masuk ke akun pemantauan.
2. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
3. Pada panel navigasi, silakan pilih Dasbor.
4. Pilih sebuah dasbor, atau buat sebuah dasbor baru.
5. Di bagian atas layar, Anda dapat beralih antara akun dan antara Wilayah. Ketika Anda membuat dasbor, Anda dapat menyertakan widget dari beberapa akun dan Wilayah. Widget termasuk grafik, alarm, dan widget Wawasan CloudWatch Log.

Membuat sebuah grafik dengan metrik dari akun dan Wilayah yang berbeda

1. Masuk ke akun pemantauan.
2. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
3. Pada panel navigasi, silakan pilih Metrik, dan kemudian pilih Semua metrik.
4. Pilih akun dan Wilayah yang Anda ingin tambahkan metriknya. Anda dapat memilih akun dan Wilayah Anda dari menu geser-turun akun dan Wilayah yang ada di sekitaran kanan atas layar.
5. Tambahkan metrik-metrik yang Anda inginkan ke grafik. Untuk informasi selengkapnya, lihat Membuat grafik metrik.
6. Ulangi langkah 4-5 untuk menambahkan metrik-metrik dari akun dan Wilayah lain.
7. (Opsional) Pilih tab Metrik bergrafik dan tambahkan satu fungsi matematika metrik yang menggunakan metrik-metrik yang telah Anda pilih. Untuk informasi selengkapnya, lihat Gunakan matematika metrik.

   Anda juga dapat menyiapkan sebuah grafik tunggal untuk menyertakan beberapa fungsi `SEARCH`. Setiap pencarian dapat mengacu pada akun atau Wilayah yang berbeda.
8. Setelah selesai membuat grafik, pilih Tindakan, Tambahkan ke dasbor.

   Pilih dasbor lintas akun Anda, kemudian pilih Tambahkan ke dasbor.

Menambahkan sebuah alarm dari akun yang berbeda ke dasbor lintas akun Anda

1. Masuk ke akun pemantauan.
2. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
3. Di bagian atas halaman, pilih akun yang menjadi lokasi alarm.
4. Pada panel navigasi, silakan pilih Alarm.
5. Centang kotak yang ada di samping alarm yang ingin ditambahkan, kemudian pilih Tambahkan ke dasbor.
6. Pilih dasbor lintas akun yang ingin Anda tambahkan, kemudian pilih Tambahkan ke dasbor.

Membuat sebuah dasbor lintas akun lintas Wilayah secara terprogram

Anda dapat menggunakan AWS API dan SDK untuk membuat dasbor secara terprogram. Untuk informasi lebih lanjut, lihat PutDashboard.

Untuk mengaktifkan dasbor lintas akun lintas Wilayah, kami telah menambahkan parameter-parameter baru ke struktur tubuh dasbor, seperti yang ditunjukkan pada tabel dan contoh berikut. Untuk informasi selengkapnya tentang struktur tubuh dasbor secara keseluruhan, silakan lihat Struktur Tubuh Dasbor dan Sintaks.

Parameter	Gunakan	Cakupan	Bawaan
`accountId`	Menentukan ID akun yang menjadi lokasi widget atau metrik.	Widget atau metrik	Akun yang saat ini masuk
`region`	Menentukan Wilayah metrik.	Widget atau metrik	Wilayah saat ini yang dipilih di konsol

Contoh-contoh berikut mengilustrasikan sumber JSON untuk widget yang ada di sebuah dasbor lintas akun lintas Wilayah.

Contoh ini akan menetapkan bidang `accountId` ke ID akun berbagi di tingkat widget. Hal ini menentukan bahwa semua metrik yang ada di widget ini akan berasal dari akun dan Wilayah berbagi tersebut.

```
{
    "widgets": [
        {
            ...
            "properties": {
              "metrics": [
                     …..
                     ],
                    "accountId": "111122223333",
                    "region": "us-east-1"
           }
       }
  ]
}
```

Contoh ini menetapkan bidang `accountId` yang berbeda di tingkat masing-masing metrik. Dalam contoh ini, metrik-metrik yang berbeda dalam persamaan matematika metrik berasal dari akun berbagi dan Wilayah yang berbeda.

```
{
"widgets": [
    {
        ...
        "properties": {
            "metrics": [
                [ { "expression": "SUM(METRICS())", "label": "[avg: ${AVG}]
 Expression1", "id": "e1", "stat": "Sum" } ],
                [ "AWS/EC2", "CPUUtilization", { "id": "m2", "accountId":
 "5555666677778888", "region": "us-east-1", "label": "[avg: ${AVG}] ApplicationALabel
 " } ],
                [ ".", ".", { "id": "m1", "accountId": "9999000011112222", "region":
 "eu-west-1", "label": "[avg: ${AVG}] ApplicationBLabel" } ]
            ],
            "view": "timeSeries",
            "region":"us-east-1", ---> home region of the metric. Not present in above
 example
            "stacked": false,
            "stat": "Sum",
            "period": 300,
            "title": "Cross account example"
        }
    }
```

```
]
}
```

Contoh ini menampilkan satu widget alarm.

```
{
    "type": "metric",
    "x": 6,
    "y": 0,
    "width": 6,
    "height": 6,
    "properties": {
        "accountID": "111122223333",
        "title": "over50",
        "annotations": {
            "alarms": [
                "arn:aws:cloudwatch:us-east-1:379642911888:alarm:over50"
            ]
        },
        "view": "timeSeries",
        "stacked": false
    }
}
```

Contoh ini untuk widget Wawasan CloudWatch Log.

```
{
    "type": "log",
    "x": 0,
    "y": 6,
    "width": 24,
    "height": 6,
    "properties": {
        "query": "SOURCE 'route53test' | fields @timestamp, @message\n| sort @timestamp
 desc\n| limit 20",
        "accountId": "111122223333",
        "region": "us-east-1",
        "stacked": false,
        "view": "table"
    }
}
```

Cara lain untuk membuat dasbor secara terprogram adalah dengan terlebih dahulu membuatnya di AWS Management Console, dan kemudian menyalin sumber JSON dari dasbor ini. Untuk melakukan hal tersebut, muat dasbor dan pilih Tindakan, Tampilkan/sunting sumber. Kemudian, Anda dapat menyalin JSON dasbor ini untuk digunakan sebagai sebuah template untuk membuat dasbor-dasbor yang serupa.

Membuat dasbor fleksibel dengan variabel dasbor

Gunakan variabel dasbor untuk membuat dasbor fleksibel yang dapat dengan cepat menampilkan konten yang berbeda di beberapa widget, sesuai dengan nilai dari sebuah bidang input yang ada dalam dasbor. Misalnya, Anda dapat membuat dasbor yang dapat dengan cepat beralih di antara fungsi Lambda yang berbeda atau ID instans Amazon EC2, atau yang dapat beralih ke Wilayah yang berbeda. AWS

Setelah membuat sebuah dasbor yang menggunakan variabel, Anda dapat menyalin pola variabel yang sama ke dasbor lain yang ada.

Menggunakan variabel dasbor akan meningkatkan alur kerja operasional bagi orang-orang yang menggunakan dasbor Anda. Hal ini juga dapat mengurangi biaya Anda karena Anda menggunakan variabel dasbor dalam satu dasbor, bukan membuat beberapa dasbor yang serupa.

> **Note**
>
> Jika Anda berbagi sebuah dasbor yang berisi variabel-variabel dasbor, orang yang berbagi dasbor tersebut dengan tidak akan dapat mengubah-ubah nilai-nilai variabel.

Jenis-jenis variabel dasbor

Variabel dasbor dapat berupa variabel sifat atau variabel pola.

- Variabel sifat mengubah instans dari sebuah sifat yang ada di semua widget dalam dasbor. Sifat ini dapat berupa sifat JSON apa pun yang ada di sumber JSON dari sebuah dasbor, seperti `region`. Atau bisa juga berupa nama dimensi untuk sebuah metrik, seperti `InstanceID` atau `FunctionName`.

  Untuk tutorial yang menggunakan variabel sifat, silakan lihat Tutorial: Cara membuat dasbor Lambda dengan nama fungsi sebagai variabel.

Untuk informasi selengkapnya tentang sumber JSON dasbor, silakan lihat Struktur dan Sintaks Tubuh Dasbor. Di CloudWatch konsol, Anda dapat melihat sumber JSON untuk dasbor kustom apa pun dengan memilih Actions, View/edit source.

- Variabel pola menggunakan suatu pola ekspresi reguler untuk mengubah semua sifat JSON atau hanya bagian tertentu saja.

  Untuk tutorial yang menggunakan suatu variabel pola, silakan lihat Tutorial: Cara membuat sebuah dasbor yang menggunakan pola ekspresi reguler untuk beralih antar Wilayah.

Variabel sifat berlaku untuk sebagian besar kasus penggunaan dan tidak begitu sulit untuk disiapkan.

Topik

- Tutorial: Cara membuat dasbor Lambda dengan nama fungsi sebagai variabel
- Tutorial: Cara membuat sebuah dasbor yang menggunakan pola ekspresi reguler untuk beralih antar Wilayah
- Salin variabel ke dasbor lain

Tutorial: Cara membuat dasbor Lambda dengan nama fungsi sebagai variabel

Langkah-langkah yang dijelaskan dalam prosedur ini menggambarkan cara membuat sebuah dasbor fleksibel yang menunjukkan berbagai grafik metrik, dengan menggunakan variabel sifat. Ini termasuk kotak pilihan geser-turun yang ada di dasbor yang dapat Anda gunakan untuk mengganti metrik di semua grafik di antara fungsi Lambda yang berbeda.

Contoh kasus penggunaan lainnya untuk dasbor jenis ini termasuk menggunakan `InstanceId` sebagai variabel untuk membuat sebuah dasbor metrik dengan menu geser-turun untuk ID instans. Atau, Anda dapat membuat sebuah dasbor yang menggunakan `region` sebagai variabel untuk menampilkan kumpulan metrik yang sama dari Wilayah yang berbeda.

Cara menggunakan sebuah variabel sifat dasbor untuk membuat dasbor Lambda yang fleksibel

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, Buat dasbor.
3. Masukkan nama untuk dasbor tersebut, dan pilih Buat dasbor.

4. Tambahkan widget ke dasbor yang menampilkan metrik-metrik untuk fungsi Lambda. Saat Anda membuat widget ini, tentukan Lambda, Berdasarkan Nama Fungsi untuk metrik widget. Untuk fungsinya, tentukan salah satu fungsi Lambda yang ingin Anda sertakan di dasbor ini.

   Untuk informasi selengkapnya tentang menambahkan widget ke sebuah dasbor, silakan lihat Buat dan bekerja dengan widget di dasbor CloudWatch .
5. Setelah Anda menambahkan widget tersebut, saat Anda melihat dasbor, pilih Tindakan, Variabel, Buat variabel.
6. Pilih Variabel sifat.
7. Untuk Properti yang variabel berubah, pilih FunctionName.
8. Untuk Tipe Input, untuk kasus penggunaan ini, sebaiknya pilih Pilih menu (geser-turun). Hal ini akan membuat sebuah menu geser-turun di dasbor di mana Anda dapat memilih nama fungsi Lambda untuk menampilkan metrik.

   Jika ini untuk sebuah dasbor yang beralih antara hanya dua atau tiga nilai berbeda untuk sebuah variabel, maka Tombol radio akan menjadi pilihan yang baik.

   Jika Anda lebih suka memasukkan atau menempelkan nilai untuk variabel tersebut, Anda sebaiknya memilih Input teks. Opsi ini tidak mencakup daftar geser-turun atau tombol radio.
9. Ketika Anda memilih Pilih menu (geser-turun), Anda kemudian harus memilih apakah akan mengisi menu dengan memasukkan nilai, atau menggunakan pencarian metrik. Untuk kasus penggunaan ini, kita asumsikan Anda memiliki fungsi Lambda dalam jumlah besar dan Anda tidak ingin memasukkan semuanya secara manual. Pilih Gunakan hasil pencarian metrik dan kemudian lakukan langkah-langkah berikut:

   a. Pilih Kueri pra-bangun, Lambda, Kesalahan.

      (Memilih Kesalahan tidak menambahkan metrik Kesalahan ke dasbor. Namun, dengan cepat mengisi kotak pemilihan FunctionNamevariabel.)

   b. Pilih Berdasarkan Nama Fungsi dan kemudian pilih Pencarian.

      Di bawah tombol Cari, Anda kemudian akan melihat FunctionNameyang dipilih. Anda juga melihat pesan tentang berapa banyak nilai FunctionNamedimensi yang ditemukan untuk mengisi kotak input.
10. (Opsional) Untuk pengaturan lainnya, pilih Pengaturan sekunder dan lakukan satu atau beberapa langkah berikut:

- Untuk melakukan kustomisasi pada nama variabel Anda, masukkan nama di Nama variabel kustom.
- Untuk melakukan kustomisasi pada label untuk bidang input variabel, masukkan label di Label input.
- Untuk mengatur nilai default untuk variabel ini saat dasbor pertama kali dibuka, masukkan nilai default-nya dalam Nilai default.

11. Pilih Tambahkan variabel.

    Kotak pilihan FunctionNamedropdown muncul di dekat bagian atas dasbor. Anda dapat memilih sebuah fungsi Lambda di kotak ini dan semua widget yang menggunakan variabel akan menampilkan informasi tentang fungsi yang sudah Anda pilih.

    Nanti, jika Anda menambahkan lebih banyak widget ke dasbor yang menonton metrik Lambda dengan dimensi, mereka akan secara otomatis menggunakan variabel FunctionNametersebut.

Tutorial: Cara membuat sebuah dasbor yang menggunakan pola ekspresi reguler untuk beralih antar Wilayah

Langkah-langkah yang diuraikan dalam prosedur ini menggambarkan cara membuat sebuah dasbor fleksibel yang dapat beralih antar Wilayah. Tutorial ini menggunakan sebuah variabel pola ekspresi reguler, bukan variabel sifat. Untuk tutorial yang menggunakan variabel sifat, silakan lihat Tutorial: Cara membuat dasbor Lambda dengan nama fungsi sebagai variabel.

Untuk banyak kasus penggunaan, Anda dapat membuat sebuah dasbor yang dapat beralih antar Wilayah dengan menggunakan variabel sifat. Tetapi jika widget bergantung pada Amazon Resource Name (ARN) yang menyertakan nama Wilayah, maka Anda harus menggunakan variabel pola untuk mengubah nama Wilayah dalam ARN.

Untuk menggunakan variabel pola dasbor untuk membuat sebuah dasbor multi-Wilayah yang fleksibel

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, Buat dasbor.
3. Masukkan nama untuk dasbor tersebut, dan pilih Buat dasbor.
4. Menambahkan widget ke dasbor. Saat Anda menambahkan widget yang ingin Anda gunakan untuk menampilkan data spesifik Wilayah, Anda tidak boleh menentukan dimensi apa pun

dengan nilai yang hanya muncul di satu Wilayah. Sebagai contoh, untuk metrik-metrik Amazon EC2, tentukan metrik yang digabungkan, bukan metrik yang menggunakan InstanceID sebagai dimensi.

Untuk informasi selengkapnya tentang menambahkan widget ke sebuah dasbor, silakan lihat Buat dan bekerja dengan widget di dasbor CloudWatch .

5. Setelah Anda menambahkan widget tersebut, saat Anda melihat dasbor, pilih Tindakan, Variabel, Buat variabel.
6. Pilih Variabel pola.
7. Untuk Sifat yang mengubah variabel, masukkan nama dasbor Wilayah saat ini, seperti **us-east-2**.

   Anda akan memasukkan Wilayah yang benar jika label yang ada di bawah kotak itu menampilkan widget yang akan dipengaruhi oleh variabel.
8. Untuk Tipe input, untuk kasus penggunaan ini, pilih Tombol radio.
9. Untuk Menentukan bagaimana input diisi, silakan pilih Buat daftar nilai kustom.
10. Untuk Buat nilai kustom Anda, masukkan Wilayah-wilayah tempat Anda ingin saling beralih, dengan satu Wilayah di setiap baris. Setelah masing-masing Wilayah, masukkan koma dan kemudian label yang akan ditampilkan untuk tombol radio itu. Sebagai contoh:

    us-east-1, N. Virginia

    us-east-2, Ohio

    eu-west-3, Paris

    Saat Anda mengisi nilai kustom, panel Pratinjau akan diperbarui untuk menampilkan seperti apa tombol radio nantinya.
11. (Opsional) Untuk pengaturan lainnya, pilih Pengaturan sekunder dan lakukan satu atau beberapa langkah berikut:

    - Untuk melakukan kustomisasi pada nama variabel Anda, masukkan nama di Nama variabel kustom.
    - Untuk melakukan kustomisasi pada label untuk bidang input variabel, masukkan label di Label input. Untuk tutorial ini, masukkan **Region:**.

Jika Anda memasukkan suatu nilai di sini, maka panel Pratinjau akan diperbarui untuk menampilkan seperti apa tombol radio nantinya.

- Untuk mengatur nilai default untuk variabel ini saat dasbor pertama kali dibuka, masukkan nilai default-nya dalam Nilai default.

12. Pilih Tambahkan variabel.

    Dasbor kemudian akan muncul, dengan sebuah label Wilayah: yang terletak di sebelah tombol radio untuk Wilayah tersebut di sekitar bagian atas. Saat Anda beralih antar Wilayah, semua widget yang menggunakan variabel akan menampilkan informasi tentang Wilayah yang Anda pilih.

Salin variabel ke dasbor lain

Setelah Anda membuat sebuah dasbor dengan variabel yang berguna, Anda dapat menyalin variabel ini ke dasbor-dasbor lain yang sudah ada. Untuk informasi selengkapnya tentang variabel dasbor, silakan lihat Membuat dasbor fleksibel dengan variabel dasbor.

Cara menyalin sebuah variabel dasbor ke dasbor yang lain

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, kemudian pilih nama dasbor yang memiliki variabel yang ingin Anda salin. Masukkan string untuk menemukan dasbor-dasbor yang memiliki nama yang cocok, jika diperlukan.
3. Pilih Tindakan, Variabel, Kelola variabel.
4. Pilih tombol radio yang ada di samping variabel yang ingin Anda salin, kemudian pilih Salin ke dasbor lain.
5. Pilih kotak pilihan, kemudian mulailah mengetikkan nama dasbor yang ingin Anda salin variabelnya.
6. Pilih nama dasbor, kemudian pilih Salin variabel.

Buat dan bekerja dengan widget di dasbor CloudWatch

Gunakan topik-topik yang ada di bagian ini untuk membuat dan menggunakan grafik, alarm, dan widget teks di dasbor Anda.

Daftar Isi

Menambahkan atau menghapus grafik dari CloudWatch dasbor

Anda dapat menambahkan grafik yang berisi satu atau beberapa metrik ke dasbor Anda CloudWatch . Jenis-jenis grafik yang dapat Anda tambahkan ke dasbor Anda termasuk Line, Area ditumpuk, Nomor, Pengukur, Batang, dan Pai. Anda dapat menghapus grafik dari dasbor Anda ketika Anda sudah tidak lagi membutuhkannya. Prosedur-prosedur yang diuraikan di bagian ini akan menjelaskan cara menambahkan dan menghapus grafik dari dasbor Anda. Untuk informasi tentang cara mengedit grafik di dasbor, lihat Mengedit grafik di CloudWatch dasbor.

Cara menambahkan grafik ke sebuah dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Pilih simbol +, dan kemudian pilih jenis grafik yang ingin Anda tambahkan ke dasbor Anda, lalu pilih Berikutnya.
    - Jika Anda memilih Line, Area ditumpuk, Batang, atau Pai, pilih Metrik.
4. Pada tab Telusuri, cari atau telusuri metrik untuk grafik, kemudian pilih yang Anda inginkan.

5. (Opsional) Untuk mengubah rentang waktu grafik Anda, pilih salah satu rentang waktu yang telah ditentukan yang ada di bagian atas layar. Rentang waktunya dari 1 jam hingga 1 minggu (1 jam, 3 jam, 12 jam, 1 hari, 3 hari, atau 1 minggu).

   Untuk mengatur rentang waktu Anda sendiri, silakan pilih Kustom.

   - (Opsional) Agar widget ini tetap menggunakan rentang waktu yang sudah Anda pilih tersebut, bahkan jika rentang waktu untuk dasbor lainnya kemudian diubah, maka Anda harus memilih Pertahankan rentang waktu.

6. (Opsional) Untuk mengubah jenis widget pada grafik Anda, gunakan menu geser-turun yang berada di sebelah rentang waktu yang telah ditentukan.

7. (Opsional) Pada Metrik bergrafik, Anda dapat menambahkan sebuah label dinamis ke metrik Anda dan mengubah label, warna label, statistik, dan periode metrik Anda. Anda juga dapat menentukan posisi label berdasarkan sumbu Y dari kiri ke kanan.

   a. Untuk menambahkan sebuah label yang dinamis, pilih Metrik bergrafik, kemudian pilih Tambahkan label dinamis. Label dinamis dapat menampilkan statistik tentang metrik Anda dalam legenda grafik. Label dinamis tersebut akan diperbarui secara otomatis saat dasbor atau grafik Anda diperbarui. Secara default, nilai dinamis yang Anda tambahkan ke label akan muncul di awal label Anda. Untuk informasi selengkapnya, lihat Gunakan label dinamis.

   b. Untuk mengubah warna sebuah metrik, pilih kotak warna yang berada di samping metrik.

   c. Untuk mengubah statistiknya, pilih menu geser-turun pada Statistik, dan kemudian pilih nilai baru. Untuk informasi selengkapnya, silakan lihat Statistik.

   d. Untuk mengubah periode, pilih menu geser-turun pada Periode, kemudian pilih nilai baru.

8. Jika Anda membuat sebuah widget pengukur, maka Anda harus memilih tab Opsi dan menentukan nilai Min dan Maks yang akan digunakan untuk kedua ujung pengukur.

9. (Opsional) Untuk melakukan kustomisasi pada sumbu Y, pilih Opsi. Anda dapat menambahkan sebuah label kustom pada Sumbu Y Kiri di bidang label. Jika grafik Anda menampilkan nilai di sisi kanan sumbu Y, maka Anda dapat melakukan kustomisasi menyesuaikan label itu juga. Anda juga dapat mengatur batas minimum dan maksimum pada nilai sumbu Y Anda, sehingga grafik Anda hanya akan menampilkan rentang nilai yang Anda tentukan.

10. (Opsional) Untuk menambah atau menyunting keterangan horizontal ke grafik garis atau area bertumpuk, atau menambahkan ambang batas untuk widget pengukur, pilih Opsi:

a. Untuk menambahkan keterangan atau ambang batas horizontal, pilih Tambahkan keterangan horizontal atau Tambahkan ambang batas.

b. Untuk Label, masukkan sebuah label untuk keterangannya, dan kemudian pilih ikon tanda centang.

c. Untuk Nilai, pilih ikon pena dan kertas yang berada di sebelah nilai saat ini, dan masukkan nilai baru yang Anda kehendaki. Setelah Anda memasukkan nilai yang Anda kehendaki, pilih ikon tanda centang.

d. Untuk Isi, pilih menu geser-turun dan tentukan bagaimana keterangan Anda akan menggunakan bayangan. Anda dapat memilih Tidak Ada, Di Atas, Antara, atau Di Bawah. Untuk mengubah warna isi, pilih kotak warna yang ada di sebelah keterangan.

e. Untuk Axis, Anda perlu menentukan apakah keterangan Anda muncul di sisi kiri atau kanan sumbu Y.

f. Untuk menyembunyikan keterangan, kosongkan kotak centang yang ada di samping keterangan yang ingin Anda sembunyikan.

g. Untuk menghapus keterangan, pilih X pada Tindakan.

> **Note**
>
> Anda dapat mengulangi langkah-langkah ini untuk menambahkan beberapa keterangan atau ambang batas horizontal ke grafik atau pengukur yang sama.

11. (Opsional) Untuk menambahkan atau menyunting keterangan vertikal, silakan pilih Opsi:

a. Untuk menambahkan sebuah keterangan vertikal, pilih Tambahkan keterangan vertikal.

b. Untuk Label, pilih ikon pena dan kertas yang ada di sebelah keterangan saat ini, kemudian masukkan keterangan baru Anda. Jika Anda hanya ingin menampilkan tanggal dan waktunya, biarkan bidang label tersebut kosong.

c. Untuk Tanggal, silakan pilih tanggal dan waktu saat ini, dan kemudian masukkan tanggal dan waktu baru.

d. Untuk Isi, pilih menu geser-turun, dan tentukan bagaimana keterangan Anda akan menggunakan bayangan. Anda dapat memilih Tidak Ada, Di Atas, Antara, atau Di Bawah. Untuk mengubah warna isiannya, pilih kotak warna yang ada di sebelah keterangan.

e. Untuk menyembunyikan keterangan, kosongkan kotak centang yang ada di samping keterangan yang ingin Anda sembunyikan.

f. Untuk menghapus keterangan, pilih X pada Tindakan.

> **Note**
>
> Anda dapat mengulangi langkah-langkah ini untuk menambahkan beberapa keterangan vertikal ke grafik yang sama.

12. Pilih Buat widget.
13. Pilih Simpan dasbor.

Cara menghapus grafik dari sebuah dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Di sudut kanan atas grafik yang ingin Anda hapus, pilih Tindakan widget, kemudian pilih Hapus.
4. Pilih Simpan dasbor.

Grafik metrik secara manual di dasbor CloudWatch

Jika metrik belum mempublikasikan data dalam 14 hari terakhir, Anda tidak dapat menemukannya saat mencari metrik untuk ditambahkan ke grafik di CloudWatch dasbor. Gunakan langkah-langkah berikut untuk menambahkan metrik apa pun secara manual ke sebuah grafik yang sudah ada.

Untuk menambahkan sebuah metrik yang tidak dapat Anda temukan dalam pencarian ke sebuah grafik

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Dasbor tersebut harus sudah memuat grafik tempat Anda ingin menambahkan metrik. Jika tidak, Anda harus membuat grafik tersebut dan menambahkan metrik apa pun padanya. Untuk informasi selengkapnya, lihat Menambahkan atau menghapus grafik dari CloudWatch dasbor .
4. Pilih Tindakan, Tampilkan/sunting sumber.

Blok JSON akan muncul. Blok tersebut menentukan widget yang ada pada dasbor dan kontennya. Berikut ini adalah contoh salah satu bagian dari blok ini, yang menentukan satu grafik.

```
{
            "type": "metric",
            "x": 0,
            "y": 0,
            "width": 6,
            "height": 3,
            "properties": {
                "view": "singleValue",
                "metrics": [
                    [ "AWS/EBS", "VolumeReadOps", "VolumeId",
 "vol-1234567890abcdef0" ]
                ],
                "region": "us-west-1"
            }
        },
```

Dalam contoh ini, bagian berikut menentukan metrik yang akan ditampilkan dalam grafik ini.

```
[ "AWS/EBS", "VolumeReadOps", "VolumeId", "vol-1234567890abcdef0" ]
```

5. Tambahkan koma setelah tanda kurung tutup jika belum ada dan kemudian tambahkan bagian dengan tanda kurung serupa setelah koma. Di bagian baru ini, tentukan namespace, nama metrik, dan dimensi metrik yang diperlukan yang Anda sedang Anda tambahkan ke grafik. Berikut ini salah satu contohnya.

```
[ "AWS/EBS", "VolumeReadOps", "VolumeId", "vol-1234567890abcdef0" ],
[ "MyNamespace", "MyMetricName", "DimensionName", "DimensionValue" ]
```

   Untuk informasi selengkapnya tentang mengatur format metrik di JSON, silakan lihat Sifat Objek Widget Metrik.

6. Pilih Perbarui.

Bekerja dengan grafik yang sudah ada

Ikuti prosedur-prosedur yang diuraikan di bagian ini untuk menyunting dan melakukan modifikasi widget grafik dasbor yang sudah ada.

Topik

Edit grafik di CloudWatch dasbor

Anda dapat mengedit grafik yang Anda tambahkan ke CloudWatch dasbor Anda. Anda dapat mengubah judul grafik, statistik, atau periode sebuah grafik. Anda dapat menambahkan, memperbarui, dan menghapus metrik-metrik dari grafik Anda. Jika grafik Anda berisi lebih dari satu metrik, Anda dapat mengurangi kesemrawutannya dengan menyembunyikan metrik-metrik yang tidak Anda gunakan. Prosedur-prosedur yang diuraikan di bagian ini menjelaskan tentang cara menyunting sebuah grafik di dasbor Anda. Untuk informasi tentang membuat grafik, lihat Menambahkan atau menghapus grafik dari CloudWatch dasbor.

New interface

Cara menyunting sebuah grafik pada dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Di sudut kanan atas grafik yang ingin Anda sunting, pilih Tindakan widget, kemudian pilih Edit.
4. Untuk mengubah judul grafik, pilih ikon pena dan kertas yang terletak di samping judul saat ini. Masukkan judul baru, lalu pilih Terapkan.
5. (Opsional) Untuk mengubah rentang waktu grafik Anda, pilih salah satu rentang waktu yang telah ditentukan yang ada di bagian atas grafik. Rentang waktunya dari 1 jam hingga 1 minggu (1 jam, 3 jam, 12 jam, 1 hari, 3 hari, atau 1 minggu).

   Untuk mengatur rentang waktu Anda sendiri, silakan pilih Kustom.

- (Opsional) Agar widget ini tetap menggunakan rentang waktu yang sudah Anda pilih tersebut, bahkan jika rentang waktu untuk dasbor lainnya kemudian diubah, maka Anda harus memilih Pertahankan rentang waktu.

6. Untuk mengubah jenis widget grafik Anda, gunakan menu geser-turun yang berada di sebelah rentang waktu yang telah ditentukan.
7. Pada Metrik bergrafik, Anda dapat menambahkan sebuah label dinamis ke metrik Anda dan mengubah label, warna label, statistik, dan periode metrik Anda. Anda juga dapat menentukan posisi label berdasarkan sumbu Y dari kiri ke kanan.

   a. Untuk menambahkan sebuah label dinamis untuk sebuah metrik, pilih Label dinamis. Label dinamis dapat menampilkan statistik tentang metrik dalam legenda grafik. Label dinamis tersebut akan diperbarui secara otomatis saat dasbor atau grafik Anda diperbarui. Secara default, nilai dinamis yang Anda tambahkan ke label akan muncul di awal label. Untuk informasi selengkapnya, lihat Gunakan label dinamis.

   b. Untuk mengubah warna sebuah metrik, pilih kotak warna yang berada di samping metrik.

   c. Untuk mengubah statistik, pilih nilai statistik yang ada di kolom Statistik, dan kemudian pilih nilai baru. Untuk informasi selengkapnya, lihat Statistik.

   d. Untuk mengubah periode, pilih menu geser-turun yang ada di kolom Periode, kemudian pilih nilai baru.

8. Untuk menambahkan atau menyunting keterangan horizontal, pilih Pilihan:

   a. Untuk menambahkan keterangan horizontal, pilih Tambahkan keterangan horizontal.

   b. Untuk Label, pilih ikon pena dan kertas yang ada di samping keterangan saat ini. Kemudian masukkan keterangan baru Anda. Setelah Anda memasukkan keterangan Anda, pilih ikon tanda centang.

   c. Untuk Nilai, pilih ikon pena dan kerta yang ada di sebelah nilai metrik saat ini. Kemudian masukkan nilai metrik baru yang Anda kehendaki. Setelah Anda memasukkan nilai yang Anda kehendaki, silakan pilih ikon tanda centang.

   d. Untuk Isi, pilih menu geser-turun yang ada di bawah kolom, kemudian tentukan bagaimana keterangan Anda akan menggunakan bayangan. Anda dapat memilih Tidak Ada, Di Atas, Antara, atau Di Bawah. Jika Anda memilih Antara, kolom label dan nilai baru lainnya akan ditampilkan.

> **Tip**
>
> Anda dapat mengubah warna isi dengan memilih kotak berwarna yang ada di samping keterangan.

e. Untuk Axis, Anda perlu menentukan apakah keterangan Anda muncul di sisi kiri atau kanan sumbu Y.

f. Untuk menyembunyikan keterangan, Anda harus membatalkan centang kotak yang ada di samping keterangan yang ingin Anda sembunyikan di grafik.

g. Untuk menghapus sebuah keterangan, pilih X di kolom Tindakan.

> **Note**
>
> Anda dapat mengulangi langkah-langkah ini untuk menambahkan beberapa keterangan horizontal ke grafik yang sama.

9. Untuk menambahkan atau menyunting keterangan vertikal, pilih Pilihan:

a. Untuk menambahkan sebuah keterangan vertikal, pilih Tambahkan keterangan vertikal.

b. Untuk Label, pilih ikon pena dan kertas yang ada di samping keterangan saat ini. Kemudian masukkan keterangan baru Anda. Setelah Anda memasukkan keterangan Anda, pilih ikon tanda centang.

> **Tip**
>
> Untuk menampilkan tanggal dan waktunya, biarkan bidang label tersebut kosong.

c. Untuk Tanggal, pilih tanggal dan waktu saat ini. Kemudian masukkan tanggal dan waktu yang baru.

d. Untuk Isi, pilih menu geser-turun yang ada di bawah kolom, kemudian tentukan bagaimana keterangan Anda akan menggunakan bayangan. Anda dapat memilih Tidak Ada, Di Atas, Antara, atau Di Bawah. Jika Anda memilih Antara, kolom label dan nilai baru akan ditampilkan.

> **Tip**
>
> Anda dapat mengubah warna isi dengan memilih kotak berwarna yang ada di samping keterangan.

> **Note**
>
> Anda dapat mengulangi langkah-langkah ini untuk menambahkan beberapa keterangan vertikal ke grafik yang sama.

e. Untuk menyembunyikan keterangan, Anda harus membatalkan centang kotak yang ada di samping keterangan yang ingin Anda sembunyikan di grafik.

f. Untuk menghapus sebuah keterangan, pilih X di kolom Tindakan.

10. Untuk melakukan kustomisasi pada sumbu Y, pilih Opsi. Pada Sumbu Y kiri, Anda dapat memasukkan label kustom untuk Label. Jika grafik menampilkan nilai-nilai pada sumbu Y kanan, maka Anda dapat melakukan kustomisasi pada label itu juga. Anda juga dapat mengatur nilai minimum dan maksimum pada nilai sumbu Y, sehingga grafik hanya akan menampilkan rentang nilai yang Anda tentukan.
11. Setelah Anda selesai membuat perubahan, pilih Perbarui widget.

Cara menyembunyikan atau mengubah posisi legenda grafik

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Di sudut kanan atas grafik yang ingin Anda sunting, pilih Tindakan widget. Pilih Legenda kemudian pilih Tersembunyi, Bawah, atau Kanan.

Cara menyembunyikan metrik untuk sebuah grafik pada sebuah dasbor untuk sementara waktu

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Pilih kotak berwarna untuk metrik yang ingin Anda sembunyikan di footer grafik. Sebuah tanda X akan muncul di kotak warna tersebut saat Anda mengarahkan kursor ke atasnya, dan kotak tersebut akan berubah menjadi abu-abu saat Anda memilihnya.

4. Untuk mengembalikan metrik tersembunyi, hilangkan tanda X yang ada di kotak abu-abu.

Original interface

Cara menyunting sebuah grafik pada dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Arahkan kursor ke sudut kanan atas grafik yang ingin Anda sunting. Pilih Tindakan widget, kemudian pilih Sunting.
4. Untuk mengubah judul grafik, pilih ikon pensil yang terletak di samping judul saat ini, kemudian masukkan judul baru.
5. Untuk mengubah rentang waktu grafik, pilih salah satu rentang waktu yang telah ditentukan sebelumnya di area atas grafik. Rentang waktunya berkisar dari 1 jam hingga 1 minggu (1 jam, 3 jam, 12 jam, 1 hari, 3 hari, atau 1 minggu).

   - Untuk mengatur rentang waktu Anda sendiri, silakan pilih Kustom.
6. Untuk mengubah jenis widget grafik Anda, pilih tab Opsi grafik. Anda dapat memilih Garis, Area ditumpuk, Nomor, Batang, atau Pai.

   > **Tip**
   >
   > Anda dapat mengubah jenis widget grafik Anda dengan memilihnya dari menu geser-turun yang berada di sebelah rentang waktu yang telah ditentukan.

7. Pada Metrik bergrafik, Anda dapat menambahkan sebuah label dinamis ke metrik Anda dan mengubah label, warna label, statistik, dan periode metrik Anda. Anda juga dapat menentukan posisi label berdasarkan sumbu Y dari kiri ke kanan.

   a. Untuk menambahkan sebuah label dinamis untuk sebuah metrik, pilih Label dinamis. Label dinamis dapat menampilkan statistik tentang metrik dalam legenda grafik. Label dinamis tersebut akan diperbarui secara otomatis saat dasbor atau grafik Anda diperbarui. Secara default, nilai dinamis yang Anda tambahkan ke label akan muncul di awal label. Untuk informasi selengkapnya, lihat Gunakan label dinamis.

   b. Untuk mengubah warna sebuah metrik, pilih kotak warna yang berada di samping metrik.

   c. Untuk mengubah statistik, pilih nilai statistik yang ada di kolom Statistik, dan kemudian pilih nilai baru. Untuk informasi selengkapnya, lihat Statistik.

d. Untuk mengubah periode, pilih menu geser-turun yang ada di kolom Periode, kemudian pilih nilai baru.

8. Untuk menambahkan atau menyunting keterangan horizontal, pilih Pilihan grafik:

   a. Untuk menambahkan keterangan horizontal, pilih Tambahkan keterangan horizontal.

   b. Untuk Label, pilih ikon pensil yang ada di sebelah keterangan saat ini. Kemudian masukkan keterangan baru Anda. Setelah Anda memasukkan keterangan Anda, pilih ikon tanda centang.

   c. Untuk Nilai, pilih ikon pensil yang ada di sebelah nilai metrik saat ini. Kemudian masukkan nilai metrik baru yang Anda kehendaki. Setelah Anda memasukkan nilai yang Anda kehendaki, silakan pilih ikon tanda centang.

   d. Untuk Isi, pilih menu geser-turun yang ada di bawah kolom, kemudian tentukan bagaimana keterangan Anda akan menggunakan bayangan. Anda dapat memilih Tidak Ada, Di Atas, Antara, atau Di Bawah. Jika Anda memilih Antara, kolom label dan nilai baru akan ditampilkan.

      > **Tip**
      >
      > Anda dapat mengubah warna isi dengan memilih kotak berwarna yang ada di samping keterangan.

   e. Untuk Axis, Anda perlu menentukan apakah keterangan Anda muncul di sisi kiri atau kanan sumbu Y.

   f. Untuk menyembunyikan keterangan, Anda harus membatalkan centang kotak yang ada di samping keterangan yang ingin Anda sembunyikan di grafik.

   g. Untuk menghapus sebuah keterangan, pilih X di kolom Tindakan.

      > **Note**
      >
      > Anda dapat mengulangi langkah-langkah ini untuk menambahkan beberapa keterangan horizontal ke grafik yang sama.

9. Untuk menambahkan atau menyunting keterangan vertikal, pilih Pilihan grafik:

   a. Untuk menambahkan sebuah keterangan vertikal, pilih Tambahkan keterangan vertikal.

b. Untuk Label, pilih ikon pensil yang ada di sebelah keterangan saat ini. Kemudian masukkan keterangan baru Anda. Setelah Anda memasukkan keterangan Anda, pilih ikon tanda centang.

> **Tip**
>
> Untuk menampilkan tanggal dan waktunya, biarkan bidang label tersebut kosong.

c. Untuk Tanggal, pilih ikon pensil yang ada di samping tanggal dan waktunya. Kemudian masukkan tanggal dan waktu yang baru.

d. Untuk Isi, pilih menu geser-turun yang ada di bawah kolom, kemudian tentukan bagaimana keterangan Anda akan menggunakan bayangan. Anda dapat memilih Tidak Ada, Di Atas, Antara, atau Di Bawah. Jika Anda memilih Antara, kolom label dan nilai baru akan ditampilkan.

> **Tip**
>
> Anda dapat mengubah warna isi dengan memilih kotak berwarna yang ada di samping keterangan.

> **Note**
>
> Anda dapat mengulangi langkah-langkah ini untuk menambahkan beberapa keterangan vertikal ke grafik yang sama.

e. Untuk menyembunyikan keterangan, Anda harus membatalkan centang kotak yang ada di samping keterangan yang ingin Anda sembunyikan di grafik.

f. Untuk menghapus sebuah keterangan, pilih X di kolom Tindakan.

10. Untuk melakukan kustomisasi pada sumbu Y, pilih Pilihan grafik. Pada Sumbu Y kiri, Anda dapat memasukkan label kustom untuk Label. Jika grafik menampilkan nilai-nilai pada sumbu Y kanan, maka Anda dapat melakukan kustomisasi pada label itu juga. Anda juga dapat mengatur nilai minimum dan maksimum pada nilai sumbu Y, sehingga grafik hanya akan menampilkan rentang nilai yang Anda tentukan.

11. Setelah Anda selesai membuat perubahan, pilih Perbarui widget.

Cara menyembunyikan atau mengubah posisi legenda grafik

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Arahkan kursor ke sudut kanan atas grafik yang ingin Anda sunting, kemudian pilih Tindakan widget. Pilih Legenda, kemudian pilih Tersembunyi, Bawah, atau Kanan.

Cara menyembunyikan metrik untuk sebuah grafik pada sebuah dasbor untuk sementara waktu

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Pilih kotak berwarna untuk metrik yang ingin Anda sembunyikan di footer grafik. Sebuah tanda X akan muncul di kotak warna tersebut saat Anda mengarahkan kursor ke atasnya, dan kotak tersebut akan berubah menjadi abu-abu saat Anda memilihnya.
4. Untuk mengembalikan metrik tersembunyi, hilangkan tanda X yang ada di kotak abu-abu.

Memindahkan atau mengubah ukuran grafik di dasbor CloudWatch

Anda dapat mengatur dan mengubah ukuran grafik di dasbor Anda CloudWatch .

Cara memindahkan sebuah grafik pada dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Lakukan salah satu langkah berikut:
   - Arahkan kursor ke judul grafik hingga ikon pilihan ditampilkan. Pilih dan seret grafik ke lokasi baru di dasbor.
   - Untuk memindahkan widget ke kiri atas atau kiri bawah dasbor, pilih elipsis vertikal yang ada di kanan atas widget untuk membuka menu Tindakan widget. Kemudian pilih Pindahkan, dan pilih ke mana harus memindahkan widget tersebut.
4. Pilih Simpan dasbor.

Cara mengubah ukuran grafik

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.

2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Cara menambah atau mengurangi ukuran, arahkan kursor ke grafik dan seret sudut kanan bawah grafik. Hentikan menyeret sudut jika Anda ukurannya sudah sesuai keinginan Anda.
4. Pilih Simpan dasbor.

Cara memperbesar grafik untuk sementara waktu

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Pilih grafik. Atau, arahkan kursor ke judul grafik kemudian pilih Tindakan widget, Perbesar.

Ganti nama grafik di dasbor CloudWatch

Anda dapat mengubah nama default yang ditetapkan CloudWatch ke grafik di dasbor Anda.

Cara mengubah nama sebuah grafik pada dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Arahkan kursor ke atas judul grafik kemudian pilih Tindakan widget dan Sunting.
4. Di layar Sunting grafik, di dekat bagian atas, pilih judul grafik.
5. Untuk Judul, masukkan nama baru dan pilih Ok (tanda centang). Di sudut kanan bawah layar Sunting grafik, kemudian pilih Perbarui widget.

Tambahkan widget penjelajah metrik ke dasbor CloudWatch

Widget penjelajah metrik mencakup grafik dari beberapa sumber daya yang memiliki tanda yang sama, atau berbagi sifat sumber daya yang sama, misalnya tipe instans. Widget ini selalu terbaru, karena sumber daya yang cocok sudah dibuat atau dihapus. Menambahkan widget penjelajah metrik ke dasbor akan membantu Anda dalam memecahkan masalah lingkungan dengan lebih efisien.

Sebagai contoh, Anda dapat memantau armada instans EC2 dengan menetapkan tanda yang mewakili lingkungan, seperti produksi atau pengujian. Kemudian, Anda dapat menggunakan tanda ini untuk menyaring dan menggabungkan metrik operasi seperti `CPUUtilization` untuk memahami kondisi kesehatan dan performa instans EC2 yang terkait dengan masing-masing tag.

Langkah-langkah berikut menjelaskan tentang cara menambahkan sebuah widget penjelajah metrik ke dasbor dengan menggunakan konsol. Anda juga dapat menambahkannya secara terprogram atau dengan menggunakan. AWS CloudFormationUntuk informasi selengkapnya, lihat Definisi Objek Widget Penjelajah Metrik dan AWS::CloudWatch::Dashboard.

Menambahkan sebuah widget penjelajah metrik ke dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor.
3. Pilih nama dasbor tempat di mana Anda ingin menambahkan widget penjelajah metrik.
4. Pilih simbol +.
5. Pilih Penjelajah, kemudian pilih Berikutnya.

   > **Note**
   >
   > Anda harus memilih tampilan dasbor yang baru untuk dapat menambahkan widget Penjelajah Metrik. Untuk memilih, pilih Dasbor di panel navigasi, kemudian pilih coba antarmuka baru yang ada di banner di bagian atas halaman.

6. Lakukan salah satu langkah berikut:
   - Untuk menggunakan sebuah template, pilih Widget penjelajah yang telah diisi sebelumnya dan kemudian pilih template yang akan digunakan.
   - Untuk membuat sebuah visualisasi kustom, pilih Widget penjelajah kosong.
7. Pilih Buat.

   Jika Anda menggunakan sebuah template, widget akan muncul di dasbor Anda dengan metrik-metrik yang telah dipilih. Jika sudah puas dengan widget penjelajah dan dasbor yang Anda siapkan, pilih Simpan dasbor.

   Jika Anda tidak menggunakan template, lanjutkan ke langkah-langkah berikut.
8. Pada widget baru di Penjelajah, di kotak Metrik, pilih satu metrik atau semua metrik yang tersedia dari sebuah layanan.

   Setelah memilih sebuah metrik, Anda dapat secara opsional mengulangi langkah ini untuk menambahkan lebih banyak metrik.

9. Untuk setiap metrik yang dipilih, CloudWatch menampilkan statistik yang akan digunakan segera setelah nama metrik. Untuk mengubah ini, pilih nama statistik, dan kemudian pilih statistik yang Anda inginkan.
10. Pada Dari, pilih sebuah tanda atau properti sumber daya untuk menyaring hasil Anda.

    Setelah melakukan ini, Anda dapat secara opsional mengulangi langkah ini untuk memilih lebih banyak tanda atau properti sumber daya.

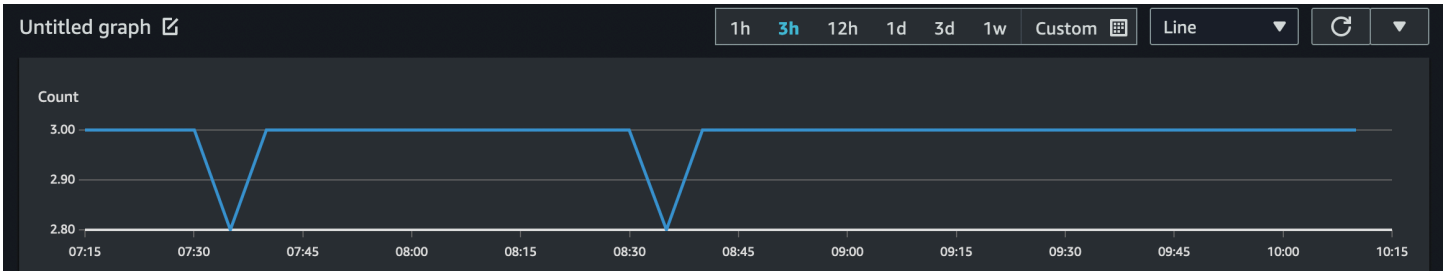
    Jika Anda memilih beberapa nilai dari properti yang sama, seperti dua tipe instans EC2, penjelajah akan menampilkan semua sumber daya yang cocok dengan properti yang dipilih. Hal ini akan diperlakukan sebagai operasi ATAU.

    Jika Anda memilih properti atau tanda yang berbeda, seperti tanda **`Production`** dan tipe instans M5, hanya sumber daya yang cocok dengan semua pilihan ini yang akan ditampilkan. Hal ini dianggap sebagai operasi DAN.
11. (Opsional) Untuk Digabungkan dengan, pilih statistik yang digunakan untuk menggabungkan metrik. Lalu, di samping untuk, pilih cara menggabungkan metrik dari daftar. Anda dapat menggabungkan bersama semua sumber daya yang saat ini ditampilkan, atau digabungkan dengan satu tanda atau properti sumber daya.

    Tergantung pada bagaimana Anda memilih untuk menggabungkan, hasilnya bisa saja berupa satu rangkaian waktu tunggal atau beberapa rangkaian waktu.
12. Pada Dibagi oleh, Anda dapat memilih untuk membagi grafik tunggal dengan beberapa rangkaian waktu ke dalam beberapa grafik. Pembagian ini dapat dilakukan dengan beragam kriteria, yang dapat Anda pilih pada Dibagi oleh.
13. Pada Pilihan grafik, Anda dapat menyempurnakan grafik dengan mengubah periode, jenis grafik, penempatan keterangan, dan tata letak.
14. Jika sudah puas dengan widget penjelajah dan dasbor yang Anda siapkan, pilih Simpan dasbor.

Menambahkan atau menghapus widget baris di CloudWatch dasbor

Dengan widget garis, Anda akan dapat membandingkan metrik selama periode waktu. Anda juga dapat menggunakan fitur zoom peta mini yang dimiliki widget untuk memeriksa bagian-bagian grafik garis tanpa bergonta-ganti antara tampilan yang diperbesar dan diperkecil. Prosedur di bagian ini menjelaskan cara menambah dan menghapus widget baris di CloudWatch dasbor. Untuk informasi tentang menggunakan fitur zoom peta mini yang dimiliki widget dengan grafik garis, silakan lihat Memperbesar grafik garis atau area bertumpuk.



Untuk menambahkan sebuah widget garis ke dasbor

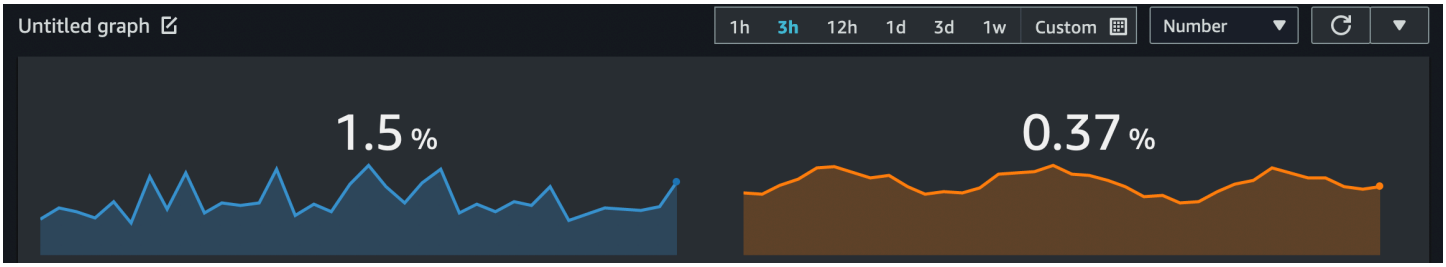
1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Pilih simbol +, kemudian pilih Garis.
4. Pilih Metrik.
5. Pilih Telusuri, dan pilih metrik yang ingin Anda buat grafiknya.
6. Pilih Buat widget, kemudian pilih Simpan dasbor.

Untuk menghapus widget garis dari sebuah dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Di sudut kanan atas widget garis yang ingin Anda hapus, pilih Tindakan wawasan, kemudian pilih Hapus.
4. Pilih Simpan dasbor.

Menambahkan atau menghapus widget angka dari CloudWatch dasbor

Dengan widget angka tersebut, Anda akan dapat melihat nilai dan tren metrik terbaru sesaat setelah muncul. Karena widget angka menyertakan fitur sparkline, Anda dapat membuat visualisasi untuk bagian atas dan bawah tren metrik dalam satu grafik. Prosedur di bagian ini menjelaskan cara menambah dan menghapus widget angka dari CloudWatch dasbor.



> **Note**
>
> Hanya antarmuka baru yang mendukung fitur sparkline ini. Saat Anda membuat sebuah widget angka, fitur sparkline akan disertakan secara otomatis.

Untuk menambahkan widget angka ke sebuah dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Pilih simbol +, kemudian pilih Angka.
4. Pada tab Telusuri, cari atau telusuri metrik yang ingin Anda tampilkan.
5. (Opsional) Untuk mengubah warna fitur sparkline, pilih Metrik bergrafik, dan pilih kotak warna yang terletak di sebelah label metrik. Menu akan muncul di tempat Anda dapat memilih warna yang berbeda atau memasukkan kode warna heksa enam digit untuk menentukan warna.
6. (Opsional) Untuk mematikan fitur sparkline, pilih Opsi. Pada Sparkline, kotak centang.
7. (Opsional) Untuk mengubah rentang waktu widget angka Anda, pilih salah satu rentang waktu yang telah ditentukan sebelumnya di area atas widget. Rentang waktunya dari 1 jam hingga 1 minggu (1 jam, 3 jam, 12 jam, 1 hari, 3 hari, atau 1 minggu).

   Untuk mengatur rentang waktu Anda sendiri, silakan pilih Kustom.

   - (Opsional) Agar widget ini tetap menggunakan rentang waktu yang sudah Anda pilih tersebut, bahkan jika rentang waktu untuk dasbor lainnya kemudian diubah, maka Anda harus memilih Pertahankan rentang waktu.
8. (Opsional) Agar widget angka menampilkan agregat (1h, 3h, 12h, 1d, 3d, atau 1w).

   Untuk mengatur rentang waktu Anda sendiri, silakan pilih Kustom.

- (Opsional) Agar widget ini menampilkan rata-rata nilai metrik di seluruh rentang waktu, alih-alih nilai terbaru, pilih Opsi, Nilai rentang waktu menunjukkan nilai dari seluruh rentang waktu.

9. Pilih Buat widget, kemudian pilih Simpan dasbor.

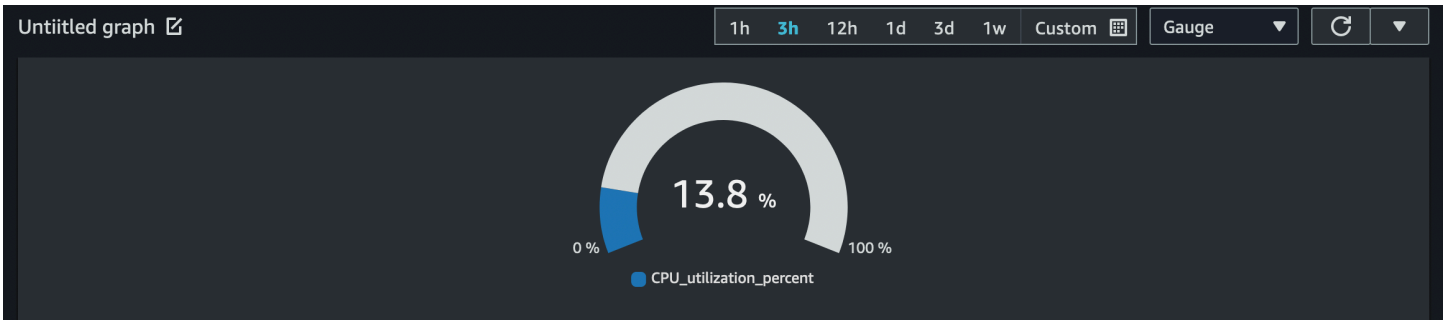
> **Tip**
>
> Anda dapat mematikan fitur sparkline dari widget angka yang ada di layar dasbor. Di sudut kanan atas widget angka yang ingin Anda ubah, pilih Tindakan widget. Pilih Sparkline, kemudian pilih Sembunyikan sparkline.

Untuk menghapus widget angka dari sebuah dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, kemudian pilih dasbor yang berisi widget angka yang ingin Anda hapus.
3. Di sudut kanan atas widget angka yang ingin Anda hapus, pilih Tindakan wawasan, kemudian pilih Hapus.
4. Pilih Simpan dasbor.

Tambahkan atau hapus widget pengukur dari dasbor CloudWatch

Dengan widget pengukur, Anda akan dapat membuat visualisasi nilai metrik yang berada di antara rentang waktu. Sebagai contoh, Anda dapat menggunakan widget pengukur untuk membuat grafik persentase dan pemanfaatan CPU, sehingga Anda akan dapat mengamati dan mendiagnosis setiap masalah performa yang terjadi. Prosedur di bagian ini menjelaskan cara menambahkan dan menghapus widget pengukur dari CloudWatch dasbor.



> **Note**
>
> Hanya antarmuka baru di CloudWatch konsol yang mendukung pembuatan widget pengukur. Anda harus menetapkan rentang waktu pengukur saat Anda membuat widget ini.

Cara menambahkan widget pengukur ke sebuah dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Dari layar dasbor, pilih simbol +, kemudian pilih Pengukur.
4. Pilih Telusuri, kemudian pilih metrik yang ingin Anda buat grafiknya.
5. Pilih Opsi. Pada Rentang pengukur, tetapkan nilai untuk Min. dan Maks. Untuk persentase, seperti pemanfaatan CPU, sebaiknya Anda menetapkan nilai untuk `Min` dengan `0`, dan `Max` dengan `100`.
6. (Opsional) Untuk mengubah warna widget pengukur, pilih Metrik bergrafik dan pilih kotak warna yang terletak di sebelah label metrik. Menu akan muncul di tempat Anda dapat memilih warna yang berbeda atau memasukkan kode warna heksa enam digit untuk menentukan warna.
7. Pilih Buat widget, kemudian pilih Simpan dasbor.

Cara menghapus widget pengukur dari sebuah dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, kemudian pilih dasbor yang berisi widget pengukur yang ingin Anda hapus.
3. Di sudut kanan atas widget pengukur yang ingin Anda hapus, pilih Tindakan wawasan, kemudian pilih Hapus.
4. Pilih Simpan dasbor.

Tambahkan widget khusus ke CloudWatch dasbor

Widget khusus adalah widget CloudWatch dasbor yang dapat memanggil AWS Lambda fungsi apa pun dengan parameter khusus. Kemudian widget ini akan menampilkan HTML atau JSON yang dikembalikan. Widget kustom adalah sebuah cara sederhana untuk membangun tampilan data kustom di sebuah dasbor. Jika Anda dapat menulis kode Lambda dan membuat HTML, maka Anda

akan dapat membuat sebuah widget kustom yang berguna. Selain itu, Amazon juga menyediakan beberapa widget kustom default yang dapat Anda buat tanpa kode apa pun.

Saat Anda membuat fungsi Lambda untuk digunakan sebagai widget kustom, sebaiknya Anda menyertakan awalan customWidget dalam nama fungsinya. Hal ini akan membantu Anda mengetahui fungsi Lambda mana yang aman digunakan saat Anda menambahkan widget kustom ke dasbor Anda.

Widget kustom berperilaku seperti widget-widget yang lain di dasbor Anda. Widget kustom dapat disegarkan dan disegarkan secara otomatis, diubah ukurannya, dan juga dapat dipindahkan. Widget kustom bereaksi terhadap rentang waktu dasbor.

Jika Anda telah menyiapkan fungsionalitas lintas akun CloudWatch konsol, Anda dapat menambahkan widget khusus yang dibuat di satu akun ke dasbor di akun lain. Untuk informasi selengkapnya, lihat Konsol CloudWatch lintas akun lintas Wilayah.

Anda juga dapat menggunakan widget khusus di situs web Anda sendiri dengan menggunakan fitur berbagi CloudWatch dasbor. Untuk informasi selengkapnya, lihat Berbagi CloudWatch dasbor.

Topik

Detail mengenai widget kustom

Widget kustom bekerja seperti berikut ini:

1. CloudWatch Dasbor memanggil fungsi Lambda yang berisi kode widget. Dasbor tersebut akan memasukkan parameter-parameter kustom apa pun yang ditentukan dalam widget.
2. Fungsi Lambda kemudian akan mengembalikan string HTML, JSON, atau Markdown. Markdown dikembalikan sebagai JSON dalam format berikut ini:

   ```
   {"markdown":"markdown content"}
   ```

3. Dasboard akan menampilkan HTML atau JSON.

Jika fungsi mengembalikan HTML, maka sebagian besar tanda HTML didukung. Anda dapat menggunakan gaya Cascading Style Sheets (CSS) dan Scalable Vector Graphics (SVG) untuk membuat tampilan-tampilan yang canggih.

Gaya default elemen HTML seperti tautan dan tabel mengikuti gaya CloudWatch dasbor. Anda dapat menyesuaikan gaya ini dengan menggunakan gaya-gaya inline, dengan menggunakan tanda `<style>`. Anda juga dapat menonaktifkan gaya default dengan menyertakan elemen HTML tunggal dengan kelas `cwdb-no-default-styles`. Contoh berikut menonaktifkan gaya default: `<div class="cwdb-no-default-styles"></div>`.

Setiap panggilan oleh sebuah widget kustom ke Lambda akan menyertakan sebuah elemen `widgetContext` dengan konten-konten berikut, untuk menyediakan pengembang fungsi Lambda dengan informasi konteks yang berguna.

```
{
    "widgetContext": {
        "dashboardName": "Name-of-current-dashboard",
        "widgetId": "widget-16",
        "accountId": "012345678901",
        "locale": "en",
        "timezone": {
            "label": "UTC",
            "offsetISO": "+00:00",
            "offsetInMinutes": 0
        },
        "period": 300,
        "isAutoPeriod": true,
        "timeRange": {
            "mode": "relative",
            "start": 1627236199729,
            "end": 1627322599729,
            "relativeStart": 86400012,
            "zoom": {
                "start": 1627276030434,
                "end": 1627282956521
             }
        },
        "theme": "light",
        "linkCharts": true,
        "title": "Tweets for Amazon website problem",
        "forms": {
            "all": {}
```

```
        },
        "params": {
            "original": "param-to-widget"
        },
        "width": 588,
        "height": 369
    }
}
```

Penataan gaya CSS default

Widget kustom menyediakan elemen-elemen penataan gaya CSS default berikut ini:

- Anda dapat menggunakan kelas CSS btn untuk menambahkan sebuah tombol. Ia akan mengubah anchor (<a>) menjadi sebuah tombol seperti pada contoh berikut:

  ```
  <a class="btn" href=https://amazon.com”>Open Amazon</a>
  ```

- Anda dapat menggunakan kelas CSS btn btn-primary untuk menambahkan sebuah tombol utama.
- Elemen-elemen berikut ditata gayanya secara default: tabel, pilih, header (h1, h2, dan h3), teks yang telah diformat sebelumnya (pra), input, dan area teks.

Menggunakan parameter describe

Kami sangat menyarankan agar Anda mendukung parameter describe dalam fungsi Anda, bahkan jika parameter itu hanya mengembalikan sebuah string kosong. Jika Anda tidak mendukungnya, dan parameter itu dipanggil di widget kustom, maka parameter itu akan menampilkan konten widget seolah-olah itu adalah dokumentasi.

Jika Anda menyertakan parameter describe, maka fungsi Lambda akan mengembalikan dokumentasi dalam format Markdown dan tidak akan melakukan hal lain.

Saat Anda membuat sebuah widget kustom di konsol, setelah Anda memilih fungsi Lambda, tombol Dapatkan dokumentasi akan muncul. Jika Anda memilih tombol ini, maka fungsi akan diinvokasi dengan parameter describe dan akan mengembalikan dokumentasi fungsi. Jika dokumentasi diformat dengan baik dalam penurunan harga, CloudWatch parsing entri pertama dalam dokumentasi yang dikelilingi oleh tiga karakter backtick tunggal (```) di YAMAL. Kemudian, ia akan secara otomatis mengisi dokumentasi dalam parameter. Berikut ini adalah satu contoh dokumentasi yang diformat dengan baik.

```
``` yaml
echo: <h1>Hello world</h1>
```
```

Keamanan dan JavaScript

Untuk alasan keamanan, tidak JavaScript diperbolehkan dalam HTML yang dikembalikan. Menghapus masalah eskalasi izin JavaScript mencegah, di mana penulis fungsi Lambda menyuntikkan kode yang dapat berjalan dengan izin yang lebih tinggi daripada pengguna yang melihat widget di dasbor.

Jika HTML yang dikembalikan berisi JavaScript kode atau kerentanan keamanan lain yang diketahui, itu dibersihkan dari HTML sebelum dirender di dasbor. Sebagai contoh, tanda **<iframe>** dan **<use>** tidak diizinkan dan dihapus.

Kustom Widget tidak akan berjalan secara default di sebuah dashboard. Sebaliknya, Anda harus secara eksplisit mengizinkan sebuah widget kustom untuk dijalankan jika Anda mempercayai fungsi Lambda yang dipanggilnya. Anda dapat memilih untuk mengizinkannya sekali atau mengizinkannya selalu, untuk widget individu dan seluruh dasbor. Anda juga dapat menolak izin untuk masing-masing widget dan juga menolaknya untuk seluruh dasbor.

Interaktivitas dalam widget kustom

Meskipun tidak JavaScript diperbolehkan, ada cara lain untuk memungkinkan interaktivitas dengan HTML yang dikembalikan.

- Setiap elemen yang ada dalam HTML yang dikembalikan dapat di-tanda dengan konfigurasi khusus dalam tanda `<cwdb-action>`, yang dapat menampilkan informasi dalam pop-up, meminta konfirmasi dengan klik, dan memanggil fungsi Lambda ketika elemen itu dipilih. Misalnya, Anda dapat menentukan tombol yang memanggil AWS API apa pun menggunakan fungsi Lambda. HTML yang dikembalikan dapat diatur untuk menggantikan konten widget Lambda yang ada, atau ditampilkan di dalam sebuah modal.
- HTML yang dikembalikan dapat mencakup tautan-tautan yang akan membuka konsol baru, membuka halaman pelanggan lain, atau memuat dasbor lainnya.
- HTML dapat menyertakan atribut `title` untuk sebuah elemen, yang akan memberikan informasi tambahan jika pengguna mengarahkan kursor ke elemen itu.
- Elemen tersebut dapat mencakup pemilih CSS, seperti `:hover`, yang dapat menginvokasi animasi atau efek CSS lainnya. Anda juga dapat menampilkan atau menyembunyikan elemen di halaman.

Definisi dan penggunaan <cwdb-action>

Elemen `<cwdb-action>` mendefinisikan perilaku pada elemen sebelumnya. Isi dari `<cwdb-action>` adalah HTML yang akan ditampilkan atau blok parameter JSON yang akan diteruskan ke fungsi Lambda, salah satunya.

Berikut ini adalah contoh dari elemen `<cwdb-action>`.

```
<cwdb-action
     action="call|html"
     confirmation="message"
     display="popup|widget"
     endpoint="<lambda ARN>"
     event="click|dblclick|mouseenter">

     html | params in JSON
</cwdb-action>
```

- tindakan— Nilai yang valid adalah `call`, yang memanggil sebuah fungsi Lambda, dan `html`, yang akan menampilkan HTML apa pun yang terkandung di dalam `<cwdb-action>`. Bawaannya adalah `html`.
- konfirmasi— Menampilkan sebuah pesan konfirmasi yang harus diakui sebelum tindakan diambil, memungkinkan pelanggan untuk membatalkan tindakan.
- display— Nilai yang valid adalah popup dan `widget`, yang akan menggantikan konten dari widget itu sendiri. Bawaannya adalah `widget`.
- titik akhir— Amazon Resource Name (ARN) dari fungsi Lambda yang akan dipanggil. Ini diperlukan jika `action` adalah `call`.
- event— Menentukan peristiwa pada elemen sebelumnya yang akan menginvokasi tindakan. Nilai yang valid adalah `click`, `dblclick`, dan `mouseenter`. Peristiwa `mouseenter` ini hanya dapat digunakan bila dikombinasikan dengan tindakan `html`. Bawaannya adalah `click`.

Contoh

Berikut ini adalah sebuah contoh cara menggunakan tanda `<cwdb-action>` untuk membuat tombol yang akan me-reboot instans Amazon EC2 dengan menggunakan sebuah panggilan fungsi Lambda. Ia akan menampilkan keberhasilan atau kegagalan panggilan dengan sebuah pop-up.

```
<a class="btn">Reboot Instance</a>
```

```
<cwdb-action action="call" endpoint="arn:aws:lambda:us-
east-1:123456:function:rebootInstance" display="popup">
       { "instanceId": "i-342389adbfef" }
</cwdb-action>
```

Contoh berikutnya akan menampilkan informasi lebih banyak dengan sebuah pop-up.

```
<a>Click me for more info in popup</a>
<cwdb-action display="popup">
   <h1>Big title</h1>
   More info about <b>something important</b>.
</cwdb-action>
```

Contoh ini adalah sebua tombol Berikutnya yang menggantikan konten dar sebuah widget dengan panggilan ke fungsi Lambda.

```
<a class="btn btn-primary">Next</a>
<cwdb-action action="call" endpoint="arn:aws:lambda:us-
east-1:123456:function:nextPage">
   { "pageNum": 2 }
</cwdb-action>
```

Membuat sebuah widget kustom

Untuk membuat widget kustom, Anda dapat menggunakan salah satu sampel yang disediakan oleh AWS, atau Anda dapat membuat sendiri. AWS Sampel termasuk sampel di keduanya JavaScript dan Python, dan dibuat oleh tumpukan. AWS CloudFormation Untuk melihat sebuah daftar sampel, silakan lihat Widget kustom sampel.

Untuk membuat widget khusus di CloudWatch dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Pilih simbol +.
4. Pilih Widget kustom.
5. Gunakan salah satu metode berikut ini:
   - Untuk menggunakan contoh widget kustom yang disediakan oleh AWS, lakukan hal berikut:

     a. Pilih sampel di kotak geser-turun.

AWS CloudFormation Konsol diluncurkan di browser baru. Di AWS CloudFormation konsol, lakukan hal berikut:

b. (Opsional) Sesuaikan nama AWS CloudFormation tumpukan.

c. Buat pilihan-pilihan untuk setiap parameter yang digunakan oleh sampel.

d. Pilih Saya mengakui yang AWS CloudFormation mungkin membuat sumber daya IAM, dan pilih Buat tumpukan.

- Untuk membuat widget kustom Anda sendiri yang disediakan oleh AWS, lakukan hal berikut:

a. Pilih Berikutnya.

b. Pilih salah satu, memilih fungsi Lambda Anda dari daftar, atau masukkan Amazon Resource Name (ARN) -nya. Jika Anda memilihnya dari daftar, maka Anda harus menentukan juga Wilayah tempat fungsi berada dan versi yang akan digunakan.

c. Untuk Parameter, buat pilihan-pilihan untuk setiap parameter yang digunakan oleh fungsi.

d. Masukkan judul untuk widget.

e. Untuk Pembaruan aktif, konfigurasikan kapan widget harus diperbarui (kapan fungsi Lambda harus dipanggil lagi). Hal ini bisa dilakukan dengan satu atau beberapa hal berikut: Segarkan untuk memperbaruinya saat dasbor melakukan penyegaran kembali otomatis, Ubah ukuran untuk memperbaruinya setiap kali widget diubah ukurannya, atau Rentang Waktu untuk memperbaruinya setiap kali rentang waktu dasbor disesuaikan, termasuk saat grafik diperbesar.

f. Jika Anda sudah merasa puas dengan pratinjaunya, pilih Buat widget.

Widget kustom sampel

AWS menyediakan contoh widget kustom di keduanya JavaScript dan Python. Anda dapat membuat sampel widget ini dengan menggunakan tautan untuk masing-masing widget dalam daftar ini. Atau, Anda dapat membuat dan menyesuaikan widget dengan menggunakan CloudWatch konsol. Tautan dalam daftar ini membuka AWS CloudFormation konsol dan menggunakan tautan AWS CloudFormation buat cepat untuk membuat widget khusus.

Anda juga dapat mengakses sampel widget kustom pada GitHub.

Mengikuti daftar ini, contoh lengkap widget Echo ditunjukkan untuk masing-masing bahasa.

JavaScript

Contoh widget kustom di JavaScript

- Echo — Echo dasar yang dapat Anda gunakan untuk menguji bagaimana HTML ditampilkan di sebuah widget kustom, tanpa harus menulis sebuah widget baru.
- Hello world — Widget pemula yang sangat mendasar.
- Debugger widget kustom — Sebuah debugger widget yang menampilkan informasi yang berguna tentang lingkungan runtime Lambda.
- Wawasan CloudWatch Log Kueri — Jalankan dan edit kueri Wawasan CloudWatch Log.
- Jalankan kueri Amazon Athena — Menjalankan dan menyunting kueri Athena.
- Panggil AWS API - Panggil AWS API hanya-baca apa pun dan tampilkan hasilnya dalam format JSON.
- Grafik CloudWatch bitmap cepat - Render CloudWatch grafik menggunakan di sisi server, untuk tampilan cepat.
- Widget teks dari CloudWatch dasbor - Menampilkan widget teks pertama dari CloudWatch dasbor yang ditentukan.
- CloudWatch data metrik sebagai tabel - Menampilkan data CloudWatch metrik mentah dalam tabel.
- Tabel Amazon EC2 — Menampilkan instans EC2 teratas berdasarkan pemanfaatan CPU. Widget ini juga menyertakan sebuah tombol Reboot, yang dinonaktifkan secara default.
- AWS CodeDeploy penerapan - Menampilkan CodeDeploy penerapan.
- AWS Cost Explorer laporan - Menampilkan laporan tentang biaya setiap AWS layanan untuk rentang waktu yang dipilih.
- Menampilkan konten URL eksternal — Menampilkan sebuah konten URL yang dapat diakses secara eksternal.
- Menampilkan objek Amazon S3 — Menampilkan sebuah objek dalam sebuah bucket Amazon S3 di akun Anda.
- Bagan lingkaran SVG sederhana — Contoh widget berbasis SVG grafis.

Python

Widget kustom sampel di Python

- Echo — Echo dasar yang dapat digunakan untuk menguji bagaimana HTML ditampilkan di sebuah widget kustom, tanpa harus menulis sebuah widget baru.
- Hello world — Widget pemula yang sangat mendasar.
- Debugger widget kustom — Sebuah debugger widget yang menampilkan informasi yang berguna tentang lingkungan runtime Lambda.
- Panggil AWS API - Panggil AWS API hanya-baca apa pun dan tampilkan hasilnya dalam format JSON.
- Grafik CloudWatch bitmap cepat - Render CloudWatch grafik menggunakan di sisi server, untuk tampilan cepat.
- Kirim snapshot dasbor melalui email — Mengambil snapshot dari dasbor saat ini dan dikirimkan ke penerima email.
- Kirim snapshot dasbor ke Amazon S3 Mengambil snapshot dari dasbor saat ini dan kemudian disimpan di Amazon S3.
- Widget teks dari CloudWatch dasbor - Menampilkan widget teks pertama dari CloudWatch dasbor yang ditentukan.
- Menampilkan konten URL eksternal — Menampilkan sebuah konten URL yang dapat diakses secara eksternal.
- Pembaca RSS — Menampilkan umpan RSS.
- Menampilkan objek Amazon S3 — Menampilkan sebuah objek dalam sebuah bucket Amazon S3 di akun Anda.
- Bagan lingkaran SVG sederhana — Contoh widget berbasis SVG grafis.

Widget gema di JavaScript

Berikut ini adalah contoh widget Echo di JavaScript.

```
const DOCS = `
## Echo
A basic echo script. Anything passed in the \`\`\`echo\`\`\` parameter is returned as
 the content of the custom widget.
### Widget parameters
```

```
Param | Description
---|---
**echo** | The content to echo back

### Example parameters
\`\`\` yaml
echo: <h1>Hello world</h1>
\`\`\`
`;

exports.handler = async (event) => {
    if (event.describe) {
        return DOCS;
    }

    let widgetContext = JSON.stringify(event.widgetContext, null, 4);
    widgetContext = widgetContext.replace(/</g, '&lt;');
    widgetContext = widgetContext.replace(/>/g, '&gt;');

    return `${event.echo || ''}<pre>${widgetContext}</pre>`;
};
```

Widget Echo dengan Python

Berikut ini adalah sampel widget Echo dalam Python.

```
import json

DOCS = """
## Echo
A basic echo script. Anything passed in the ```echo``` parameter is returned as the
 content of the custom widget.
### Widget parameters
Param | Description
---|---
**echo** | The content to echo back

### Example parameters
``` yaml
echo: <h1>Hello world</h1>
```"""

def lambda_handler(event, context):
    if 'describe' in event:
```

```
        return DOCS

    echo = event.get('echo', '')
    widgetContext = event.get('widgetContext')
    widgetContext = json.dumps(widgetContext, indent=4)
    widgetContext = widgetContext.replace('<', '&lt;')
    widgetContext = widgetContext.replace('>', '&gt;')

    return f'{echo}<pre>{widgetContext}</pre>'
```

Widget Echo dalam Java

Berikut ini adalah sampel widget Echo dalam Java.

```
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

public class Handler implements RequestHandler<Event, String>{

  static String DOCS = ""
    + "## Echo\n"
    + "A basic echo script. Anything passed in the ```echo``` parameter is returned as
 the content of the custom widget.\n"
    + "### Widget parameters\n"
    + "Param | Description\n"
    + "---|---\n"
    + "**echo** | The content to echo back\n\n"
    + "### Example parameters\n"
    + "```yaml\n"
    + "echo: <h1>Hello world</h1>\n"
    + "```\n";

  Gson gson = new GsonBuilder().setPrettyPrinting().create();

  @Override
  public String handleRequest(Event event, Context context) {

    if (event.describe) {
      return DOCS;
```

```
    }

    return (event.echo != null ? event.echo : "") + "<pre>" +
 gson.toJson(event.widgetContext) + "</pre>";
  }
}

class Event {

    public boolean describe;
    public String echo;
    public Object widgetContext;

    public Event() {}

    public Event(String echo, boolean describe, Object widgetContext) {
        this.describe = describe;
        this.echo = echo;
        this.widgetContext = widgetContext;
    }
}
```

Menambahkan atau menghapus widget teks dari CloudWatch dasbor

Sebuah widget teks memuat blok teks dalam format Markdown. Anda dapat menambahkan, mengedit, atau menghapus widget teks dari CloudWatch dasbor Anda.

Cara menambahkan sebuah widget teks ke sebuah dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Pilih simbol +.
4. Pilih Teks.
5. Untuk Markdown, tambahkan dan format teks Anda menggunakan Markdown kemudian pilih Buat widget.
6. Untuk membuat widget teks menjadi transparan, pilih Latar belakang transparan.
7. Pilih Simpan dasbor.

Untuk menyunting widget teks di sebuah dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Arahkan kursor ke sudut kanan atas blok teks dan pilih Tindakan widget. Kemudian pilih Edit.
4. Perbarui teks tersebut sesuai kebutuhan, kemudian pilih Perbarui widget.
5. Pilih Simpan dasbor.

Untuk menghapus widget teks dari sebuah dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Arahkan kursor ke sudut kanan atas blok teks dan pilih Tindakan widget. Kemudian, pilih Hapus.
4. Pilih Simpan dasbor.

Menambahkan atau menghapus widget alarm dari CloudWatch dasbor

Cara menambahkan widget alarm ke sebuah dasbor, pilih salah satu opsi berikut:

- Tambahkan satu alarm dalam sebuah widget, yang menampilkan grafik metrik alarm dan juga menampilkan status alarm.
- Tambahkan widget status alarm, yang menampilkan status beberapa alarm dalam kisi. Hanya nama alarm dan status saat ini yang akan ditampilkan, Grafik tidak ditampilkan. Anda dapat menyertakan hingga 100 alarm dalam satu widget status alarm.

Cara menambahkan satu alarm, termasuk grafiknya, ke sebuah dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Alarm, silakan pilih alarm yang akan ditambahkan, dan kemudian pilih Tambahkan ke Dasbor.
3. Pilih sebuah dasbor, pilih jenis widget (Garis, Area yang ditumpuk, atau Nomor), dan kemudian pilih Tambahkan ke dasbor.
4. Untuk melihat alarm Anda di dasbor, pilih Dasbor dalam panel navigasi kemudian pilih dasbor.
5. (Opsional) Untuk membuat grafik alarm menjadi lebih besar untuk sementara waktu, pilih grafiknya.

6. (Opsional) Untuk mengubah jenis widget, arahkan kursor ke judul grafik, pilih Tindakan widget, kemudian pilih Jenis widget.

Cara menambahkan widget status alarm ke sebuah dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Pilih simbol +.
4. Pilih Status alarm.
5. Centang kotak yang ada di samping alarm yang ingin Anda tambahkan ke widget, dan kemudian pilih Buat widget.
6. Pilih Tambahkan ke dasbor.

Cara menghapus widget alarm dari sebuah dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Arahkan kursor ke widget, pilih Tindakan widget, kemudian pilih Hapus.
4. Pilih Simpan dasbor. Jika Anda mencoba keluar dari dasbor sebelum menyimpan perubahan, Anda akan diminta untuk menyimpan atau membuang perubahan tersebut.

Menambahkan atau menghapus widget tabel data dari CloudWatch dasbor

Dengan widget tabel data, Anda dapat melihat titik data mentah dari metrik Anda dan ringkasan singkat dari data mentah tersebut. Karena widget tabel data bukan sebuah bagan untuk melakukan abstraksi terhadap data aktual dari Anda, akan lebih mudah untuk memahami titik data yang disajikan. Prosedur di bagian ini menjelaskan cara menambahkan dan menghapus widget tabel data dari CloudWatch dasbor.

Label	Min	Max	Sum	Average	11/20 06:00	11/20 00:00	11/19 18:00	11/19 12:00	11/ 06:
TestMetric295	991	1,000	12k	998	996	1,000	997	999	
TestMetric296	995	1,000	12k	998	995	1,000	1,000	998	
TestMetric297	991	1,000	12k	998	998	1,000	999	997	
TestMetric298	994	1,000	12k	997	996	999	995	995	
TestMetric3	993	1,000	12k	998	1,000	999	999	1,000	
TestMetric299	995	999	12k	998	999	995	999	998	
TestMetric30	994	999	12k	998	999	998	999	999	
StackMetric2	99	99.9	1.2k	99.6	99.2	99.7	99.5	99.8	
StackMetric20	99	100	1.19k	99.5	100	99.1	99.4	99.4	
StackMetric21	97.5	100	1.19k	99.4	99.6	99.7	97.6	99.8	

Properti tabel

Tabel data memiliki seperangkat properti default yang tidak memerlukan adanya perubahan apa pun untuk dilakukan pada opsi atau sumber. Properti-properti ini mencakup kolom label lengket, semua kolom ringkasan yang diaktifkan, titik data yang dibulatkan, dan unitnya yang dikonversi.

Masing-masing widget tabel data dapat memiliki properti-properti berikut. Informasi tentang masing-masing properti mencakup cara mengonfigurasinya di sumber JSON dasbor. Untuk informasi selengkapnya tentang JSON dasbor, silakan lihat Struktur dan Sintaks Tubuh Dasbor.

Ringkasan

Kolom-kolom ringkasan merupakan properti baru yang diperkenalkan dalam widget tabel data. Kolom-kolom ini adalah subset spesifik dari ringkasan tabel Anda saat ini. Sebagai contoh, ringkasan Jumlah adalah jumlah dari semua titik data yang ditampilkan dalam barisnya. Kolom ringkasan tidak sama dengan CloudWatch statistik. Diwakili dalam sumber sebagai:

```
"table": {
       "summaryColumns": [
           "MIN",
           "MAX",
           "SUM",
           "AVG"
       ]
   },
```

Ambang batas

Gunakan ini untuk menerapkan ambang batas terhadap tabel Anda. Ketika sebuah titik data berada dalam ambang batas, selnya akan disorot dengan warna ambang batasnya. Diwakili dalam sumber sebagai:

```
"annotations": {
    "horizontal": [
        {
            "label": string,
            "value": int,
            "fill": "above" | "below"
        }
    ]
}
```

Satuan di kolom label

Untuk menampilkan unit apa yang dikaitkan dengan metrik, Anda dapat mengaktifkan opsi ini untuk menampilkan unit di kolom label yang ada di samping label. Diwakili dalam sumber sebagai:

```
"yAxis": {
    "left": {
        "showUnits": true | false
    }
}
```

Balikkan baris dan kolom

Hal ini akan mengubah tabel sehingga titik data berubah dari kolom menjadi baris, dan dari metrik menjadi kolom. Diwakili dalam sumber sebagai:

```
 "table": {
    "layout": "vertical" | "horizontal"
}
```

Kolom ringkasan lengket

Hal ini akan membuat kolom ringkasan lengket, sehingga kolom itu akan tetap terlihat saat Anda menggulir. Labelnya sudah lengket. Diwakili dalam sumber sebagai:

```
"table": {
```

```
    "stickySummary": true | false
}
```

Tampilkan hanya kolom ringkasan

Hal ini akan mencegah kolom titik data ditampilkan, sehingga hanya label dan kolom ringkasan saja yang ditampilkan. Diwakili dalam sumber sebagai:

```
 "table": {
    "showTimeSeriesData": false | true
}
```

Data langsung

Hal ini akan menampilkan titik data terbaru, bahkan jika itu belum sepenuhnya diagregasi. Diwakili dalam sumber sebagai:

```
"liveData": true | false
```

Format widget angka

Ini akan menampilkan digit sebanyak mungkin yang bisa dimuat di dalam sel, sebelum pembulatan dan konversi. Diwakili dalam sumber sebagai:

```
"singleValueFullPrecision": true | false
```

Cara menambahkan sebuah widget tabel data ke sebuah dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Pilih tombol +, pilih Tabel data, dan kemudian pilih Berikutnya.
4. Pada tab Telusuri, cari atau telusuri metrik-metrik yang ingin Anda tampilkan dalam widget tabel. Kemudian pilih metrik-metriknya.
5. (Opsional) Untuk mengubah tata letak tabel, silakan pilih tab Opsi dan kemudian pilih Balikkan baris dan kolom.

   Anda juga dapat menggunakan tab Opsi untuk mengubah kolom apa yang akan dimunculkan dalam tabel dan menampilkan unit yang digunakan di kolom Label.

> **Tip**
>
> Untuk menampilkan ambang batas yang lebih akurat, silakan pilih Tampilkan digit sebanyak mungkin sebelum pembulatan.

6. (Opsional) Untuk mengubah rentang waktu widget tabel data Anda, silakan pilih salah satu rentang waktu yang telah ditentukan sebelumnya di area atas widget. Rentang waktu dari 1 jam hingga 1 minggu. Untuk mengatur rentang waktu Anda sendiri, silakan pilih Kustom.
7. (Opsional) Untuk mengubah rentang waktu widget tabel data Anda, silakan pilih salah satu rentang waktu yang telah ditentukan sebelumnya di area atas widget. Rentang waktu dari 1 jam hingga 1 minggu. Untuk mengatur rentang waktu Anda sendiri, silakan pilih Kustom.
8. (Opsional) Agar widget ini tetap menggunakan rentang waktu yang sudah Anda pilih tersebut, bahkan jika rentang waktu untuk dasbor lainnya kemudian diubah, Anda harus memilih Pertahankan rentang waktu.
9. Pilih Buat widget dan kemudian pilih Simpan dasbor.

Cara menghapus sebuah widget tabel ke sebuah dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Di sudut kanan atas widget yang ingin Anda hapus, pilih Tindakan widget, dan kemudian pilih Hapus.
4. Pilih Simpan dasbor.

Tautkan dan putuskan tautan grafik di dasbor CloudWatch

Anda dapat menautkan grafik di dasbor Anda secara bersamaan, sehingga ketika Anda memperbesar atau memperkecil satu grafik, grafik lainnya akan memperbesar atau memperkecil dalam waktu bersamaan. Anda dapat membatalkan tautan grafik untuk membatasi perbesaran terhadap satu grafik.

Cara menautkan grafik pada sebuah dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.

3. Pilih Tindakan, kemudian Tautkan grafik.

Cara membatalkan tautan grafik pada sebuah dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Hapus pilihan pada Tindakan dan Tautkan grafik.

Berbagi CloudWatch dasbor

Anda dapat membagikan CloudWatch dasbor Anda dengan orang-orang yang tidak memiliki akses langsung ke AWS akun Anda. Hal ini akan memungkinkan Anda berbagi dasbor lintas tim, dengan pemangku kepentingan, dan dengan orang-orang yang ada di luar organisasi Anda. Bahkan, Anda dapat menampilkan dasbor pada layar-layar besar di area tim, atau menyematkannya di Wiki dan halaman web lainnya.

> **Warning**
>
> Semua orang yang Anda ajak berbagi dasbor akan mendapatkan izin yang tercantum di Izin-izin yang diberikan kepada orang yang Anda berbagi dasbor dengannya untuk akun tersebut. Jika Anda membagikan dasbor secara publik, maka setiap orang yang memiliki tautan ke dasbor tersebut akan mendapatkan izin-izin ini. `ec2:DescribeTags`Izin `cloudwatch:GetMetricData` dan tidak dapat dicakup ke metrik tertentu atau instans EC2, sehingga orang yang memiliki akses ke dasbor dapat menanyakan semua CloudWatch metrik dan nama serta tag dari semua instans EC2 di akun.

Ketika Anda berbagi dasbor, Anda akan dapat menetapkan orang yang dapat melihat dasbor dengan tiga cara:

- Bagikan satu dasbor dan tentukan sebanyak lima alamat email orang yang dapat melihat dasbor. Masing-masing pengguna ini bisa membuat kata sandi mereka sendiri yang harus dimasukkan untuk melihat dasbor.
- Anda bisa membagikan satu dasbor tunggal secara publik, sehingga setiap orang yang memiliki tautan akan dapat melihat dasbor tersebut.

- Bagikan semua CloudWatch dasbor di akun Anda dan tentukan penyedia sistem masuk tunggal (SSO) pihak ketiga untuk akses dasbor. Semua pengguna yang merupakan anggota dari daftar penyedia SSO ini akan dapat mengakses semua dasbor yang ada di akun. Untuk memungkinkan hal ini, Anda harus mengintegrasikan penyedia SSO tersebut dengan Amazon Cognito. Penyedia SSO harus mendukung Security Assertion Markup Language (SAML). Untuk informasi selengkapnya tentang Amazon Cognito, silakan lihat Apa itu Amazon Cognito?

Berbagi dasbor tidak dikenakan biaya, tetapi widget di dalam dasbor bersama dikenakan biaya dengan tarif standar. CloudWatch Untuk informasi selengkapnya tentang CloudWatch harga, lihat CloudWatch Harga Amazon.

Saat Anda berbagi dasbor, sumber daya Amazon Cognito dibuat di Wilayah AS Timur (Virginia N.).

Important

Jangan mengubah nama sumber daya dan pengidentifikasi yang sudah dibuat berdasarkan proses berbagi dasbor. Ini termasuk sumber daya Amazon Cognito dan IAM. Melakukan modifikasi terhadap sumber daya ini akan dapat menyebabkan fungsionalitas dasbor yang dibagikan menjadi tidak terduga dan salah.

Note

Jika Anda berbagi sebuah dasbor yang memiliki widget metrik dengan keterangan alarm, maka orang-orang yang berbagi dasbor dengan Anda tidak akan melihat widget tersebut. Mereka malah akan melihat sebuah widget kosong dengan tulisan yang menyampaikan bahwa widget tidak tersedia. Anda masih akan melihat widget-widget metrik dengan keterangan alarm saat Anda melihat dasbor sendiri.

Izin-izin yang diperlukan untuk berbagi sebuah dasbor

Untuk dapat membagikan dasbor menggunakan salah satu metode berikut dan untuk melihat dasbor yang telah dibagikan, Anda harus masuk sebagai seorang pengguna atau Anda memiliki peran IAM yang memiliki izin tertentu.

Agar dapat berbagi dasbor, pengguna atau peran IAM Anda harus menyertakan izin-izin yang disertakan dalam pernyataan kebijakan berikut:

```
{
    "Effect": "Allow",
    "Action": [
        "iam:CreateRole",
        "iam:CreatePolicy",
        "iam:AttachRolePolicy",
        "iam:PassRole"
    ],
    "Resource": [
        "arn:aws:iam::*:role/service-role/CWDBSharing*",
        "arn:aws:iam::*:policy/*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "cognito-idp:*",
        "cognito-identity:*",
    ],
    "Resource": [
        "*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "cloudwatch:GetDashboard",
    ],
    "Resource": [
        "*"
        // or the ARNs of dashboards that you want to share
    ]
}
```

Untuk dapat melihat dasbor yang dibagikan, tetapi tidak dapat membagikan dasbor, seorang pengguna atau pemilik peran IAM dapat menyertakan sebuah pernyataan kebijakan yang serupa dengan hal berikut:

```
{
    "Effect": "Allow",
    "Action": [
        "cognito-idp:*",
```

```
            "cognito-identity:*"
        ],
        "Resource": [
            "*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "cloudwatch:ListDashboards",
        ],
        "Resource": [
            "*"
        ]
    }
```

Izin-izin yang diberikan kepada orang yang Anda berbagi dasbor dengannya

Saat Anda berbagi dasbor, CloudWatch buat peran IAM di akun yang memberikan izin berikut kepada orang-orang yang berbagi dasbor dengan Anda:

- `cloudwatch:GetInsightRuleReport`
- `cloudwatch:GetMetricData`
- `cloudwatch:DescribeAlarms`
- `ec2:DescribeTags`

> **Warning**
>
> Semua orang yang Anda berbagi dasbor dengannya mendapatkan izin-izin ini untuk akun tersebut. Jika Anda membagikan dasbor secara publik, maka setiap orang yang memiliki tautan ke dasbor tersebut akan mendapatkan izin-izin ini. `ec2:DescribeTags`Izin `cloudwatch:GetMetricData` dan tidak dapat dicakup ke metrik tertentu atau instans EC2, sehingga orang yang memiliki akses ke dasbor dapat menanyakan semua CloudWatch metrik dan nama serta tag dari semua instans EC2 di akun.

Saat Anda membagikan dasbor, secara default izin yang CloudWatch dibuat membatasi akses hanya ke alarm dan aturan Wawasan Kontributor yang ada di dasbor saat dibagikan. Jika Anda menambahkan alarm baru atau peraturan Wawasan Kontributor baru ke dasbor tersebut dan menginginkan alarm dan aturan itu juga dilihat oleh orang yang Anda ajak berbagi dasbor,maka Anda harus memperbarui kebijakan untuk mengizinkan sumber daya ini.

Berbagi sebuah dasbor tunggal dengan pengguna tertentu

Gunakan langkah-langkah di bagian ini untuk berbagi dasbor dengan sebanyak lima alamat email yang Anda pilih.

> **Note**
>
> Secara default, widget CloudWatch Log apa pun di dasbor tidak terlihat oleh orang yang berbagi dasbor dengan Anda. Untuk informasi selengkapnya, lihat Mengizinkan orang yang Anda ajak berbagi dasbor melihat widget tabel log.
> Secara bawaan, widget alarm gabungan apa pun yang ada di dasbor tidak akan terlihat oleh orang yang Anda ajak berbagi dasbor. Untuk informasi selengkapnya, lihat Memungkinkan orang yang Anda ajak berbagi dasbor melihat alarm gabungan.

Cara berbagi sebuah dasbor dengan pengguna tertentu

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor.
3. Pilih nama dasbor Anda.
4. Pilih Tindakan, Bagikan dasbor.
5. Di samping Bagikan dasbor Anda dan minta nama pengguna dan kata sandi, pilih Mulai berbagi.
6. Pada Tambahkan alamat email, masukkan alamat-alamat email yang Anda ingin berbagi dasbor dengannya. Anda dapat memasukkan sebanyak lima alamat email.
7. Ketika semua alamat email sudah Anda masukkan, baca perjanjian dan pilih kotak konfirmasi. Kemudian pilih Pratinjau kebijakan.
8. Konfirmasi bahwa sumber daya yang akan dibagikan adalah apa yang Anda inginkan, kemudian pilih Konfirmasi dan buat tautan yang dapat dibagikan.
9. Pada halaman berikutnya, pilih Salin tautan ke clipboard. Kemudian, Anda dapat menempelkan tautan ini ke dalam email dan mengirimkannya kepada para pengguna yang ingin Anda undang.

Mereka secara otomatis akan menerima email terpisah dengan nama pengguna dan kata sandi sementara yang bisa digunakan untuk terhubung ke dasbor.

Berbagi sebuah dasbor tunggal secara publik

Lakukan langkah-langkah yang diuraikan dalam bagian ini untuk berbagi sebuah dasbor secara publik. Hal ini berguna untuk menampilkan sebuah dasbor di layar besar di sebuah ruang tim, atau Anda dapat menyematkannya di halaman Wiki.

Important

Berbagi dasbor secara publik akan membuat dasbor dapat diakses oleh orang-orang yang memiliki tautan, tanpa ada autentikasi. Lakukan ini hanya untuk dasbor-dasbor yang tidak memuat informasi sensitif.

Note

Secara default, widget CloudWatch Log apa pun di dasbor tidak terlihat oleh orang yang berbagi dasbor dengan Anda. Untuk informasi selengkapnya, lihat Mengizinkan orang yang Anda ajak berbagi dasbor melihat widget tabel log.
Secara bawaan, widget alarm gabungan apa pun yang ada di dasbor tidak akan terlihat oleh orang yang Anda ajak berbagi dasbor. Untuk informasi selengkapnya, lihat Memungkinkan orang yang Anda ajak berbagi dasbor melihat alarm gabungan.

Cara berbagi sebuah dasbor secara publik

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor.
3. Pilih nama dasbor Anda.
4. Pilih Tindakan, Bagikan dasbor.
5. Di samping Bagikan dasbor Anda secara publik, pilih Mulai berbagi.
6. Masukkan **`Confirm`** di kotak teks.
7. Baca perjanjian dan pilih kotak konfirmasi. Kemudian pilih Pratinjau kebijakan.

8. Konfirmasi bahwa sumber daya yang akan dibagikan adalah apa yang Anda inginkan, kemudian pilih Konfirmasi dan buat tautan yang dapat dibagikan.
9. Pada halaman berikutnya, pilih Salin tautan ke clipboard. Kemudian, Anda akan dapat membagikan tautan ini. Setiap orang yang berbagi tautan dengan Anda akan dapat mengakses dasbor, tanpa perlu memberikan kredensial.

Bagikan semua CloudWatch dasbor di akun dengan menggunakan SSO

Gunakan langkah-langkah di bagian ini untuk berbagi semua dasbor yang ada di akun Anda dengan para pengguna dengan menggunakan sistem masuk tunggal (SSO).

> **Note**
>
> Secara default, widget CloudWatch Log apa pun di dasbor tidak terlihat oleh orang yang berbagi dasbor dengan Anda. Untuk informasi selengkapnya, lihat Mengizinkan orang yang Anda ajak berbagi dasbor melihat widget tabel log.
> Secara bawaan, widget alarm gabungan apa pun yang ada di dasbor tidak akan terlihat oleh orang yang Anda ajak berbagi dasbor. Untuk informasi selengkapnya, lihat Memungkinkan orang yang Anda ajak berbagi dasbor melihat alarm gabungan.

Untuk berbagi CloudWatch dasbor Anda dengan pengguna yang ada dalam daftar penyedia SSO

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor.
3. Pilih nama dasbor Anda.
4. Pilih Tindakan, Bagikan dasbor.
5. Pilih Buka CloudWatch Pengaturan.
6. Jika penyedia SSO yang Anda inginkan tidak tercantum dalam Penyedia SSO yang tersedia, pilih Kelola penyedia SSO dan ikuti petunjuk-petunjuk yang dijelaskan di Siapkan SSO untuk berbagi CloudWatch dasbor.

   Kemudian kembali ke CloudWatch konsol dan segarkan browser. Penyedia SSO yang Anda aktifkan sekarang akan dimunculkan dalam daftar tersebut.
7. Pilih penyedia layanan SSO yang Anda kehendaki di daftar Penyedia SSO yang tersedia.

8. Pilih Simpan perubahan.

Siapkan SSO untuk berbagi CloudWatch dasbor

Untuk menyiapkan berbagi dasbor melalui penyedia masuk tunggal pihak ketiga yang mendukung SAML, Anda harus menyelesaikan langkah-langkah ini.

> **Important**
>
> Kami sangat menyarankan agar Anda tidak membagikan dasbor dengan menggunakan penyedia SSO non-SAML. Karena hal tersebut akan menimbulkan risiko di mana Anda membiarkan pihak ketiga dapat mengakses dasbor akun Anda.secara tidak sengaja.

Cara mengatur penyedia SSO agar memungkinkan berbagi dasbor

1. Integrasikan penyedia SSO dengan Amazon Cognito. Untuk informasi selengkapnya, silakan lihat Mengintegrasikan Penyedia Identitas SAML Pihak Ketiga dengan Kolam Pengguna Amazon Cognito.
2. Unduh file XML metadata dari penyedia SSO Anda.
3. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
4. Pada panel navigasi, silakan pilih Pengaturan.
5. Di bagian Berbagi dasbor, pilih Konfigurasikan.
6. Pilih Kelola penyedia SSO.

   Langkah ini akan membuka konsol Amazon Cognito di Wilayah AS Timur (Virginia Utara) (us-east-1). Jika Anda tidak melihat Kolam Pengguna, ada kemungkinan konsol Amazon Cognito telah dibuka di Wilayah yang berbeda. Jika demikian, Anda harus mengubah Wilayah menjadi AS Timur (Virginia Utara) us-east-1 dan melanjutkan dengan langkah selanjutnya.
7. Pilih CloudWatchDashboardSharing kumpulan.
8. Pada panel navigasi, silakan pilih Penyedia identitas.
9. Pilih SAML.
10. Masukkan nama untuk penyedia SSO Anda di Nama penyedia.
11. Pilih Pilih file, kemudian pilih berkas XML metadata yang Anda unduh di langkah 1.

12. Pilih Buat penyedia.

Melihat berapa banyak dasbor Anda yang dibagikan

Anda dapat menggunakan CloudWatch konsol untuk melihat berapa banyak CloudWatch dasbor Anda yang saat ini dibagikan dengan orang lain.

Cara melihat berapa banyak dasbor Anda yang sedang dibagikan

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Pengaturan.
3. Bagian Berbagi dasbor menampilkan berapa banyak dasbor yang Anda bagikan.
4. Untuk melihat dasbor mana yang Anda bagikan, pilih ***nomor*** dasbor bersama di Nama pengguna dan kata sandi dan di Dasbor publik.

Melihat dasbor mana yang Anda dibagikan

Anda dapat menggunakan CloudWatch konsol untuk melihat dasbor mana yang sedang dibagikan dengan orang lain.

Cara melihat dasbor mana yang sedang Anda bagikan

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor.
3. Dalam daftar dasbor, silakan lihat kolom Bagikan. Dasbor yang memiliki ikon terisi di kolom ini, artinya dasbor itu saat ini sedang dibagikan.
4. Untuk melihat pengguna mana yang sedang berbagi dasbor, pilih nama dasbor, kemudian pilih Tindakan, Bagikan dasbor.

   Halaman Bagikan dasbor ***nama dasbor*** menampilkan bagaimana dasbor-dasbor itu dibagikan. Jika Anda mau, Anda bisa berhenti berbagi dasbor dengan memilih Berhenti berbagi.

Berhenti berbagi satu atau beberapa dasbor

Anda dapat berhenti berbagi satu dasbor yang sudah Anda bagikan, atau berhenti berbagi semua dasbor yang sudah Anda bagikan secara sekaligus.

Cara berhenti berbagi satu dasbor tunggal

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor.
3. Pilih nama dasbor yang Anda bagikan.
4. Pilih Tindakan, Bagikan dasbor.
5. Pilih Berhenti berbagi.
6. Di kotak konfirmasi, pilih Hentikan berbagi.

Cara berhenti berbagi semua dasbor yang sudah Anda bagikan

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Pengaturan.
3. Di bagian Berbagi dasbor, pilih Berhenti berbagi semua dasbor.
4. Dalam kotak konfirmasi, pilih Berhenti berbagi semua dasbor.

Meninjau izin dasbor yang dibagikan dan mengubah cakupan izin

Gunakan langkah-langkah yang diuraikan di bagian ini jika Anda ingin meninjau izin pengguna dasbor yang sudah Anda bagikan, atau mengubah cakupan izin dasbor yang sudah Anda bagikan.

Cara meninjau izin dasbor yang sudah Anda bagikan

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor.
3. Pilih nama dasbor yang Anda bagikan.
4. Pilih Tindakan, Bagikan dasbor.
5. Di Sumber Daya, pilih Peran IAM.
6. Di konsol IAM, pilih kebijakan yang ditampilkan.
7. (Opsional) Untuk membatasi alarm mana yang dapat dilihat pengguna dasbor yang sudah Anda bagikan, pilih Edit kebijakan dan pindahkan izin `cloudwatch:DescribeAlarms` dari posisi saat ini ke sebuah pernyataan `Allow` baru yang berisi ARN hanya dari alarm yang Anda inginkan untuk dilihat oleh pengguna dasbor yang sudah Anda bagikan. Lihat contoh berikut ini.

```
{
```

```
   "Effect": "Allow",
    "Action": "cloudwatch:DescribeAlarms",
    "Resource": [
        "AlarmARN1",
        "AlarmARN2"
    ]
}
```

Jika Anda melakukan hal ini, Anda harus memastikan untuk menghapus izin `cloudwatch:DescribeAlarms` dari bagian kebijakan saat ini yang terlihat seperti ini:

```
{
   "Effect": "Allow",
    "Action": [
        "cloudwatch:GetInsightRuleReport",
        "cloudwatch:GetMetricData",
        "cloudwatch:DescribeAlarms",
        "ec2:DescribeTags"
    ],
    "Resource": "*"
}
```

8. (Opsional) Untuk membatasi cakupan aturan-aturan Wawasan Kontributor yang dapat dilihat oleh pengguna dasbor yang sudah Anda bagikan, pilih Edit kebijakan dan pindahkan `cloudwatch:GetInsightRuleReport` dari posisinya saat ini ke sebuah pernyataan `Allow` baru yang mencantumkan ARN hanya dari aturan-aturan Wawasan Kontributor yang Anda inginkan untuk dilihat oleh pengguna dasbor yang sudah Anda bagikan. Lihat contoh berikut ini.

```
{
   "Effect": "Allow",
    "Action": "cloudwatch:GetInsightRuleReport",
    "Resource": [
        "PublicContributorInsightsRuleARN1",
        "PublicContributorInsightsRuleARN2"
    ]
}
```

Jika Anda melakukan hal ini, Anda harus memastikan untuk menghapus `cloudwatch:GetInsightRuleReport` dari bagian kebijakan saat ini yang terlihat seperti ini:

```
{
```

```
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetInsightRuleReport",
                "cloudwatch:GetMetricData",
                "cloudwatch:DescribeAlarms",
                "ec2:DescribeTags"
            ],
            "Resource": "*"
        }
```

Memungkinkan orang yang Anda ajak berbagi dasbor melihat alarm gabungan

Saat Anda berbagi dasbor, secara default widget alarm gabungan yang ada di dasbor tidak akan terlihat oleh orang yang Anda ajak berbagi dasbor. Agar widget alarm gabungan terlihat oleh mereka, Anda perlu menambahkan sebuah izin `DescribeAlarms: *` pada kebijakan berbagi dasbor. Izin itu akan terlihat seperti ini:

```
{
    "Effect": "Allow",
    "Action": "cloudwatch:DescribeAlarms",
    "Resource": "*"
}
```

Warning

Pernyataan kebijakan sebelumnya memberikan akses ke semua alarm yang ada di akun. Untuk mengurangi ruang lingkup `cloudwatch:DescribeAlarms`, Anda harus menggunakan sebuah pernyataan `Deny`. Anda dapat menambahkan sebuah pernyataan `Deny` ke kebijakan tersebut dan menentukan ARN alarm yang ingin Anda kunci. Pernyataan penolakan itu akan terlihat seperti berikut ini:

```
{
    "Effect": "Allow",
    "Action": "cloudwatch:DescribeAlarms",
    "Resource": "*"
},
{
    "Effect": "Deny",
```

```
    "Action": "cloudwatch:DescribeAlarms",
    "Resource": [
        "SensitiveAlarm1ARN",
        "SensitiveAlarm1ARN"
    ]
}
```

Mengizinkan orang yang Anda ajak berbagi dasbor melihat widget tabel log

Saat Anda berbagi dasbor, secara default widget Wawasan CloudWatch Log yang ada di dasbor tidak terlihat oleh orang-orang yang berbagi dasbor dengan Anda. Ini memengaruhi widget Wawasan CloudWatch Log yang ada sekarang dan semua yang ditambahkan ke dasbor setelah Anda membagikannya.

Jika Anda ingin orang-orang ini dapat melihat widget CloudWatch Log, Anda harus menambahkan izin ke peran IAM untuk berbagi dasbor.

Untuk memungkinkan orang yang berbagi dasbor dengan Anda melihat widget CloudWatch Log

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor.
3. Pilih nama dasbor yang Anda bagikan.
4. Pilih Tindakan, Bagikan dasbor.
5. Di Sumber Daya, pilih Peran IAM.
6. Di konsol IAM, pilih kebijakan yang ditampilkan.
7. Pilih Edit kebijakan dan tambahkan pernyataan berikut. Di dalam pernyataan baru ini, kami menyarankan Anda untuk menentukan ARN hanya dari grup log yang ingin Anda bagikan. Lihat contoh berikut ini.

```
{
            "Effect": "Allow",
            "Action": [
                "logs:FilterLogEvents",
                "logs:StartQuery",
                "logs:StopQuery",
                "logs:GetLogRecord",
                "logs:DescribeLogGroups"
            ],
```

```
            "Resource": [
                "SharedLogGroup1ARN",
                "SharedLogGroup2ARN"
            ]
        },
```

8. Pilih Simpan Perubahan.

Jika kebijakan IAM Anda yang digunakan untuk berbagi dasbor sudah mencakup lima izin tersebut dengan * sebagai sumber daya, maka kami sangat menyarankan Anda untuk mengubah kebijakan dan menentukan hanya ARN grup log yang Anda ingin bagikan. Sebagai contoh, jika bagian `Resource` Anda untuk izin-izin ini adalah sebagai berikut:

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

Ubah kebijakan itu sehingga menentukan hanya ARN grup log yang Anda ingin bagikan, seperti dalam contoh berikut:

```
"Resource": [
  "SharedLogGroup1ARN",
  "SharedLogGroup2ARN"
]
```

Mengizinkan orang yang Anda ajak berbagi untuk melihat widget kustom

Saat Anda berbagi dasbor, secara default widget kustom yang ada di dasbor tidak akan terlihat oleh orang yang Anda ajak berbagi dasbor. Hal ini memengaruhi widget kustom yang ada sekarang dan setiap widget yang ditambahkan ke dasbor setelah Anda berbagi dasbor tersebut.

Jika Anda ingin orang tersebut dapat melihat widget kustom, maka Anda harus menambahkan izin ke peran IAM yang digunakan untuk berbagi dasbor.

Cara memungkinkan orang yang Anda berbagi dasbor dengannya untuk melihat widget kustom

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor.
3. Pilih nama dasbor yang Anda bagikan.
4. Pilih Tindakan, Bagikan dasbor.
5. Di Sumber Daya, pilih Peran IAM.

6. Di konsol IAM, pilih kebijakan yang ditampilkan.
7. Pilih Edit kebijakan dan tambahkan pernyataan berikut. Dalam pernyataan baru ini, kami menyarankan Anda untuk menentukan ARN hanya dari fungsi Lambda yang ingin Anda bagikan. Lihat contoh berikut ini.

```
{
  "Sid": "Invoke",
  "Effect": "Allow",
  "Action": [
      "lambda:InvokeFunction"
  ],
  "Resource": [
    "LambdaFunction1ARN",
    "LambdaFunction2ARN"
  ]
 }
```

8. Pilih Simpan Perubahan.

Jika kebijakan IAM Anda yang digunakan untuk berbagi dasbor sudah mencakup izin tersebut * sebagai sumber daya, maka kami sangat menyarankan Anda untuk mengubah kebijakan itu dan menentukan hanya ARN fungsi Lambda yang ingin Anda bagikan. Sebagai contoh, jika bagian `Resource` Anda untuk izin-izin ini adalah sebagai berikut:

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

Mengubah kebijakan untuk menentukan hanya ARN widget kustom yang ingin Anda bagikan saja, seperti dalam contoh berikut:

```
"Resource": [
  "LambdaFunction1ARN",
  "LambdaFunction2ARN"
]
```

Menggunakan data langsung

Anda dapat memilih widget metrik Anda akan menampilkan data langsung. Data langsung adalah data yang diterbitkan dalam menit terakhir yang belum dikumpulkan sepenuhnya.

- Jika data langsung mati, maka hanya titik data yang memiliki periode gabungan setidaknya satu menit di masa lalu yang akan ditampilkan. Sebagai contoh, ketika Anda menggunakan periode 5 menit, maka titik data untuk 12:35 akan digabungkan dari 12:35 hingga 12:40, dan akan ditampilkan pada 12:41.
- Jika data langsung hidup, maka titik data terbaru akan ditampilkan segera setelah data apa pun diterbitkan dalam rentang penggabungan terkait. Setiap kali Anda menyegarkan tampilan, titik data terbaru dapat berubah ketika data baru dalam periode pengumpulan tersebut diterbitkan. Jika Anda menggunakan statistik kumulatif seperti Jumlah atau Hitungan Sampel, menggunakan data langsung akan dapat menyebabkan penurunan di akhir grafik Anda.

Anda dapat memilih untuk mengaktifkan data langsung untuk seluruh dasbor, atau untuk widget individu yang ada di dasbor.

Cara memilih apakah Anda akan menggunakan data langsung di seluruh dasbor Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Untuk mengaktifkan atau menonaktifkan data langsung secara permanen untuk semua widget yang ada di dasbor, lakukan hal berikut:
   a. Pilih Tindakan, Pengaturan, Data langsung pembaruan massal.
   b. Pilih Data Langsung aktif atau Data Langsung nonaktif, dan pilih Tetapkan.
4. Untuk mengesampingkan pengaturan data langsung dari setiap widget untuk sementara waktu, pilih Tindakan. Kemudian, di Mengesampingkan, yang ada di samping Data langsung, lakukan salah satu langkah berikut:
   - Pilih Aktif untuk mengaktifkan data langsung untuk semua widget untuk sementara waktu.
   - Pilih Nonaktif untuk menonaktifkan data langsung untuk semua widget untuk sementara waktu.
   - Pilih Jangan diganti untuk tetap membiarkan pengaturan data langsung setiap widget pada pengaturan saat ini.

Untuk memilih apakah Anda akan menggunakan data langsung di sebuah widget tunggal

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Pilih widget, kemudian pilih Tindakan, Edit.

4. Pilih tab Pilihan grafik.
5. Pilih atau hapus centang di Data Langsung.

Menampilkan dasbor animasi

Anda dapat melihat dasbor animasi yang memutar ulang data CloudWatch metrik yang diambil dari waktu ke waktu. Hal ini akan dapat membantu Anda melihat tren, membuat presentasi, atau menganalisis masalah setelah masalah-masalah itu terjadi.

Widget animasi yang ada di dasbor mencakup widget garis, widget area bertumpuk, widget angka, dan widget eksplorer metrik. Grafik lingkaran, diagram batang, widget teks, dan widget log ditampilkan di dasbor tetapi tidak dianimasikan.

Cara melihat dasbor animasi

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor.
3. Pilih nama dasbor.
4. Pilih Tindakan, Pemutaran ulang dasbor.
5. (Opsional) Secara bawaan, ketika Anda memulai animasi, animasi yang muncul adalah jendela geser. Jika Anda ingin animasi muncul sebagai point-by-point animasi, pilih ikon kaca pembesar saat animasi dijeda dan setel ulang zoom.
6. Untuk memulai animasi, pilih tombol Play. Anda juga dapat memilih tombol mundur dan maju untuk pindah ke titik waktu lainnya.
7. (Opsional) Untuk mengubah jendela waktu animasi, pilih kalender dan pilih periode waktunya.
8. Untuk mengubah kecepatan animasi, pilih Kecepatan otomatis, kemudian pilih kecepatan baru.
9. Setelah Anda selesai, pilih Keluar dari animasi.

Tambahkan CloudWatch dasbor ke daftar favorit Anda

Di CloudWatch konsol, Anda dapat menambahkan dasbor, alarm, dan grup log ke daftar favorit. Anda dapat mengakses daftar favorit ini dari panel navigasi. Prosedur berikut akan menjelaskan cara menambahkan sebuah dasbor ke dalam daftar favorit.

Cara menambahkan sebuah dasbor ke dalam daftar favorit

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor.
3. Dari daftar dasbor, pilih simbol bintang yang ada di samping nama dasbor yang ingin Anda masukkan ke dalam daftar favorit.
   - (Opsional) Anda juga dapat memasukkan dasbor ke dalam daftar favorit dengan memilih dasbor dari daftar dan memilih simbol bintang yang ada di sebelah nama dasbor.
4. Untuk mengakses daftar favorit, pilih Favorit dan terbaru yang ada di panel navigasi. Menunya berisi dua kolom. Satu kolom berisi dasbor, alarm, dan grup log favorit Anda, dan kolom lainnya berisi dasbor, alarm, dan grup log yang baru saja Anda kunjungi.

> **Tip**
>
> Anda dapat membuat dasbor favorit, serta alarm dan grup log, dari menu Favorit dan terbaru di panel navigasi CloudWatch konsol. Di kolom Baru dikunjungi, arahkan kursor ke dasbor yang ingin Anda masukkan ke dalam daftar favorit, dan pilih simbol bintang yang ada di sebelahnya.

Ubah pengaturan penggantian periode atau interval penyegaran untuk dasbor CloudWatch

Anda dapat menentukan bagaimana pengaturan periode grafik yang ditambahkan ke dasbor ini dipertahankan atau dimodifikasi.

Ketika periode otomatis atau rentang waktu yang dipertahankan diterapkan ke widget, maka rentang waktu keseluruhan grafik dapat memengaruhi periode yang telah Anda tetapkan.

- Jika rentang waktunya satu hari atau kurang, maka pengaturan periode tidak akan berubah.
- Jika rentang waktunya antara satu hari dan tiga hari, maka periode yang diatur di bawah lima menit akan diubah menjadi 5 menit.
- Jika rentang waktunya lebih dari tiga hari, maka periode yang diatur di bawah satu jam akan diubah menjadi satu jam.

Langkah-langkah berikut akan menjelaskan cara menggunakan konsol untuk mengubah opsi penggantian periode. Anda juga dapat mengubahnya dengan menggunakan bidang `period0verride` dalam struktur JSON dasbor. Untuk informasi selengkapnya, silakan lihat Struktur Keseluruhan Tubuh Dasbor.

Cara mengubah pilihan penggantian periode

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pilih Tindakan.
3. Di Penggantian periode, pilih salah satu dari berikut ini:

   - Pilih Otomatis untuk membuat periode metrik di setiap grafik secara otomatis menyesuaikan dengan rentang waktu dasbor.
   - Pilih Jangan diganti untuk memastikan bahwa pengaturan periode dari setiap grafik selalu dipatuhi.
   - Pilih salah satu pilihan lain untuk membuat grafik yang ditambahkan ke dasbor selalu menyesuaikan dengan waktu yang dipilih sebagai pengaturan periode.

   Penggantian periode selalu kembali ke Otomatis ketika dasbor ditutup atau browser disegarkan kembali. Pengaturan-pengaturan berbeda untuk Penggantian periode tidak dapat disimpan.

Anda dapat mengubah seberapa sering data di CloudWatch dasbor Anda di-refresh.

Cara mengubah interval penyegaran kembali dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Di menu Pilihan segarkan kembali (sudut kanan atas), pilih 10 Detik, 1 Menit, 2 Menit, 5 Menit, atau 15 Menit.

Ubah rentang waktu atau format zona waktu CloudWatch dasbor

Anda dapat mengubah rentang waktu untuk menampilkan data dasbor dalam hitungan menit, jam, hari, atau minggu. Anda juga dapat mengubah format zona waktu untuk menampilkan data dasbor dengan menggunakan UTC atau waktu lokal. Waktu lokal adalah zona waktu yang ditentukan dalam sistem operasi yang digunakan oleh komputer Anda.

> **Note**
>
> Jika Anda membuat sebuah dasbor dengan grafik yang berisi 100 metrik resolusi tinggi atau lebih, maka kami menyarankan Anda untuk tidak mengatur rentang waktunya lebih dari 1 jam. Untuk informasi selengkapnya, lihat Metrik resolusi tinggi.

> **Note**
>
> Jika rentang waktu dasbor lebih pendek dari periode yang digunakan untuk widget yang ada di dasbor, maka hal-hal berikut akan terjadi:
>
> - Widget akan dimodifikasi untuk menampilkan jumlah data yang sesuai satu periode lengkap untuk widget itu, meskipun ini lebih panjang dari rentang waktu dasbor. Hal ini memastikan bahwa setidaknya ada satu titik data pada grafik.
> - Waktu mulai periode untuk titik data ini akan disesuaikan secara mundur untuk memastikan bahwa setidaknya satu titik data dapat ditampilkan.

New console

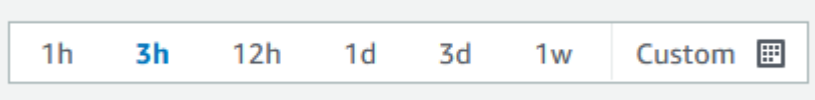
Cara mengubah rentang waktu dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Dari layar dasbor, Anda bisa melakukan salah satu hal berikut:
   - Di bagian atas dasbor, pilih salah satu rentang waktu yang telah ditentukan sebelumnya. Rentang waktunya berkisar dari 1 jam hingga 1 minggu (1 jam, 3 jam, 12 jam, 1 hari, atau 1 minggu).
   - Atau, Anda dapat memilih salah satu pilihan rentang waktu kustom berikut:
     - Pilih Kustom, kemudian pilih tab Relatif. Pilih rentang waktu dari 1 menit hingga 15 bulan.
     - Pilih Kustom, kemudian pilih tab Absolut. Gunakan bidang kalender atau teks untuk menentukan rentang waktu Anda.

> **Tip**
>
> Jika periode agregasi diatur ke Otomatis saat Anda mengubah rentang waktu grafik, CloudWatch mungkin mengubah periode. Untuk mengatur periode secara manual, pilih menu geser-turun Tindakan, kemudian pilih Penggantian periode.

Cara mengubah format zona waktu dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Di bagian atas dasbor, pilih Kustom.



4. Di sudut kanan atas kotak yang muncul, pilih UTC atau Waktu lokal dari menu geser-turun.
5. Pilih Terapkan.

Old console

Cara mengubah rentang waktu dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Dari layar dasbor, Anda bisa melakukan salah satu hal berikut:
   - Di bagian atas dasbor, pilih salah satu rentang waktu yang telah ditentukan sebelumnya. Rentang waktunya berkisar dari 1 jam hingga 1 minggu (1 jam, 3 jam, 12 jam, 1 hari, 3 hari, atau 1 minggu).
   - Atau, Anda dapat memilih salah satu pilihan rentang waktu kustom berikut:
     - Pilih menu geser-turun kustom, kemudian pilih tab Relatif. Pilih salah satu rentang yang telah ditetapkan sebelumnya, yang berkisar dari 1 menit hingga 15 bulan.
     - Pilih menu geser-turun kustom, kemudian pilih tab Absolut. Gunakan bidang kalender atau teks untuk menentukan rentang waktu Anda.

> **Tip**
>
> Jika periode agregasi diatur ke Otomatis saat Anda mengubah rentang waktu grafik, CloudWatch mungkin mengubah periode. Untuk mengatur periode secara manual, pilih menu geser-turun Tindakan, kemudian pilih Penggantian periode.

Cara mengubah format zona waktu dasbor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor, dan kemudian pilih sebuah dasbor.
3. Di sudut kanan atas layar dasbor, pilih menu geser-turun Kustom.
4. Di sudut kanan atas kotak yang muncul, pilih UTC atau Zona waktu lokal dari menu geser-turun.

Gunakan CloudWatch metrik Amazon

Metrik adalah data tentang performa sistem Anda. Secara bawaan, banyak layanan menyediakan metrik gratis untuk sumber daya (seperti instans Amazon EC2, volume Amazon EBS, dan instans Amazon RDS DB). Anda juga dapat mengaktifkan pemantauan terperinci untuk beberapa sumber daya, seperti instans Amazon EC2, atau menerbitkan metrik aplikasi Anda sendiri. Amazon CloudWatch dapat memuat semua metrik di akun Anda (baik metrik AWS sumber daya maupun metrik aplikasi yang Anda berikan) untuk penelusuran, grafik, dan alarm.

Data metrik disimpan selama 15 bulan, memungkinkan Anda untuk melihat up-to-the-minute data dan data historis.

Untuk membuat grafik metrik di konsol, Anda dapat menggunakan CloudWatch Metrics Insights, mesin kueri SQL berkinerja tinggi yang dapat Anda gunakan untuk mengidentifikasi tren dan pola dalam semua metrik Anda secara real time.

Daftar Isi

Pemantauan dasar dan pemantauan terperinci

CloudWatch menyediakan dua kategori pemantauan: pemantauan dasar dan pemantauan terperinci.

Banyak AWS layanan menawarkan pemantauan dasar dengan menerbitkan seperangkat metrik default tanpa biaya kepada pelanggan. CloudWatch Secara default, ketika Anda mulai menggunakan

salah satu dari ini Layanan AWS, pemantauan dasar diaktifkan secara otomatis. Untuk daftar layanan yang menawarkan pemantauan dasar, silakan lihat AWS layanan yang mempublikasikan CloudWatch metrik.

Pemantauan terperinci hanya ditawarkan oleh beberapa layanan. Itu juga menimbulkan biaya. Untuk menggunakannya untuk suatu AWS layanan, Anda harus memilih untuk mengaktifkannya. Untuk informasi selengkapnya tentang harga, lihat CloudWatch harga Amazon.

Opsi pemantauan terperinci berbeda berdasarkan layanan yang menawarkannya. Misalnya, pemantauan terperinci Amazon EC2 memberikan metrik yang lebih sering, yang diterbitkan pada interval satu menit, alih-alih interval lima menit yang digunakan dalam pemantauan dasar Amazon EC2. Pemantauan terperinci untuk Amazon S3 dan Amazon Managed Streaming untuk Apache Kafka berarti metrik yang lebih halus.

Dalam AWS layanan yang berbeda, pemantauan terperinci juga memiliki nama yang berbeda. Misalnya, di Amazon EC2 disebut pemantauan terperinci, di AWS Elastic Beanstalk dalamnya disebut pemantauan yang ditingkatkan, dan di Amazon S3 disebut metrik permintaan.

Menggunakan pemantauan terperinci untuk Amazon EC2 akan membantu Anda dalam mengelola sumber daya Amazon EC2 dengan lebih baik, sehingga Anda dapat menemukan tren dan mengambil tindakan dengan lebih cepat. Untuk Amazon S3, metrik permintaan tersedia dengan interval satu menit untuk membantu Anda mengidentifikasi dan menindaklanjuti masalah operasional dengan cepat. Di Amazon MSK, saat mengaktifkan pemantauan level `PER_BROKER`, `PER_TOPIC_PER_BROKER`, atau `PER_TOPIC_PER_PARTITION`, Anda mendapatkan metrik tambahan yang memberikan visibilitas lebih.

Tabel berikut mencantumkan layanan-layanan yang menawarkan pemantauan terperinci. Tabel ini juga mencakup tautan ke dokumentasi untuk layanan yang menjelaskan lebih lanjut tentang pemantauan terperinci dan memberikan instruksi tentang cara mengaktifkannya.

Layanan	Dokumentasi
Amazon API Gateway	Dimensi untuk metrik API Gateway
Amazon CloudFront	Melihat metrik CloudFron

Layanan	Dokumentasi
	t distribusi tambahan
Amazon EC2	Aktifkan atau nonaktifkan pemantauan terperinci untuk instans Anda
Elastic Beanstalk	Pelaporan dan pemantauan kondisi yang ditingkatkan
Amazon Kinesis Data Streams	Metrik Tingkat Serpihan yang Ditingkatkan
Amazon MSK	Metrik MSK Amazon untuk Pemantaua n dengan CloudWatch
Amazon S3	Metrik permintaa n Amazon S3 di CloudWatch
Amazon SES	Kumpulkan metrik pemantaua n CloudWatc h terperinci menggunakan penerbitan acara Amazon SES.

Selain itu, CloudWatch menawarkan solusi out-of-the-box pemantauan dengan metrik yang lebih rinci dan dasbor yang telah dibuat sebelumnya untuk beberapa AWS layanan, seperti yang ditunjukkan pada tabel berikut.

Layanan	Dokumentasi fitur
Lambda	Wawasan Lambda
Amazon ECS	Wawasan Kontainer untuk Amazon ECS
Amazon EKS	Wawasan Kontainer untuk Amazon EKS dan Kubernetes

Kueri metrik Anda dengan Wawasan CloudWatch Metrik

CloudWatch Metrics Insights adalah mesin kueri SQL berkinerja tinggi yang kuat yang dapat Anda gunakan untuk menanyakan metrik Anda dalam skala besar. Anda dapat mengidentifikasi tren dan pola dalam semua CloudWatch metrik Anda secara real time.

Anda juga dapat menyetel alarm pada setiap kueri Wawasan Metrik yang menampilkan satu deret waktu tunggal. Ini dapat sangat berguna untuk membuat alarm yang mengamati metrik agregat di seluruh armada infrastruktur atau aplikasi Anda. Buat alarm sekali, dan secara dinamis menyesuaikan ketika sebagai sumber daya ditambahkan atau dihapus dari armada.

Anda dapat melakukan kueri Wawasan CloudWatch Metrik di konsol dengan editor kueri CloudWatch Metrics Insights. Anda juga dapat melakukan kueri Wawasan CloudWatch Metrik dengan AWS CLI atau AWS SDK dengan menjalankan atau. `GetMetricData` `PutDashboard` Tidak ada biaya untuk kueri yang Anda jalankan dengan editor kueri CloudWatch Metrics Insights. Untuk informasi selengkapnya tentang CloudWatch harga, lihat CloudWatch Harga Amazon.

Dengan editor kueri CloudWatch Metrics Insights, Anda dapat memilih dari berbagai kueri sampel bawaan dan juga membuat kueri Anda sendiri. Saat membuat kueri Anda, Anda dapat menggunakan

tampilan pembangun untuk menelusuri metrik dan dimensi yang ada. Atau, gunakan tampilan editor untuk menulis kueri secara manual.

Anda juga dapat menggunakan bahasa alami untuk membuat kueri Wawasan CloudWatch Metrik. Untuk melakukan hal itu, ajukan pertanyaan atau jelaskan data yang Anda cari. Kemampuan berbantuan AI ini menghasilkan kueri berdasarkan prompt Anda dan memberikan line-by-line penjelasan tentang cara kerja kueri. Untuk informasi selengkapnya, lihat Menggunakan bahasa alami untuk membuat dan memperbarui kueri Wawasan CloudWatch Metrik.

Dengan Metrics Insights, Anda dapat menjalankan kueri dalam skala besar. Dengan klausa GROUP BY, Anda dapat mengelompokkan metrik secara waktu nyata ke dalam deret waktu terpisah per nilai dimensi tertentu. Karena kueri Wawasan Metrik menyertakan kemampuan ORDER BY, Anda dapat menggunakan Wawasan Metrik untuk membuat kueri tipe "N Teratas". Misalnya, kueri tipe "N Teratas" dapat memindai jutaan metrik di akun Anda dan mengembalikan 10 instans yang paling banyak menggunakan CPU. Ini dapat membantu Anda menentukan dan memperbaiki masalah latensi dalam aplikasi Anda.

Topik

Bangun kueri Anda

Anda dapat menjalankan kueri CloudWatch Metrics Insights menggunakan CloudWatch konsol, file AWS CLI, atau SDK AWS . Kueri yang dijalankan di konsol tidak dikenakan biaya. Untuk informasi selengkapnya tentang CloudWatch harga, lihat CloudWatch Harga Amazon.

Untuk informasi selengkapnya tentang penggunaan AWS SDK untuk melakukan kueri Wawasan Metrik, lihat. GetMetricData

Untuk menjalankan kueri menggunakan CloudWatch konsol, ikuti langkah-langkah berikut:

Untuk menanyakan metrik Anda menggunakan Metrics Insights

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik, Semua metrik.
3. Pilih tab Kueri.
4. (Opsional) Untuk menjalankan kueri sampel yang telah dibuat sebelumnya, pilih Tambahkan kueri dan pilih kueri yang akan dijalankan. Jika puas dengan kueri ini, Anda dapat melewati sisa prosedur ini. Atau, Anda dapat memilih Editor untuk mengedit kueri sampel dan kemudian memilih Jalankan untuk menjalankan kueri yang dimodifikasi.
5. Untuk membuat kueri Anda sendiri, Anda dapat menggunakan tampilan Builder, tampilan Editor, dan juga menggunakan kombinasi keduanya. Anda dapat beralih di antara dua tampilan tersebut kapan saja dan melihat pekerjaan Anda yang sedang berlangsung di kedua tampilan.

   Pada tampilan Builder, Anda dapat menelusuri dan memilih namespace metrik, nama metrik, filter, grup, dan opsi urutan. Untuk masing-masing opsi ini, pembuat kueri menawarkan sebuah daftar kemungkinan pilihan dari lingkungan Anda untuk dipilih.

   Pada tampilan Editor, Anda dapat mulai menulis kueri Anda. Saat Anda mengetik, editor menawarkan saran berdasarkan karakter yang telah Anda ketik sejauh ini.
6. Saat Anda puas dengan kueri Anda, pilih Jalankan.
7. (Opsional) Cara lain untuk mengedit kueri yang telah Anda gambarkan adalah dengan memilih tab Metrik bergrafik dan memilih ikon edit di sebelah rumus kueri di kolom Detail.
8. (Opsional) Untuk menghapus kueri dari grafik, pilih Metrik bergrafik dan pilih ikon X di sisi kanan baris yang menampilkan kueri Anda.

Komponen dan sintaks kueri Metrics Insights

CloudWatch Sintaks Wawasan Metrik adalah sebagai berikut.

```
SELECT FUNCTION(metricName)
FROM namespace | SCHEMA(...)
```

```
[ WHERE labelKey OPERATOR labelValue [AND ... ] ]
[ GROUP BY labelKey [ , ... ] ]
[ ORDER BY FUNCTION() [ DESC | ASC ] ]
[ LIMIT number ]
```

Klausa yang mungkin dalam kueri Wawasan Metrik adalah sebagai berikut. Tidak ada kata kunci yang peka huruf besar/kecil, tetapi pengidentifikasi seperti nama metrik, namespace, dan dimensi peka huruf besar/kecil.

SELECT

Wajib. Menentukan fungsi yang akan digunakan untuk agregat pengamatan di setiap bucket waktu (ditentukan oleh periode yang disediakan). Juga menentukan nama metrik untuk kueri.

Nilai yang benar untuk FUNCTION adalah `AVG`, `COUNT`, `MAX`, `MIN`, dan `SUM`.

- `AVG` menghitung rata-rata observasi yang dicocokkan berdasarkan kueri.
- `COUNT` mengembalikan jumlah observasi yang dicocokkan berdasarkan kueri.
- `MAX` mengembalikan nilai maksimal observasi yang dicocokkan berdasarkan kueri.
- `MIN` mengembalikan nilai minimal observasi yang dicocokkan berdasarkan kueri.
- `SUM` menghitung jumlah observasi yang dicocokkan berdasarkan kueri.

FROM

Wajib. Menentukan sumber metrik. Anda dapat menentukan namespace metrik yang berisi metrik yang akan ditanyakan, atau fungsi tabel SCHEMA. Contoh namespace metrik meliputi`"AWS/EC2"`, `"AWS/Lambda"`, dan namespace metrik yang telah Anda buat untuk metrik kustom Anda.

Namespace metrik yang menyertakan / atau karakter lain yang bukan huruf, angka, atau garis bawah harus dikelilingi dengan tanda kutip ganda. Untuk informasi selengkapnya, lihat Apa yang membutuhkan tanda kutip atau karakter pelarian?.

SCHEMA

Fungsi tabel opsional yang dapat digunakan dalam klausa FROM. Gunakan SCHEMA untuk mencatat hasil kueri hanya ke metrik yang sama persis dengan daftar dimensi, atau metrik yang tidak memiliki dimensi.

Jika Anda menggunakan klausa SCHEMA, ini setidaknya harus berisi satu argumen, dan argumen pertama ini harus berupa namespace metrik yang ditanyakan. Jika Anda menentukan

SCHEMA hanya dengan argumen namespace ini, hasilnya akan tercakup hanya ke metrik yang tidak memiliki dimensi apa pun.

Jika Anda menentukan SCHEMA dengan argumen tambahan, argumen tambahan setelah argumen namespace harus berupa kunci label. Kunci label harus berupa nama dimensi. Jika Anda menentukan satu atau beberapa kunci label ini, hasilnya akan tercakup hanya ke metrik yang memiliki kumpulan dimensi yang tepat. Urutan kunci label ini tidak masalah.

Sebagai contoh:

- SELECT AVG (CPUUtilization) FROM "AWS/EC2" mencocokkan semua metrik `CPUUtilization` di namespace, `AWS/EC2`apa pun dimensinya, dan mengembalikan satu deret waktu gabungan.
- SELECT AVG(CPUUtilization) FROM SCHEMA("AWS/EC2") hanya mencocokkan metrik `CPUUtilization` di namespace `AWS/EC2` yang tidak memiliki dimensi yang ditentukan.
- SELECT AVG (CPUUutilization) FROM SCHEMA (“AWS/EC2", InstanceId) hanya cocok dengan `CPUUtilization` metrik yang dilaporkan dengan tepat satu dimensi,. CloudWatch `InstanceId`
- SELECT SUM (RequestCount) FROM SCHEMA (“AWS/ApplicationElb”, LoadBalancer, AvailabilityZone) hanya cocok dengan `RequestCount` metrik yang dilaporkan CloudWatch dari `AWS/ApplicationELB` dengan tepat dua dimensi, dan. `LoadBalancer` `AvailabilityZone`

WHERE

Opsional. Memfilter hasil hanya ke metrik yang cocok dengan ekspresi yang Anda tentukan menggunakan nilai label tertentu untuk satu kunci label atau lebih. Misalnya, WHERE InstanceType = 'c3.4xlarge' memfilter hasil hanya ke tipe **`c3.4xlarge`** instance, dan WHERE! InstanceType = 'c3.4xlarge' memfilter hasil ke semua jenis instance kecuali. `c3.4xlarge`

Saat Anda menjalankan kueri dalam sebuah akun pemantauan, Anda dapat memilih `WHERE AWS.AccountId` untuk membatasi hasilnya hanya untuk akun yang Anda tentukan. Sebagai contoh, metrik kueri `WHERE AWS.AccountId=444455556666` hanya dari akun `444455556666`. Untuk membatasi kueri Anda hanya pada metrik-metrik yang ada di akun pemantauan itu sendiri, gunakan `WHERE AWS.AccountId=CURRENT_ACCOUNT_ID()`.

Nilai label harus selalu diapit dengan tanda kutip tunggal.

Operator yang didukung

Klausa WHERE mendukung operator berikut:

- Nilai label = harus cocok dengan string yang ditentukan.
- Nilai label != tidak harus cocok dengan string yang ditentukan.
- AND Kedua kondisi yang ditentukan harus benar agar cocok. Anda dapat menggunakan beberapa kata kunci AND untuk menentukan dua atau lebih kondisi.

GROUP BY

Opsional. Kelompokkan hasil kueri ke dalam beberapa deret waktu, masing-masing sesuai dengan nilai yang berbeda untuk kunci atau beberapa kunci label yang ditentukan. Misalnya, menggunakan `GROUP BY InstanceId` mengembalikan deret waktu yang berbeda untuk setiap nilai `InstanceId`. Menggunakan `GROUP BY ServiceName, Operation` membuat satu deret waktu yang berbeda untuk setiap kemungkinan kombinasi dari nilai-nilai `ServiceName` dan`Operation`.

Dengan klausa GROUP BY, secara default, hasilnya diurutkan dalam urutan naik menurut abjad, menggunakan urutan label yang ditentukan dalam klausa GROUP BY. Untuk mengubah urutan hasil, tambahkan klausa ORDER BY ke kueri Anda.

Saat menjalankan kueri dalam sebuah akun pemantauan, Anda dapat menggunakan `GROUP BY AWS.AccountId` untuk mengelompokkan hasilnya berdasarkan akun asal mereka.

> **Note**
>
> Jika beberapa metrik yang cocok tidak menyertakan kunci label tertentu yang ditentukan dalam klausa GROUP BY, grup kosong bernama `Other` dikembalikan. Sebagai contoh, jika Anda menentukan `GROUP BY ServiceName, Operation` dan beberapa metrik yang dikembalikan tidak menyertakan `ServiceName` sebagai dimensi, maka metrik tersebut ditampilkan memiliki `Other` sebagai nilai untuk `ServiceName`.

ORDER BY

Opsional. Menentukan urutan yang akan digunakan untuk deret waktu yang dikembalikan, jika kueri mengembalikan lebih dari satu deret waktu. Urutan didasarkan pada nilai yang ditemukan oleh FUNCTION yang Anda tentukan dalam klausa ORDER BY. FUNCTION digunakan untuk menghitung nilai skalar tunggal dari setiap deret waktu yang dikembalikan, dan nilai itu digunakan untuk menentukan urutan.

Anda juga menentukan apakah akan menggunakan urutan ASC naik atau DESC turun. Jika Anda menghilangkan ini, defaultnya adalah ASC naik.

Misalnya, menambahkan klausa `ORDER BY MAX() DESC` memerintahkan hasil dengan titik data maksimum yang diamati dalam rentang waktu, dalam urutan menurun: artinya deret waktu yang memiliki titik data maksimum tertinggi dikembalikan terlebih dahulu.

Fungsi yang valid untuk digunakan dalam klausa ORDER BY adalah `AVG()`, `COUNT()`, `MAX()`, `MIN()`, dan `SUM()`.

Jika Anda menggunakan klausa ORDER BY dengan klausa LIMIT, kueri yang dihasilkan adalah kueri "N Teratas". ORDER BY juga berguna untuk kueri yang mungkin mengembalikan sejumlah besar metrik, karena setiap kueri dapat mengembalikan tidak lebih dari 500 deret waktu. Jika kueri cocok dengan lebih dari 500 deret waktu, dan Anda menggunakan klausa ORDER BY, deret waktu diurutkan dan kemudian 500 deret waktu yang muncul pertama dalam urutan pengurutan merupakan deret waktu yang dikembalikan.

LIMIT

Opsional. Membatasi jumlah deret waktu yang dikembalikan oleh kueri ke nilai yang Anda tentukan. Nilai maksimum yang dapat Anda tentukan adalah 500, dan kueri yang tidak menentukan LIMIT juga dapat mengembalikan tidak lebih dari 500 deret waktu.

Menggunakan klausa LIMIT dengan klausa ORDER BY memberi Anda kueri "N Teratas".

Apa yang membutuhkan tanda kutip atau karakter pelarian?

Dalam kueri, nilai label harus selalu dikurung dengan tanda kutip tunggal. Misalnya, PILIH MAX (CPUUutilization) DARI “AWS/EC2" WHERE = ". AutoScalingGroupName my-production-fleet

Namespace metrik, nama metrik, dan kunci label yang berisi karakter selain huruf, angka, dan garis bawah (_) harus dikurung dengan tanda kutip ganda. Misalnya, SELECT MAX("My.Metric").

Jika salah satu dari ini berisi tanda kutip ganda atau tanda kutip tunggal itu sendiri (seperti `Bytes"Input"`), Anda harus keluar dari setiap tanda kutip dengan garis miring terbalik, seperti pada SELECT AVG("Bytes\"Input\"").

Jika namespace metrik, nama metrik, atau kunci label, berisi kata yang merupakan kata kunci cadangan dalam Metrics Insights, ini juga harus diapit dalam tanda kutip ganda. Sebagai contoh, jika Anda memiliki metrik yang disebut `LIMIT`, Anda akan menggunakan `SELECT AVG("LIMIT")`. Ini juga benar untuk mengapit namespace, nama metrik, atau label dalam tanda kutip ganda bahkan jika itu tidak menyertakan kata kunci yang dicadangkan.

Untuk daftar lengkap kata kunci yang disimpan, silakan lihat Kata kunci terpesan.

Bangun kueri kaya langkah demi langkah

Bagian ini menggambarkan membangun contoh lengkap yang menggunakan semua kemungkinan klausa, langkah demi langkah.

Kita mulai dengan kueri berikut, yang menggabungkan semua metrik `RequestCount` Penyeimbang Beban Aplikasi yang dikumpulkan dengan dimensi `LoadBalancer` dan `AvailabilityZone`.

```
SELECT SUM(RequestCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer, AvailabilityZone)
```

Sekarang, jika kita ingin melihat metrik hanya dari penyeimbang beban tertentu, kita dapat menambahkan klausa WHERE untuk membatasi metrik yang dikembalikan hanya ke metrik di mana nilai dimensi `LoadBalancer` adalah `app/load-balancer-1`.

```
SELECT SUM(RequestCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer, AvailabilityZone)
WHERE LoadBalancer = 'app/load-balancer-1'
```

Kueri sebelumnya menggabungkan metrik `RequestCount` dari semua Availability Zone untuk penyeimbang beban ini menjadi satu deret waktu. Jika ingin melihat deret waktu yang berbeda untuk setiap Availability Zone, kita dapat menambahkan klausa GROUP BY.

```
SELECT SUM(RequestCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer, AvailabilityZone)
WHERE LoadBalancer = 'app/load-balancer-1'
GROUP BY AvailabilityZone
```

Berikutnya, kita mungkin ingin memesan hasil ini untuk melihat nilai tertinggi terlebih dahulu. Klausa ORDER BY berikut memerintahkan deret waktu dalam urutan menurun, dengan nilai maksimum yang dilaporkan oleh setiap deret waktu selama rentang waktu kueri:

```
SELECT SUM(RequestCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer, AvailabilityZone)
WHERE LoadBalancer = 'app/load-balancer-1'
GROUP BY AvailabilityZone
ORDER BY MAX() DESC
```

Akhirnya, jika kita tertarik terutama pada jenis kueri "N Teratas", kita dapat menggunakan klausa LIMIT. Contoh terakhir ini membatasi hasil deret waktu hanya dengan lima nilai `MAX` tertinggi.

```
SELECT SUM(RequestCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer, AvailabilityZone)
WHERE LoadBalancer = 'app/load-balancer-1'
GROUP BY AvailabilityZone
ORDER BY MAX() DESC
LIMIT 5
```

Contoh kueri lintas akun

Contoh-contoh ini valid ketika dijalankan di akun yang disiapkan sebagai akun pemantauan dalam observabilitas CloudWatch lintas akun.

Contoh berikut ini mencari semua instans Amazon EC2 yang ada di akun sumber 123456789012 dan mengembalikan rata-ratanya.

```
SELECT AVG(CpuUtilization)
FROM "AWS/EC2"
WHERE AWS.AccountId ='123456789012'
```

Contoh berikut menjalankan kueri terhadap metrik `CPUUtilization` yang ada di `AWS/EC2` di semua akun sumber terkait, dan mengelompokkan hasilnya berdasarkan ID akun dan tipe instans.

```
SELECT AVG(CpuUtilization)
FROM "AWS/EC2"
GROUP BY AWS.AccountId, InstanceType
```

Contoh berikut menjalankan kueri terhadap `CPUUtilization` di akun pemantauan itu sendiri.

```
SELECT AVG(CpuUtilization)
FROM "AWS/EC2"
WHERE AWS.AccountId = CURRENT_ACCOUNT_ID()
```

Kata kunci terpesan

Berikut ini adalah kata kunci yang dicadangkan dalam Wawasan CloudWatch Metrik. Jika salah satu dari kata-kata ini berada di namespace, nama metrik, atau kunci label dalam kueri, Anda harus mengapitnya dalam tanda kutip ganda. Kata kunci yang dicadangkan tidak peka huruf besar/kecil.

```
"ABORT" "ABORTSESSION" "ABS" "ABSOLUTE" "ACCESS" "ACCESSIBLE" "ACCESS_LOCK" "ACCOUNT"
 "ACOS" "ACOSH" "ACTION" "ADD" "ADD_MONTHS"
```

```
"ADMIN" "AFTER" "AGGREGATE" "ALIAS" "ALL" "ALLOCATE" "ALLOW" "ALTER" "ALTERAND" "AMP"
 "ANALYSE" "ANALYZE" "AND" "ANSIDATE" "ANY" "ARE" "ARRAY",
"ARRAY_AGG" "ARRAY_EXISTS" "ARRAY_MAX_CARDINALITY" "AS" "ASC" "ASENSITIVE" "ASIN"
 "ASINH" "ASSERTION" "ASSOCIATE" "ASUTIME" "ASYMMETRIC" "AT",
"ATAN" "ATAN2" "ATANH" "ATOMIC" "AUDIT" "AUTHORIZATION" "AUX" "AUXILIARY" "AVE"
 "AVERAGE" "AVG" "BACKUP" "BEFORE" "BEGIN" "BEGIN_FRAME" "BEGIN_PARTITION",
"BETWEEN" "BIGINT" "BINARY" "BIT" "BLOB" "BOOLEAN" "BOTH" "BREADTH" "BREAK" "BROWSE"
 "BT" "BUFFERPOOL" "BULK" "BUT" "BY" "BYTE" "BYTEINT" "BYTES" "CALL",
"CALLED" "CAPTURE" "CARDINALITY" "CASCADE" "CASCADED" "CASE" "CASESPECIFIC" "CASE_N"
 "CAST" "CATALOG" "CCSID" "CD" "CEIL" "CEILING" "CHANGE" "CHAR",
"CHAR2HEXINT" "CHARACTER" "CHARACTERS" "CHARACTER_LENGTH" "CHARS" "CHAR_LENGTH" "CHECK"
 "CHECKPOINT" "CLASS" "CLASSIFIER" "CLOB" "CLONE" "CLOSE" "CLUSTER",
"CLUSTERED" "CM" "COALESCE" "COLLATE" "COLLATION" "COLLECT" "COLLECTION" "COLLID"
 "COLUMN" "COLUMN_VALUE" "COMMENT" "COMMIT" "COMPLETION" "COMPRESS" "COMPUTE",
"CONCAT" "CONCURRENTLY" "CONDITION" "CONNECT" "CONNECTION" "CONSTRAINT" "CONSTRAINTS"
 "CONSTRUCTOR" "CONTAINS" "CONTAINSTABLE" "CONTENT" "CONTINUE" "CONVERT",
"CONVERT_TABLE_HEADER" "COPY" "CORR" "CORRESPONDING" "COS" "COSH" "COUNT" "COVAR_POP"
 "COVAR_SAMP" "CREATE" "CROSS" "CS" "CSUM" "CT" "CUBE" "CUME_DIST",
"CURRENT" "CURRENT_CATALOG" "CURRENT_DATE" "CURRENT_DEFAULT_TRANSFORM_GROUP"
 "CURRENT_LC_CTYPE" "CURRENT_PATH" "CURRENT_ROLE" "CURRENT_ROW" "CURRENT_SCHEMA",
"CURRENT_SERVER" "CURRENT_TIME" "CURRENT_TIMESTAMP" "CURRENT_TIMEZONE"
 "CURRENT_TRANSFORM_GROUP_FOR_TYPE" "CURRENT_USER" "CURRVAL" "CURSOR" "CV" "CYCLE"
 "DATA",
"DATABASE" "DATABASES" "DATABLOCKSIZE" "DATE" "DATEFORM" "DAY" "DAYS" "DAY_HOUR"
 "DAY_MICROSECOND" "DAY_MINUTE" "DAY_SECOND" "DBCC" "DBINFO" "DEALLOCATE" "DEC",
"DECFLOAT" "DECIMAL" "DECLARE" "DEFAULT" "DEFERRABLE" "DEFERRED" "DEFINE" "DEGREES"
 "DEL" "DELAYED" "DELETE" "DENSE_RANK" "DENY" "DEPTH" "DEREF" "DESC" "DESCRIBE",
"DESCRIPTOR" "DESTROY" "DESTRUCTOR" "DETERMINISTIC" "DIAGNOSTIC" "DIAGNOSTICS"
 "DICTIONARY" "DISABLE" "DISABLED" "DISALLOW" "DISCONNECT" "DISK" "DISTINCT",
"DISTINCTROW" "DISTRIBUTED" "DIV" "DO" "DOCUMENT" "DOMAIN" "DOUBLE" "DROP" "DSSIZE"
 "DUAL" "DUMP" "DYNAMIC" "EACH" "ECHO" "EDITPROC" "ELEMENT" "ELSE" "ELSEIF",
"EMPTY" "ENABLED" "ENCLOSED" "ENCODING" "ENCRYPTION" "END" "END-EXEC" "ENDING"
 "END_FRAME" "END_PARTITION" "EQ" "EQUALS" "ERASE" "ERRLVL" "ERROR" "ERRORFILES",
"ERRORTABLES" "ESCAPE" "ESCAPED" "ET" "EVERY" "EXCEPT" "EXCEPTION" "EXCLUSIVE" "EXEC"
 "EXECUTE" "EXISTS" "EXIT" "EXP" "EXPLAIN" "EXTERNAL" "EXTRACT" "FALLBACK
"FALSE" "FASTEXPORT" "FENCED" "FETCH" "FIELDPROC" "FILE" "FILLFACTOR" "FILTER" "FINAL"
 "FIRST" "FIRST_VALUE" "FLOAT" "FLOAT4" "FLOAT8" "FLOOR"
"FOR" "FORCE" "FOREIGN" "FORMAT" "FOUND" "FRAME_ROW" "FREE" "FREESPACE" "FREETEXT"
 "FREETEXTTABLE" "FREEZE" "FROM" "FULL" "FULLTEXT" "FUNCTION"
"FUSION" "GE" "GENERAL" "GENERATED" "GET" "GIVE" "GLOBAL" "GO" "GOTO" "GRANT" "GRAPHIC"
 "GROUP" "GROUPING" "GROUPS" "GT" "HANDLER" "HASH"
"HASHAMP" "HASHBAKAMP" "HASHBUCKET" "HASHROW" "HAVING" "HELP" "HIGH_PRIORITY" "HOLD"
 "HOLDLOCK" "HOUR" "HOURS" "HOUR_MICROSECOND" "HOUR_MINUTE"
```

```
"HOUR_SECOND" "IDENTIFIED" "IDENTITY" "IDENTITYCOL" "IDENTITY_INSERT" "IF" "IGNORE"
 "ILIKE" "IMMEDIATE" "IN" "INCLUSIVE" "INCONSISTENT" "INCREMENT"
"INDEX" "INDICATOR" "INFILE" "INHERIT" "INITIAL" "INITIALIZE" "INITIALLY" "INITIATE"
 "INNER" "INOUT" "INPUT" "INS" "INSENSITIVE" "INSERT" "INSTEAD"
"INT" "INT1" "INT2" "INT3" "INT4" "INT8" "INTEGER" "INTEGERDATE" "INTERSECT"
 "INTERSECTION" "INTERVAL" "INTO" "IO_AFTER_GTIDS" "IO_BEFORE_GTIDS"
"IS" "ISNULL" "ISOBID" "ISOLATION" "ITERATE" "JAR" "JOIN" "JOURNAL" "JSON_ARRAY"
 "JSON_ARRAYAGG" "JSON_EXISTS" "JSON_OBJECT" "JSON_OBJECTAGG"
"JSON_QUERY" "JSON_TABLE" "JSON_TABLE_PRIMITIVE" "JSON_VALUE" "KEEP" "KEY" "KEYS"
 "KILL" "KURTOSIS" "LABEL" "LAG" "LANGUAGE" "LARGE" "LAST"
"LAST_VALUE" "LATERAL" "LC_CTYPE" "LE" "LEAD" "LEADING" "LEAVE" "LEFT" "LESS" "LEVEL"
 "LIKE" "LIKE_REGEX" "LIMIT" "LINEAR" "LINENO" "LINES"
"LISTAGG" "LN" "LOAD" "LOADING" "LOCAL" "LOCALE" "LOCALTIME" "LOCALTIMESTAMP" "LOCATOR"
 "LOCATORS" "LOCK" "LOCKING" "LOCKMAX" "LOCKSIZE" "LOG"
"LOG10" "LOGGING" "LOGON" "LONG" "LONGBLOB" "LONGTEXT" "LOOP" "LOWER" "LOW_PRIORITY"
 "LT" "MACRO" "MAINTAINED" "MAP" "MASTER_BIND"
"MASTER_SSL_VERIFY_SERVER_CERT" "MATCH" "MATCHES" "MATCH_NUMBER" "MATCH_RECOGNIZE"
 "MATERIALIZED" "MAVG" "MAX" "MAXEXTENTS" "MAXIMUM" "MAXVALUE"
"MCHARACTERS" "MDIFF" "MEDIUMBLOB" "MEDIUMINT" "MEDIUMTEXT" "MEMBER" "MERGE" "METHOD"
 "MICROSECOND" "MICROSECONDS" "MIDDLEINT" "MIN" "MINDEX"
"MINIMUM" "MINUS" "MINUTE" "MINUTES" "MINUTE_MICROSECOND" "MINUTE_SECOND" "MLINREG"
 "MLOAD" "MLSLABEL" "MOD" "MODE" "MODIFIES" "MODIFY"
"MODULE" "MONITOR" "MONRESOURCE" "MONSESSION" "MONTH" "MONTHS" "MSUBSTR" "MSUM"
 "MULTISET" "NAMED" "NAMES" "NATIONAL" "NATURAL" "NCHAR" "NCLOB"
"NE" "NESTED_TABLE_ID" "NEW" "NEW_TABLE" "NEXT" "NEXTVAL" "NO" "NOAUDIT" "NOCHECK"
 "NOCOMPRESS" "NONCLUSTERED" "NONE" "NORMALIZE" "NOT" "NOTNULL"
"NOWAIT" "NO_WRITE_TO_BINLOG" "NTH_VALUE" "NTILE" "NULL" "NULLIF" "NULLIFZERO" "NULLS"
 "NUMBER" "NUMERIC" "NUMPARTS" "OBID" "OBJECT" "OBJECTS"
"OCCURRENCES_REGEX" "OCTET_LENGTH" "OF" "OFF" "OFFLINE" "OFFSET" "OFFSETS" "OLD"
 "OLD_TABLE" "OMIT" "ON" "ONE" "ONLINE" "ONLY" "OPEN" "OPENDATASOURCE"
"OPENQUERY" "OPENROWSET" "OPENXML" "OPERATION" "OPTIMIZATION" "OPTIMIZE"
 "OPTIMIZER_COSTS" "OPTION" "OPTIONALLY" "OR" "ORDER" "ORDINALITY" "ORGANIZATION"
"OUT" "OUTER" "OUTFILE" "OUTPUT" "OVER" "OVERLAPS" "OVERLAY" "OVERRIDE" "PACKAGE" "PAD"
 "PADDED" "PARAMETER" "PARAMETERS" "PART" "PARTIAL" "PARTITION"
"PARTITIONED" "PARTITIONING" "PASSWORD" "PATH" "PATTERN" "PCTFREE" "PER" "PERCENT"
 "PERCENTILE" "PERCENTILE_CONT" "PERCENTILE_DISC" "PERCENT_RANK" "PERIOD" "PERM"
"PERMANENT" "PIECESIZE" "PIVOT" "PLACING" "PLAN" "PORTION" "POSITION" "POSITION_REGEX"
 "POSTFIX" "POWER" "PRECEDES" "PRECISION" "PREFIX" "PREORDER"
"PREPARE" "PRESERVE" "PREVVAL" "PRIMARY" "PRINT" "PRIOR" "PRIQTY" "PRIVATE"
 "PRIVILEGES" "PROC" "PROCEDURE" "PROFILE" "PROGRAM" "PROPORTIONAL"
"PROTECTION" "PSID" "PTF" "PUBLIC" "PURGE" "QUALIFIED" "QUALIFY" "QUANTILE" "QUERY"
 "QUERYNO" "RADIANS" "RAISERROR" "RANDOM" "RANGE" "RANGE_N" "RANK"
"RAW" "READ" "READS" "READTEXT" "READ_WRITE" "REAL" "RECONFIGURE" "RECURSIVE" "REF"
 "REFERENCES" "REFERENCING" "REFRESH" "REGEXP" "REGR_AVGX" "REGR_AVGY"
```

```
"REGR_COUNT" "REGR_INTERCEPT" "REGR_R2" "REGR_SLOPE" "REGR_SXX" "REGR_SXY" "REGR_SYY"
 "RELATIVE" "RELEASE" "RENAME" "REPEAT" "REPLACE" "REPLICATION"
"REPOVERRIDE" "REQUEST" "REQUIRE" "RESIGNAL" "RESOURCE" "RESTART" "RESTORE" "RESTRICT"
 "RESULT" "RESULT_SET_LOCATOR" "RESUME" "RET" "RETRIEVE" "RETURN"
"RETURNING" "RETURNS" "REVALIDATE" "REVERT" "REVOKE" "RIGHT" "RIGHTS" "RLIKE" "ROLE"
 "ROLLBACK" "ROLLFORWARD" "ROLLUP" "ROUND_CEILING" "ROUND_DOWN"
"ROUND_FLOOR" "ROUND_HALF_DOWN" "ROUND_HALF_EVEN" "ROUND_HALF_UP" "ROUND_UP" "ROUTINE"
 "ROW" "ROWCOUNT" "ROWGUIDCOL" "ROWID" "ROWNUM" "ROWS" "ROWSET"
"ROW_NUMBER" "RULE" "RUN" "RUNNING" "SAMPLE" "SAMPLEID" "SAVE" "SAVEPOINT" "SCHEMA"
 "SCHEMAS" "SCOPE" "SCRATCHPAD" "SCROLL" "SEARCH" "SECOND" "SECONDS"
"SECOND_MICROSECOND" "SECQTY" "SECTION" "SECURITY" "SECURITYAUDIT" "SEEK" "SEL"
 "SELECT" "SEMANTICKEYPHRASETABLE" "SEMANTICSIMILARITYDETAILSTABLE"
"SEMANTICSIMILARITYTABLE" "SENSITIVE" "SEPARATOR" "SEQUENCE" "SESSION" "SESSION_USER"
 "SET" "SETRESRATE" "SETS" "SETSESSRATE" "SETUSER" "SHARE" "SHOW"
"SHUTDOWN" "SIGNAL" "SIMILAR" "SIMPLE" "SIN" "SINH" "SIZE" "SKEW" "SKIP" "SMALLINT"
 "SOME" "SOUNDEX" "SOURCE" "SPACE" "SPATIAL" "SPECIFIC" "SPECIFICTYPE"
"SPOOL" "SQL" "SQLEXCEPTION" "SQLSTATE" "SQLTEXT" "SQLWARNING" "SQL_BIG_RESULT"
 "SQL_CALC_FOUND_ROWS" "SQL_SMALL_RESULT" "SQRT" "SS" "SSL" "STANDARD"
"START" "STARTING" "STARTUP" "STAT" "STATE" "STATEMENT" "STATIC" "STATISTICS" "STAY"
 "STDDEV_POP" "STDDEV_SAMP" "STEPINFO" "STOGROUP" "STORED" "STORES"
"STRAIGHT_JOIN" "STRING_CS" "STRUCTURE" "STYLE" "SUBMULTISET" "SUBSCRIBER" "SUBSET"
 "SUBSTR" "SUBSTRING" "SUBSTRING_REGEX" "SUCCEEDS" "SUCCESSFUL"
"SUM" "SUMMARY" "SUSPEND" "SYMMETRIC" "SYNONYM" "SYSDATE" "SYSTEM" "SYSTEM_TIME"
 "SYSTEM_USER" "SYSTIMESTAMP" "TABLE" "TABLESAMPLE" "TABLESPACE" "TAN"
"TANH" "TBL_CS" "TEMPORARY" "TERMINATE" "TERMINATED" "TEXTSIZE" "THAN" "THEN"
 "THRESHOLD" "TIME" "TIMESTAMP" "TIMEZONE_HOUR" "TIMEZONE_MINUTE" "TINYBLOB"
"TINYINT" "TINYTEXT" "TITLE" "TO" "TOP" "TRACE" "TRAILING" "TRAN" "TRANSACTION"
 "TRANSLATE" "TRANSLATE_CHK" "TRANSLATE_REGEX" "TRANSLATION" "TREAT"
"TRIGGER" "TRIM" "TRIM_ARRAY" "TRUE" "TRUNCATE" "TRY_CONVERT" "TSEQUAL" "TYPE" "UC"
 "UESCAPE" "UID" "UNDEFINED" "UNDER" "UNDO" "UNION" "UNIQUE"
"UNKNOWN" "UNLOCK" "UNNEST" "UNPIVOT" "UNSIGNED" "UNTIL" "UPD" "UPDATE" "UPDATETEXT"
 "UPPER" "UPPERCASE" "USAGE" "USE" "USER" "USING" "UTC_DATE"
"UTC_TIME" "UTC_TIMESTAMP" "VALIDATE" "VALIDPROC" "VALUE" "VALUES" "VALUE_OF"
 "VARBINARY" "VARBYTE" "VARCHAR" "VARCHAR2" "VARCHARACTER" "VARGRAPHIC"
"VARIABLE" "VARIADIC" "VARIANT" "VARYING" "VAR_POP" "VAR_SAMP" "VCAT" "VERBOSE"
 "VERSIONING" "VIEW" "VIRTUAL" "VOLATILE" "VOLUMES" "WAIT" "WAITFOR"
"WHEN" "WHENEVER" "WHERE" "WHILE" "WIDTH_BUCKET" "WINDOW" "WITH" "WITHIN"
 "WITHIN_GROUP" "WITHOUT" "WLM" "WORK" "WRITE" "WRITETEXT" "XMLCAST" "XMLEXISTS"
"XMLNAMESPACES" "XOR" "YEAR" "YEARS" "YEAR_MONTH" "ZEROFILL" "ZEROIFNULL" "ZONE"
```

Membuat alarm pada kueri Metrics Insights

Anda dapat membuat alarm pada kueri Metrics Insights. Ini membantu Anda memiliki alarm yang melacak beberapa sumber daya tanpa perlu diperbarui nanti. Kueri menangkap sumber daya baru dan sumber daya yang berubah. Misalnya, Anda dapat membuat alarm yang mengawasi penggunaan CPU armada Anda, dan alarm secara otomatis mengevaluasi instansbaru yang Anda luncurkan setelah membuat alarm.

Dalam akun pemantauan yang disiapkan untuk pengamatan CloudWatch lintas akun, alarm Wawasan Metrik Anda dapat menonton sumber daya di akun sumber dan di akun pemantauan itu sendiri. Untuk informasi selengkapnya tentang cara membatasi kueri alarm hanya ke sebuah akun tertentu atau mengelompokkan hasil berdasarkan ID akun, silakan lihat bagian `WHERE` dan bagian `GROUP BY` di Komponen dan sintaks kueri Metrics Insights.

Daftar Isi

Membuat alarm Metrics Insights

Untuk membuat alarm pada kueri Wawasan Metrik menggunakan konsol

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik, Semua metrik.
3. Pilih tab Kueri.
4. (Opsional) Untuk menjalankan kueri sampel yang telah dibuat sebelumnya, pilih Tambahkan kueri dan pilih kueri yang akan dijalankan. Atau, Anda dapat memilih Editor untuk mengedit kueri sampel dan kemudian memilih Jalankan untuk menjalankan kueri yang dimodifikasi.
5. Untuk membuat kueri Anda sendiri, Anda dapat menggunakan tampilan Builder, tampilan Editor, atau kombinasi keduanya. Anda dapat beralih di antara dua tampilan tersebut kapan saja dan melihat pekerjaan Anda yang sedang berlangsung di kedua tampilan.

   Pada tampilan Builder, Anda dapat menelusuri dan memilih namespace metrik, nama metrik, filter, grup, dan opsi urutan. Untuk masing-masing opsi ini, pembuat kueri menawarkan sebuah daftar kemungkinan pilihan dari lingkungan Anda untuk dipilih.

Pada tampilan Editor, Anda dapat mulai menulis kueri Anda. Saat Anda mengetik, editor menawarkan saran berdasarkan karakter yang telah Anda ketik sejauh ini.

> **Important**
>
> Untuk menyetel alarm pada kueri Metrics Insights, kueri harus menampilkan deret waktu tunggal. Jika berisi ekspresi GROUP BY, ekspresi GROUP BY harus dibungkus dalam ekspresi matematika metrik yang mengembalikan satu deret waktu saja sebagai hasil akhir ekspresi.

6. Saat Anda puas dengan kueri Anda, pilih Jalankan.
7. Pilih Buat alarm.
8. Pada Ketentuan, tentukan hal-hal berikut:

   a. Setiap kali ***metrik*** diukur, tentukan apakah metrik harus lebih besar dari, kurang dari, atau sama dengan ambang batas. Di bawah dari..., tentukan nilai ambang batas.

   b. Pilih Konfigurasi tambahan. Untuk Titik data alarm, tentukan berapa banyak periode evaluasi (titik data) yang harus ada dalam status `ALARM` untuk memicu alarm. Jika kedua nilai di sini cocok, Anda membuat alarm yang beralih ke status `ALARM` jika terjadi pelanggaran selama sebanyak itu dalam periode berturut-turut.

      Untuk membuat alarm M dari N, tentukan angka yang lebih rendah untuk nilai pertama dibandingkan dengan nilai yang Anda tentukan untuk nilai kedua. Untuk informasi selengkapnya, lihat Melakukan evaluasi alarm.

   c. Untuk Perlakuan data yang hilang, pilih cara alarm berperilaku ketika beberapa titik data hilang. Untuk informasi selengkapnya, lihat Mengkonfigurasi bagaimana CloudWatch alarm memperlakukan data yang hilang.

9. Pilih Berikutnya.
10. Pada Notifikasi, silakan pilih topik SNS yang akan diberitahukan sebagai notifikasi ketika alarm berada dalam status `ALARM`, status `OK`, atau status `INSUFFICIENT_DATA`.

    Agar alarm mengirimkan beberapa notifikasi untuk status alarm yang sama atau untuk status alarm yang berbeda, silakan pilih Tambahkan notifikasi.

    Agar alarm tidak mengirim notifikasi, silakan pilih Hapus.

11. Agar alarm dapat melakukan tindakan penskalaan otomatis, EC2, atau Systems Manager, pilih tombol yang sesuai dan pilih status alarm serta tindakan yang harus dilakukan. Alarm dapat melakukan tindakan Systems Manager hanya ketika masuk ke status ALARM. Untuk informasi selengkapnya tentang tindakan Systems Manager, lihat Mengkonfigurasi CloudWatch untuk membuat OpsItems dari alarm dan pembuatan Insiden.

    > **Note**
    >
    > Untuk membuat alarm yang melakukan tindakan SSM Incident Manager, Anda harus memiliki izin tertentu. Untuk informasi selengkapnya, lihat contoh kebijakan berbasis identitas untuk Manajer Insiden AWS Systems Manager.

12. Setelah selesai, silakan pilih Berikutnya.
13. Masukkan sebuah nama dan deskripsi untuk alarm tersebut. Nama harus menggunakan karakter ASCII saja. Lalu pilih Berikutnya.
14. Pada Pratinjau dan buat, konfirmasikan bahwa informasi dan ketentuannya sudah sesuai keinginan Anda, kemudian pilih Buat alarm.

Untuk membuat alarm pada kueri Wawasan Metrik menggunakan AWS CLI

- Gunakan perintah `put-metric-alarm` dan tentukan kueri Wawasan Metrik dalam parameter `metrics`. Misalnya, perintah berikut menyetel alarm yang beralih ke status ALARM jika salah satu instans Anda melebihi 50% dalam pemanfaatan CPU.

```
aws cloudwatch put-metric-alarm --alarm-name Metrics-Insights-alarm --
evaluation-periods 1 --comparison-operator GreaterThanThreshold --metrics
 '[{"Id":"m1","Expression":"SELECT MAX(CPUUtilization) FROM SCHEMA(\"AWS/EC2\",
 InstanceId)", "Period":60}]' --threshold 50
```

Kasus data parsial

Jika kueri Wawasan Metrik yang digunakan untuk alarm cocok dengan lebih dari 10.000 metrik, alarm dievaluasi berdasarkan 10.000 metrik pertama yang ditemukan kueri. Ini berarti alarm sedang dievaluasi pada data parsial.

Anda dapat menggunakan metode berikut untuk mengetahui apakah alarm Wawasan Metrik saat ini sedang mengevaluasi status alarmnya berdasarkan data parsial:

- Di konsol, jika Anda memilih alarm untuk melihat halaman Detail, pesan Peringatan evaluasi: Tidak mengevaluasi semua data muncul di halaman itu.
- Anda melihat nilai `PARTIAL_DATA` di `EvaluationState` bidang saat Anda menggunakan AWS CLI perintah deskripsi-alarm atau API. DescribeAlarms

Alarm juga mempublikasikan peristiwa ke Amazon EventBridge ketika masuk ke status data sebagian, sehingga Anda dapat membuat EventBridge aturan untuk menonton peristiwa ini. Dalam peristiwa ini, kolom `evaluationState` memiliki nilai `PARTIAL_DATA`. Berikut ini salah satu contohnya.


```
{
    "version": "0",
    "id": "12345678-3bf9-6a09-dc46-12345EXAMPLE",
    "detail-type": "CloudWatch Alarm State Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2022-11-08T11:26:05Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:my-alarm-name"
    ],
    "detail": {
        "alarmName": "my-alarm-name",
        "state": {
            "value": "ALARM",
            "reason": "Threshold Crossed: 3 out of the last 3 datapoints [20000.0
 (08/11/22 11:25:00), 20000.0 (08/11/22 11:24:00), 20000.0 (08/11/22 11:23:00)] were
 greater than the threshold (0.0) (minimum 1 datapoint for OK -> ALARM transition).",
            "reasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2022-11-08T11:26:05.399+0000\",\"startDate\":\"2022-11-08T11:23:00.000+0000\",
\"period\":60,\"recentDatapoints\":[20000.0,20000.0,20000.0],\"threshold\":0.0,
\"evaluatedDatapoints\":[{\"timestamp\":\"2022-11-08T11:25:00.000+0000\",\"value
\":20000.0}]}",
            "timestamp": "2022-11-08T11:26:05.401+0000",
            "evaluationState": "PARTIAL_DATA"
        },
        "previousState": {
            "value": "INSUFFICIENT_DATA",
            "reason": "Unchecked: Initial alarm creation",
            "timestamp": "2022-11-08T11:25:51.227+0000"
        },
```

```
        "configuration": {
            "metrics": [
                {
                    "id": "m2",
                    "expression": "SELECT SUM(PartialDataTestMetric) FROM
 partial_data_test",
                    "returnData": true,
                    "period": 60
                }
            ]
        }
    }
}
```

Jika kueri untuk alarm menyertakan pernyataan GROUP BY yang awalnya mengembalikan lebih dari 500 deret waktu, alarm dievaluasi berdasarkan 500 deret waktu pertama yang ditemukan kueri. Namun demikian, jika Anda menggunakan klausa ORDER BY, maka semua deret waktu yang ditemukan kueri dipilah, dan 500 yang memiliki nilai tertinggi atau terendah sesuai dengan klausa ORDER BY Anda digunakan untuk mengevaluasi alarm.

Gunakan kueri Wawasan Metrik dengan matematika metrik

Anda dapat menggunakan kueri Wawasan Metrik sebagai input ke fungsi matematika metrik. Untuk informasi selengkapnya tentang pernyataan matematika metrik, silakan lihat Gunakan matematika metrik.

Kueri Wawasan Metrik yang tidak menyertakan klausa GROUP BY mengembalikan satu deret waktu. Oleh karena itu, hasil yang dikembalikan dapat digunakan dengan setiap fungsi matematika metrik yang mengambil satu deret waktu sebagai input.

Kueri Wawasan Metrik yang menyertakan klausa GROUP BY mengembalikan banyak deret waktu. Oleh karena itu, hasil yang dikembalikan dapat digunakan dengan setiap fungsi matematika metrik yang mengambil satu array deret waktu sebagai input.

Misalnya, kueri berikut mengembalikan jumlah total byte yang diunduh untuk setiap bucket di Wilayah, sebagai susunan deret waktu:

```
SELECT SUM(BytesDownloaded)
FROM SCHEMA("AWS/S3", BucketName, FilterId)
WHERE FilterId = 'EntireBucket'
```

```
GROUP BY BucketName
```

Pada grafik di konsol atau dalam GetMetricDataoperasi, hasil kueri ini adalahq1. Kueri ini mengembalikan hasil dalam byte, jadi jika ingin melihat hasilnya sebagai MB, Anda dapat menggunakan fungsi matematika berikut:

```
q1/1024/1024
```

Gunakan bahasa alami untuk menghasilkan dan memperbarui kueri Wawasan CloudWatch Metrik

Fitur ini dalam rilis pratinjau di AS Timur (Virginia N.), AS Barat (Oregon), dan Asia Pasifik (Tokyo) untuk CloudWatch dan dapat berubah sewaktu-waktu.

CloudWatch mendukung kemampuan kueri bahasa alami untuk membantu Anda menghasilkan dan memperbarui kueri untuk Wawasan CloudWatch Metrik dan CloudWatch Wawasan Log.

Dengan kemampuan ini, Anda dapat mengajukan pertanyaan tentang atau menjelaskan CloudWatch data yang Anda cari dalam bahasa Inggris biasa. Kemampuan bahasa alami menghasilkan kueri berdasarkan prompt yang Anda masukkan dan memberikan line-by-line penjelasan tentang cara kerja kueri. Anda juga dapat memperbarui kueri Anda untuk menyelidiki lebih lanjut data Anda.

Tergantung pada lingkungan Anda, Anda dapat memasukkan prompt seperti "Instans Amazon Elastic Compute Cloud mana yang memiliki jaringan tertinggi?" dan "Tunjukkan pada sata 10 Tabel Amazon DynamoDB teratas dengan pembacaan yang dikonsumsi."

Untuk menghasilkan kueri Wawasan CloudWatch Metrik dengan kemampuan ini, buka editor kueri Wawasan CloudWatch Metrik di tampilan pembuat atau editor, lalu pilih Buat kueri.

Important

Untuk menggunakan kemampuan kueri bahasa alami, Anda harus menggunakan CloudWatchFullAccess, CloudWatchReadOnlyAccess, CloudWatchFullAccessV2 AdministratorAccess, atau ReadOnlyAccesskebijakan.
Anda juga dapat menyertakan tindakan `cloudwatch:GenerateQuery` dalam kebijakan terkelola atau inline pelanggan baru atau yang sudah ada.

Kueri contoh

Contoh di bagian ini menjelaskan cara menghasilkan dan memperbarui kueri menggunakan kemampuan bahasa alami.

> **Note**
>
> Untuk informasi selengkapnya tentang editor dan sintaks kueri Wawasan CloudWatch Metrik, lihat komponen dan sintaks kueri Wawasan CloudWatch Metrik.

Contoh: Menghasilkan kueri bahasa alami

Untuk menghasilkan kueri menggunakan bahasa alami, masukkan prompt dan pilih Hasilkan kueri baru. Contoh ini menunjukkan kueri yang melakukan pencarian dasar.

Prompt

Berikut ini contoh prompt yang mengarahkan kemampuan untuk pencarian 10 Tabel DynamoDB teratas yang menggunakan kapasitas baca paling banyak.

```
Show top 10 DynamoDB Tables by consumed reads
```

Kueri

Berikut ini contoh kueri yang dihasilkan kemampuan bahasa alami berdasarkan prompt. Perhatikan bagaimana prompt muncul di komentar sebelum kueri. Setelah kueri, Anda dapat membaca penjelasan yang menggambarkan cara kerja kueri.

```
# Show top 10 DynamoDB Tables by consumed reads
SELECT SUM("ConsumedReadCapacityUnits")
FROM "AWS/DynamoDB"
GROUP BY TableName
ORDER BY SUM() DESC
LIMIT 10
# This query selects the sum of consumed read capacity units for each DynamoDB table,
 groups the results by table name, orders the results from highest to lowest read
 capacity consumption, and limits the results to the top 10 tables.
```

> **Note**
>
> Untuk mematikan tampilan prompt Anda dan penjelasan tentang cara kerja kueri, gunakan ikon roda gigi di editor Anda.

Contoh: Memperbarui kueri bahasa alami

Anda dapat memperbarui kueri dengan mengedit prompt awal dan kemudian memilih Perbarui kueri.

Prompt yang diperbarui

Contoh berikut menunjukkan versi yang diperbarui dari prompt sebelumnya. Alih-alih prompt yang mencari 10 Tabel DynamoDB teratas yang menggunakan kapasitas baca paling banyak, prompt ini sekarang mengarahkan kemampuan untuk memilah hasil berdasarkan jumlah byte yang dikembalikan.

```
Sort by bytes returned instead
```

Kueri yang diperbarui

Berikut ini contoh kueri yang diperbarui. Perhatikan bagaimana prompt muncul di komentar sebelum kueri yang diperbarui. Setelah kueri, Anda dapat membaca penjelasan yang menggambarkan bagaimana kueri asli diperbarui.

```
# Sort by bytes returned instead
SELECT SUM("ReturnedBytes")
FROM "AWS/DynamoDB"
GROUP BY TableName
ORDER BY SUM() DESC
LIMIT 10
# This query modifies the original query to select the sum of returned bytes instead
 of consumed read capacity units, and orders the results from highest to lowest sum of
 returned bytes, limiting the results to the top 10 tables.
```

Memilih untuk tidak menggunakan data Anda untuk perbaikan layanan

Data prompt bahasa alami yang Anda berikan untuk melatih model AI dan menghasilkan kueri yang relevan digunakan semata-mata untuk menyediakan dan memelihara layanan Anda. Data ini dapat digunakan untuk meningkatkan kualitas Wawasan CloudWatch Metrik. Kepercayaan dan privasi

Anda, serta keamanan konten Anda, menjadi prioritas utama kami. Untuk informasi selengkapnya, silakan lihat AWS Ketentuan Layanan dan AWS kebijakan AI yang bertanggung jawab.

Anda dapat memilih untuk tidak menggunakan konten Anda untuk mengembangkan atau memperbaiki mutu kueri bahasa alami dengan membuat kebijakan penolakan layanan AI. Untuk memilih keluar dari pengumpulan data untuk semua fitur CloudWatch AI, termasuk kemampuan pembuatan kueri, Anda harus membuat kebijakan opt-out. CloudWatch Untuk informasi selengkapnya, silakan lihat kebijakan penolakan layanan AI di AWS Organizations Panduan Pengguna.

Inferensi SQL

CloudWatch Metrics Insights menggunakan beberapa mekanisme untuk menyimpulkan maksud dari kueri SQL yang diberikan.

Topik

- Bucketing waktu
- Proyeksi bidang
- Agregasi global ORDER BY

Bucketing waktu

Titik data deret waktu yang dihasilkan dari kueri digulirkan ke dalam bucket waktu berdasarkan periode yang diminta. Untuk menggabungkan nilai dalam SQL standar, klausa GROUP BY eksplisit harus didefinisikan untuk mengumpulkan semua pengamatan periode tertentu bersama-sama. Karena ini adalah cara standar untuk menanyakan data deret waktu, CloudWatch Metrics Insights menyimpulkan bucketing waktu tanpa perlu mengekspresikan klausa GROUP BY eksplisit.

Misalnya, ketika kueri dilakukan dengan periode satu menit, semua pengamatan yang termasuk dalam menit itu hingga berikutnya (dikecualikan) digulirkan ke waktu mulai bucket waktu. Ini membuat pernyataan SQL Wawasan Metrik lebih ringkas dan tidak bertele-tele.

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/EC2", InstanceId)
```

Kueri sebelumnya mengembalikan satu deret waktu (pasangan timestamp-nilai tunggal), yang mewakili rata-rata penggunaan CPU semua instans Amazon EC2. Dengan asumsi periode yang diminta adalah satu menit, setiap titik data yang dikembalikan mewakili rata-rata semua pengamatan

yang diukur dalam interval satu menit tertentu (termasuk waktu mulai, waktu akhir eksklusif). Timestamp terkait titik data tertentu adalah waktu mulai bucket

Proyeksi bidang

Kueri Wawasan Metrik selalu menampilkan proyeksi timestamp. Anda tidak perlu menentukan kolom timestamp di klausa SELECT untuk mendapatkan timestamp setiap nilai titik data yang sesuai. Untuk detail tentang bagaimana timestamp dihitung, silakan lihat Bucketing waktu.

Saat menggunakan GROUP BY, setiap nama kelompok juga disimpulkan dan diproyeksikan dalam hasilnya, sehingga Anda dapat mengelompokkan deret waktu yang dikembalikan.

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/EC2", InstanceId)
GROUP BY InstanceId
```

Kueri sebelumnya mengembalikan deret waktu untuk setiap instans Amazon EC2. Setiap deret waktu diberi label setelah nilai ID instans.

Agregasi global ORDER BY

Saat menggunakan ORDER BY, FUNCTION () menyimpulkan fungsi agregat mana yang ingin Anda urutkan (nilai titik data dari metrik yang ditanyakan). Operasi agregat dilakukan di seluruh titik data yang cocok dari setiap deret waktu individu di dalam jendela waktu yang ditanyakan.

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/EC2", InstanceId)
GROUP BY InstanceId
ORDER BY MAX()
LIMIT 10
```

Kueri sebelumnya mengembalikan penggunaan CPU untuk setiap instans Amazon EC2, membatasi hasilnya hingga 10 entri. Hasilnya diurutkan berdasarkan nilai maksimum deret waktu individu dalam jendela waktu yang diminta. Klausa ORDER BY diterapkan sebelum LIMIT, sehingga pengurutan dihitung terhadap lebih dari 10 deret waktu.

Kueri sampel Metrics Insights

Bagian ini berisi contoh kueri Wawasan CloudWatch Metrik yang berguna yang dapat Anda salin dan gunakan secara langsung atau salin dan modifikasi di editor kueri. Beberapa contoh ini sudah

tersedia di konsol, dan Anda dapat mengaksesnya dengan memilih Tambahkan kueri di tampilan Metrik.

Contoh Penyeimbang Beban Aplikasi

Total permintaan di semua penyeimbang beban

```
SELECT SUM(RequestCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer)
```

10 teratas penyeimbang beban paling aktif

```
SELECT MAX(ActiveConnectionCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer)
GROUP BY LoadBalancer
ORDER BY SUM() DESC
LIMIT 10
```

AWS Contoh penggunaan API

20 AWS API teratas berdasarkan jumlah panggilan di akun Anda

```
SELECT COUNT(CallCount)
FROM SCHEMA("AWS/Usage", Class, Resource, Service, Type)
WHERE Type = 'API'
GROUP BY Service, Resource
ORDER BY COUNT() DESC
LIMIT 20
```

CloudWatch API diurutkan berdasarkan panggilan

```
SELECT COUNT(CallCount)
FROM SCHEMA("AWS/Usage", Class, Resource, Service, Type)
WHERE Type = 'API' AND Service = 'CloudWatch'
GROUP BY Resource
ORDER BY COUNT() DESC
```

Contoh DynamoDB

10 teratas tabel dengan bacaan yang dikonsumsi

```
SELECT SUM(ProvisionedWriteCapacityUnits)
FROM SCHEMA("AWS/DynamoDB", TableName)
GROUP BY TableName
ORDER BY MAX() DESC LIMIT 10
```

10 teratas tabel dengan byte yang dikembalikan

```
SELECT SUM(ReturnedBytes)
FROM SCHEMA("AWS/DynamoDB", TableName)
GROUP BY TableName
ORDER BY MAX() DESC LIMIT 10
```

10 teratas tabel berdasarkan kesalahan pengguna

```
SELECT SUM(UserErrors)
FROM SCHEMA("AWS/DynamoDB", TableName)
GROUP BY TableName
ORDER BY MAX() DESC LIMIT 10
```

Contoh Amazon Elastic Block Store

10 teratas Volume Amazon EBS berdasarkan byte yang ditulis

```
SELECT SUM(VolumeWriteBytes)
FROM SCHEMA("AWS/EBS", VolumeId)
GROUP BY VolumeId
ORDER BY SUM() DESC
LIMIT 10
```

Rata-rata waktu tulis volume Amazon EBS

```
SELECT AVG(VolumeTotalWriteTime)
FROM SCHEMA("AWS/EBS", VolumeId)
```

Contoh-contoh Amazon EC2

Penggunaan CPU instans EC2 yang diurutkan berdasarkan tertinggi

```
SELECT AVG(CPUUtilization)
```

```
  FROM SCHEMA("AWS/EC2", InstanceId)
  GROUP BY InstanceId
  ORDER BY AVG() DESC
```

Penggunaan CPU rata-rata di seluruh armada

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/EC2", InstanceId)
```

10 teratas instans berdasarkan penggunaan CPU tertinggi

```
SELECT MAX(CPUUtilization)
FROM SCHEMA("AWS/EC2", InstanceId)
GROUP BY InstanceId
ORDER BY MAX() DESC
LIMIT 10
```

Dalam hal ini, CloudWatch agen mengumpulkan **`CPUUtilization`** metrik per aplikasi. Kueri ini menyaring rata-rata metrik ini untuk nama aplikasi tertentu.

```
SELECT AVG(CPUUtilization)
FROM "AWS/CWAgent"
WHERE ApplicationName = 'eCommerce'
```

Contoh Layanan Amazon Elastic Container

Rata-rata penggunaan CPU di seluruh klaster ECS

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/ECS", ClusterName)
```

10 teratas klaster berdasarkan penggunaan memori

```
SELECT AVG(MemoryUtilization)
FROM SCHEMA("AWS/ECS", ClusterName)
GROUP BY ClusterName
ORDER BY AVG() DESC
LIMIT 10
```

10 teratas layanan berdasarkan penggunaan CPU

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/ECS", ClusterName, ServiceName)
GROUP BY ClusterName, ServiceName
ORDER BY AVG() DESC
LIMIT 10
```

10 teratas layanan dengan menjalankan tugas (Wawasan Kontainer)

```
SELECT AVG(RunningTaskCount)
FROM SCHEMA("ECS/ContainerInsights", ClusterName, ServiceName)
GROUP BY ClusterName, ServiceName
ORDER BY AVG() DESC
LIMIT 10
```

Contoh Amazon Elastic Kubernetes Service Container Insights

Rata-rata penggunaan CPU di seluruh klaster EKS

```
SELECT AVG(pod_cpu_utilization)
FROM SCHEMA("ContainerInsights", ClusterName)
```

10 teratas klaster berdasarkan penggunaan CPU simpul

```
SELECT AVG(node_cpu_utilization)
FROM SCHEMA("ContainerInsights", ClusterName)
GROUP BY ClusterName
ORDER BY AVG() DESC LIMIT 10
```

10 teratas klaster berdasarkan penggunaan memori pod

```
SELECT AVG(pop_memory_utilization)
FROM SCHEMA("ContainerInsights", ClusterName)
GROUP BY ClusterName
ORDER BY AVG() DESC LIMIT 10
```

10 teratas simpul berdasarkan penggunaan CPU

```
SELECT AVG(node_cpu_utilization)
```

```
FROM SCHEMA("ContainerInsights", ClusterName, NodeName)
GROUP BY ClusterName, NodeName
ORDER BY AVG() DESC LIMIT 10
```

10 teratas pod berdasarkan penggunaan memori

```
SELECT AVG(pod_memory_utilization)
FROM SCHEMA("ContainerInsights", ClusterName, PodName)
GROUP BY ClusterName, PodName
ORDER BY AVG() DESC LIMIT 10
```

EventBridge contoh

10 teratas aturan berdasarkan invokasi

```
SELECT SUM(Invocations)
FROM SCHEMA("AWS/Events", RuleName)
GROUP BY RuleName
ORDER BY MAX() DESC LIMIT 10
```

10 teratas aturan berdasarkan invokasi yang gagal

```
SELECT SUM(FailedInvocations)
FROM SCHEMA("AWS/Events", RuleName)
GROUP BY RuleName
ORDER BY MAX() DESC LIMIT 10
```

10 teratas aturan berdasarkan aturan yang cocok

```
SELECT SUM(MatchedEvents)
FROM SCHEMA("AWS/Events", RuleName)
GROUP BY RuleName
ORDER BY MAX() DESC LIMIT 10
```

Contoh Kinesis

10 teratas aliran berdasarkan byte yang ditulis

```
SELECT SUM("PutRecords.Bytes")
```

```
FROM SCHEMA("AWS/Kinesis", StreamName)
GROUP BY StreamName
ORDER BY SUM() DESC LIMIT 10
```

10 teratas aliran berdasarkan item paling awal dalam aliran

```
SELECT MAX("GetRecords.IteratorAgeMilliseconds")
FROM SCHEMA("AWS/Kinesis", StreamName)
GROUP BY StreamName
ORDER BY MAX() DESC LIMIT 10
```

Contoh Lambda

Fungsi Lambda yang diurutkan berdasarkan jumlah invokasi

```
SELECT SUM(Invocations)
FROM SCHEMA("AWS/Lambda", FunctionName)
GROUP BY FunctionName
ORDER BY SUM() DESC
```

10 teratas Fungsi Lambda berdasarkan runtime terpanjang

```
SELECT AVG(Duration)
FROM SCHEMA("AWS/Lambda", FunctionName)
GROUP BY FunctionName
ORDER BY MAX() DESC
LIMIT 10
```

10 teratas Fungsi Lambda berdasarkan jumlah kesalahan

```
SELECT SUM(Errors)
FROM SCHEMA("AWS/Lambda", FunctionName)
GROUP BY FunctionName
ORDER BY SUM() DESC
LIMIT 10
```

CloudWatch Contoh log

10 teratas grup log berdasarkan peristiwa yang masuk

```
SELECT SUM(IncomingLogEvents)
FROM SCHEMA("AWS/Logs", LogGroupName)
GROUP BY LogGroupName
ORDER BY SUM() DESC LIMIT 10
```

10 teratas grup log berdasarkan byte tertulis

```
SELECT SUM(IncomingBytes)
FROM SCHEMA("AWS/Logs", LogGroupName)
GROUP BY LogGroupName
ORDER BY SUM() DESC LIMIT 10
```

Contoh-contoh Amazon RDS

10 teratas Instans Amazon RDS berdasarkan penggunaan CPU tertinggi

```
SELECT MAX(CPUUtilization)
FROM SCHEMA("AWS/RDS", DBInstanceIdentifier)
GROUP BY DBInstanceIdentifier
ORDER BY MAX() DESC
LIMIT 10
```

10 teratas klaster Amazon RDS berdasarkan tulisan

```
SELECT SUM(WriteIOPS)
FROM SCHEMA("AWS/RDS", DBClusterIdentifier)
GROUP BY DBClusterIdentifier
ORDER BY MAX() DESC
LIMIT 10
```

Contoh Amazon Simple Storage Service

Latensi rata-rata berdasarkan bucket

```
SELECT AVG(TotalRequestLatency)
FROM SCHEMA("AWS/S3", BucketName, FilterId)
WHERE FilterId = 'EntireBucket'
GROUP BY BucketName
ORDER BY AVG() DESC
```

10 teratas bucket berdasarkan byte yang diunduh

```
SELECT SUM(BytesDownloaded)
FROM SCHEMA("AWS/S3", BucketName, FilterId)
WHERE FilterId = 'EntireBucket'
GROUP BY BucketName
ORDER BY SUM() DESC
LIMIT 10
```

Contoh Amazon Simple Notification Service

Total pesan yang diterbitkan oleh topik SNS

```
SELECT SUM(NumberOfMessagesPublished)
FROM SCHEMA("AWS/SNS", TopicName)
```

10 teratas topik berdasarkan pesan yang diterbitkan

```
SELECT SUM(NumberOfMessagesPublished)
FROM SCHEMA("AWS/SNS", TopicName)
GROUP BY TopicName
ORDER BY SUM() DESC
LIMIT 10
```

10 teratas topik berdasarkan kegagalan pengiriman pesan

```
SELECT SUM(NumberOfNotificationsFailed)
FROM SCHEMA("AWS/SNS", TopicName)
GROUP BY TopicName
ORDER BY SUM() DESC
LIMIT 10
```

Contoh Amazon SQS

10 teratas kueri berdasarkan umur jumlah pesan yang terlihat

```
SELECT AVG(ApproximateNumberOfMessagesVisible)
FROM SCHEMA("AWS/SQS", QueueName)
GROUP BY QueueName
ORDER BY AVG() DESC
```

```
LIMIT 10
```

10 teratas kueri paling aktif

```
SELECT SUM(NumberOfMessagesSent)
FROM SCHEMA("AWS/SQS", QueueName)
GROUP BY QueueName
ORDER BY SUM() DESC
LIMIT 10
```

10 teratas antrian berdasarkan umur pesan paling awal

```
SELECT AVG(ApproximateAgeOfOldestMessage)
FROM SCHEMA("AWS/SQS", QueueName)
GROUP BY QueueName
ORDER BY AVG() DESC
LIMIT 10
```

Batas Wawasan Metrik

CloudWatch Metrics Insights saat ini memiliki batasan berikut:

- Saat ini, Anda hanya dapat menanyakan tiga jam data terbaru.
- Satu kueri dapat memproses tidak lebih dari 10.000 metrik. Ini berarti bahwa jika klausa SELECT, FROM, dan WHERE cocok dengan lebih dari 10.000 metrik, kueri hanya memproses 10.000 metrik pertama yang ditemukannya.
- Satu kueri dapat mengembalikan tidak lebih dari 500 deret waktu. Ini berarti bahwa jika kueri akan mengembalikan lebih dari 500 metrik, tidak semua metrik akan dikembalikan dalam hasil kueri. Jika Anda menggunakan klausa ORDER BY, maka semua metrik yang sedang diproses akan diurutkan, dan 500 yang memiliki nilai tertinggi atau terendah sesuai dengan klausa ORDER BY Anda dikembalikan.

  Jika Anda tidak menyertakan klausa ORDER BY, Anda tidak dapat mengontrol 500 metrik pencocokan mana yang dikembalikan.
- Anda dapat memiliki sebanyak 200 alarm Wawasan Metrik per Wilayah.
- Wawasan Metrik tidak mendukung data resolusi tinggi, yang merupakan data metrik yang dilaporkan dengan pedetail kurang dari satu menit. Jika Anda meminta data resolusi tinggi, permintaan tidak gagal, tetapi output dikumpulkan pada pedetail satu menit.

- Setiap GetMetricDataoperasi hanya dapat memiliki satu kueri, tetapi Anda dapat memiliki beberapa widget di dasbor yang masing-masing menyertakan kueri.

Glosarium Wawasan Metrik

label

Dalam Metrics Insights, label adalah pasangan kunci-nilai yang digunakan untuk menyaring kueri agar mengembalikan satu himpunan data tertentu, atau untuk mendefinisikan kriteria di mana hasil kueri dipisahkan menjadi deret waktu terpisah. Kunci label mirip dengan nama kolom di SQL. Saat ini, label harus dimensi CloudWatch metrik.

observasi

Pengamatan adalah nilai yang dicatat untuk metrik tertentu pada waktu tertentu.

Pemecahan Masalah Metrik

Hasilnya termasuk "Lainnya," tetapi saya tidak memiliki ini sebagai dimensi

Ini berarti bahwa kueri menyertakan klausa GROUP BY yang menentukan kunci label yang tidak digunakan dalam beberapa metrik yang dikembalikan oleh kueri. Dalam hal ini, grup kosong yang disebut `Other` dikembalikan. Metrik yang tidak menyertakan kunci label tersebut mungkin merupakan metrik agregat yang mengembalikan nilai yang dikumpulkan di semua nilai kunci label tersebut.

Sebagai contoh, anggap kita memiliki kueri berikut:

```
SELECT AVG(Faults)
FROM MyCustomNamespace
GROUP BY Operation, ServiceName
```

Jika beberapa metrik yang dikembalikan tidak mencakup `ServiceName` sebagai dimensi, maka metrik tersebut ditampilkan memiliki `Other` sebagai nilai untuk `ServiceName`.

Untuk mencegah melihat "Lainnya" dalam hasil Anda, gunakan SCHEMA dalam klausa FROM Anda, seperti pada contoh berikut:

```
SELECT AVG(Faults)
FROM SCHEMA(MyCustomNamespace, Operation)
```

```
GROUP BY Operation, ServiceName
```

Ini membatasi hasil yang dikembalikan hanya pada metrik yang memiliki dimensi `Operation` dan dimensi `ServiceName`.

Timestamp tertua dalam grafik saya memiliki nilai metrik yang lebih rendah daripada yang lain

CloudWatch Metrics Insights saat ini hanya mendukung tiga jam data terbaru. Ketika Anda membuat grafik dengan periode yang lebih lama dari satu menit, mungkin ada kasus di mana titik data tertua berbeda dengan nilai yang diharapkan. Ini karena kueri Wawasan Metrik hanya menampilkan tiga jam data terbaru. Dalam hal ini, titik data tertua dalam kueri hanya mengembalikan pengamatan yang telah diukur dalam batas tiga jam terakhir, alih-alih mengembalikan semua pengamatan dalam periode titik data tersebut.

Gunakan penjelajah metrik untuk memantau sumber daya menurut tag dan properti mereka

Penjelajah metrik adalah alat berbasis tag yang memungkinkan Anda untuk menyaring, menggabungkan, dan memvisualisasikan metrik Anda berdasarkan tag dan properti sumber daya, untuk meningkatkan kemampuan observabilitas bagi layanan Anda. Alat ini memberikan Anda pengalaman pemecahan masalah yang fleksibel dan dinamis, sehingga Anda dapat membuat bererapa grafik sekaligus dan menggunakan grafik-grafik tersebut untuk membuat dasbor kesehatan untuk aplikasi Anda.

Visualisasi penjelajah metrik bersifat dinamis, sehingga jika ada sumber daya yang cocok dibuat setelah Anda membuat widget penjelajah metrik dan menambahkannya ke dasbor CloudWatch, maka sumber daya baru secara otomatis muncul di widget penjelajah.

Sebagai contoh, jika semua instans produksi EC2 Anda memiliki tag **`production`**, Anda dapat menggunakan penjelajah metrik untuk menyaring dan menggabungkan metrik dari semua instans ini untuk memahami kesehatan dan performanya. Jika instans yang baru dengan tag yang cocok kemudian dibuat, maka instans akan secara otomatis ditambahkan ke widget penjelajah metrik.

> **Note**
>
> Penjelajah metrik memberikan pengalaman di kemudian hari. Sumber daya yang telah dihentikan, atau tidak lagi ada dengan properti atau tag yang Anda tentukan tidak akan

ditampilkan dalam visualisasi. Namun demikian, Anda masih dapat menemukan metriknya untuk sumber daya ini di tampilan metrik CloudWatch.

Dengan penjelajah metrik, Anda dapat memilih bagaimana menggabungkan metrik dari sumber daya yang cocok dengan kriteria, dan apakah akan menampilkan semuanya dalam satu grafik atau pada grafik yang berbeda dalam satu widget penjelajah metrik.

Penjelajah metrik mencakup template-template yang dapat Anda gunakan untuk melihat grafik visualisasi yang bermanfaat dengan satu klik, dan Anda juga dapat memperluas template untuk membuat widget penjelajah metrik yang sepenuhnya dapat diubahsuai.

Penjelajah metrik mendukung metrik-metrik yang dikeluarkan oleh AWS dan metrik EC2 yang dipublikasikan oleh agen CloudWatch, termasuk memori, cakram, dan metrik CPU. Untuk bisa menggunakan penjelajah metrik guna melihat metrik yang dipublikasikan oleh agen CloudWatch, Anda mungkin harus memperbarui berkas konfigurasi agen CloudWatch Anda. Untuk informasi selengkapnya, silakan lihat Konfigurasi agen CloudWatch untuk penjelajah metrik

Untuk membuat visualisasi dengan penjelajah metrik dan secara opsional menambahkannya ke dasbor, ikuti langkah-langkah ini.

Untuk membuat visualisasi dengan penjelajah metrik

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Penjelajah.
3. Lakukan salah satu langkah berikut:
   - Untuk menggunakan templat, pilih di kotak yang saat ini menampilkan Penjelajah Kosong.

     Dengan bergantung pada templat, penjelajah dapat segera menampilkan grafik metrik. Jika tidak, pilih satu atau lebih tag atau properti di kotak Dari dan kemudian data akan muncul. Jika tidak muncul, gunakan pilihan di bagian atas halaman untuk menampilkan rentang waktu yang lebih panjang di grafik.
   - Untuk membuat visualisasi kustom, pada Metrik, pilih metrik tunggal atau semua metrik yang tersedia dari sebuah layanan.

     Setelah memilih sebuah metrik, Anda dapat secara opsional mengulangi langkah ini untuk menambahkan lebih banyak metrik.

4. Untuk setiap metrik yang dipilih, CloudWatch menampilkan statistik yang akan segera digunakan setelah nama metrik. Untuk mengubah ini, pilih nama statistik, dan kemudian pilih statistik yang Anda inginkan.
5. Pada Dari, pilih sebuah tag atau properti sumber daya untuk menyaring hasil Anda.

   Setelah melakukan ini, Anda dapat secara opsional mengulangi langkah ini untuk memilih lebih banyak tag atau properti sumber daya.

   Jika Anda memilih beberapa nilai dari properti yang sama, seperti dua jenis instans EC2, penjelajah akan menampilkan semua sumber daya yang cocok dengan properti yang dipilih. Hal ini akan diperlakukan sebagai operasi ATAU.

   Jika Anda memilih properti atau tag yang berbeda, seperti tag **`Production`** dan jenis instans M5, hanya sumber daya yang cocok dengan semua pilihan ini yang akan ditampilkan. Hal ini akan diperlakukan sebagai operasi DAN.
6. (Opsional) Untuk Digabungkan dengan, pilih statistik yang digunakan untuk menggabungkan metrik. Lalu, di samping untuk, pilih cara menggabungkan metrik dari daftar. Anda dapat menggabungkan bersama semua sumber daya yang saat ini ditampilkan, atau digabungkan dengan satu tag atau properti sumber daya.

   Tergantung pada bagaimana Anda memilih untuk menggabungkan, hasilnya bisa saja berupa satu rangkaian waktu tunggal atau beberapa rangkaian waktu.
7. Pada Dibagi oleh, Anda dapat memilih untuk membagi grafik tunggal dengan beberapa rangkaian waktu ke dalam beberapa grafik. Pembagian ini dapat dilakukan dengan beragam kriteria, yang dapat Anda pilih pada Dibagi oleh.
8. Pada Pilihan grafik, Anda dapat menyempurnakan grafik dengan mengubah periode, jenis grafik, penempatan keterangan, dan tata letak.
9. Untuk menambahkan visualisasi ini sebagai sebuah widget ke dasbor CloudWatch, pilih Tambahkan ke dasbor.

Konfigurasi agen CloudWatch untuk penjelajah metrik

Untuk mengaktifkan penjelajah metrik untuk menjelajahi metrik EC2 yang dipublikasikan oleh agen CloudWatch, pastikan berkas konfigurasi agen CloudWatch berisikan nilai-nilai berikut:

- Di bagian `metrics`, pastikan bahwa parameter `aggregation_dimensions` memasukkan `["InstanceId"]`. Bagian itu juga bisa berisikan dimensi lain.

- Di bagian `metrics`, pastikan bahwa parameter `append_dimensions` memasukkan baris `{"InstanceId":"${aws:InstanceId}"}`. Bagian tersebut juga bisa berisikan baris lain.
- Di bagian `metrics`, di dalam bagian `metrics_collected`, periksa bagian untuk setiap jenis sumber daya yang Anda harap dapat ditemukan oleh penjelajah metrik, seperti bagian `cpu`, `disk`, dan `memory`. Pastikan bahwa masing-masing bagian ini memiliki `"resources": [ "*"]` `line.`.
- Di bagian `cpu` dari bagian `metrics_collected`, pastikan ada baris `"totalcpu": true`.
- Anda harus menggunakan ruang nama `CWAgent` bawaan untuk metrik yang dikumpulkan oleh agen CloudWatch, dan bukannya ruang nama yang dikustom.

Pengaturan dalam daftar sebelumnya akan menyebabkan agen CloudWatch memublikasikan metrik kumpulan untuk cakram, CPU, dan sumber daya lain yang dapat diplot dalam penjelajah metrik untuk semua instans yang menggunakannya.

Pengaturan ini akan memublikasikan ulang metrik-metrik yang sebelumnya telah Anda siapkan untuk dipublikasikan dengan berbagai dimensi, yang akan menambah biaya metrik Anda.

Untuk informasi selengkapnya tentang cara mengedit file konfigurasi agen CloudWatch, silakan lihat Buat atau edit file konfigurasi CloudWatch agen secara manual.

Gunakan stream metrik

Anda dapat menggunakan aliran metrik untuk terus mengalirkan CloudWatch metrik ke tujuan pilihan Anda, dengan near-real-time pengiriman dan latensi rendah. Tujuan yang didukung termasuk AWS tujuan seperti Amazon Simple Storage Service dan beberapa tujuan penyedia layanan pihak ketiga.

Ada tiga skenario penggunaan utama untuk aliran CloudWatch metrik:

- Penyiapan khusus dengan Firehose — Buat aliran metrik dan arahkan ke aliran pengiriman Amazon Data Firehose yang mengirimkan CloudWatch metrik Anda ke tempat yang Anda inginkan. Anda dapat mengalirkannya ke data lake seperti Amazon S3, atau ke tujuan atau titik akhir apa pun yang didukung oleh Firehose termasuk penyedia pihak ketiga. Format JSON, OpenTelemetry 1.0.0, dan OpenTelemetry 0.7.0 didukung secara native, atau Anda dapat mengonfigurasi transformasi dalam aliran pengiriman Firehose untuk mengonversi data ke format lain seperti Parket. Dengan aliran metrik, Anda dapat terus memperbarui data pemantauan, atau menggabungkan data CloudWatch metrik ini dengan data penagihan dan kinerja untuk membuat kumpulan data yang kaya. Anda kemudian dapat menggunakan alat-alat seperti Amazon Athena

untuk mendapatkan wawasan optimisasi biaya, performa sumber daya, dan pemanfaatan sumber daya.

- Pengaturan S3 Cepat - Lakukan Stream ke Layanan Penyimpanan Amazon Simple dengan proses pengaturan cepat. Secara default, CloudWatch buat sumber daya yang dibutuhkan untuk streaming. Format JSON, OpenTelemetry 1.0.0, dan OpenTelemetry 0.7.0 didukung.
- Pengaturan AWS mitra cepat - CloudWatch memberikan pengalaman penyiapan cepat untuk beberapa mitra pihak ketiga. Anda dapat menggunakan penyedia layanan pihak ketiga untuk memantau, memecahkan masalah, dan menganalisis aplikasi Anda menggunakan data yang dialirkan CloudWatch . Saat Anda menggunakan alur kerja penyiapan mitra cepat, Anda hanya perlu menyediakan URL tujuan dan kunci API untuk tujuan Anda, dan CloudWatch menangani sisa penyiapan. Pengaturan mitra cepat tersedia untuk penyedia pihak ketiga berikut:

  - Datadog
  - Dynatrace
  - Relik Baru
  - Cloud Observabilitas Splunk
  - SumoLogic

Anda dapat melakukan streaming semua CloudWatch metrik Anda, atau menggunakan filter untuk melakukan streaming hanya metrik tertentu. Setiap stream metrik dapat mencakup hingga 1000 filter yang memasukkan atau mengecualikan namespace metrik atau metrik tertentu. Sebuah stream metrik tunggal hanya dapat memasukkan atau mengecualikan filter, namun tidak keduanya.

Setelah sebuah stream metrik dibuat, jika metrik yang baru dibuat sesuai dengan filter yang ada, metrik yang baru akan otomatis dimasukkan ke dalam stream.

Tidak ada batasan jumlah stream metrik per akun atau per Wilayah, dan tidak ada batasan jumlah pembaruan metrik yang sedang dialirkan.

Setiap aliran dapat menggunakan format JSON, OpenTelemetry 1.0.0, atau OpenTelemetry format 0.7.0. Anda dapat mengedit format keluaran aliran metrik kapan saja, seperti untuk meningkatkan dari OpenTelemetry 0.7.0 ke 1.0.0. OpenTelemetry Untuk informasi selengkapnya tentang format output, silakan lihat Format hasil akhir stream metrik.

Untuk stream metrik di akun pemantauan, Anda dapat memilih apakah akan memasukkan metrik dari akun sumber yang ditautkan ke akun pemantauan tersebut. Untuk informasi selengkapnya, lihat CloudWatch observabilitas lintas akun.

Stream metrik selalu memasukkan `Minimum`, `Maximum`, `SampleCount`, dan statistik `Sum`. Anda juga dapat memilih untuk memasukkan statistik tambahan dengan biaya tambahan. Untuk informasi selengkapnya, lihat Statistik yang dapat di-stream.

Penetapan harga stream metrik adalah berdasarkan pada jumlah pembaruan metrik. Anda juga dikenakan biaya dari Firehose untuk aliran pengiriman yang digunakan untuk aliran metrik. Untuk informasi selengkapnya, lihat CloudWatch Harga Amazon.

Topik

Menyiapkan sebuah stream metrik

Gunakan langkah-langkah di bagian berikut untuk menyiapkan aliran CloudWatch metrik.

Setelah aliran metrik dibuat, waktu yang diperlukan untuk data metrik muncul di tujuan bergantung pada pengaturan buffering yang dikonfigurasi pada aliran pengiriman Firehose. Pemrosesan dinyatakan dalam ukuran muatan maksimum atau waktu tunggu maksimum, mana saja yang tercapai lebih dulu. Jika ini disetel ke nilai minimum (60 detik, 1MB) latensi yang diharapkan adalah dalam 3 menit jika CloudWatch ruang nama yang dipilih memiliki pembaruan metrik aktif.

Dalam aliran CloudWatch metrik, data dikirim setiap menit. Data mungkin tiba di tujuan akhir dalam keadaan rusak. Semua metrik yang ditentukan dalam ruang nama yang ditentukan dikirim dalam aliran metrik, kecuali metrik dengan stempel waktu yang berumur lebih dari dua hari.

Untuk setiap kombinasi nama metrik dan namespace yang Anda stream, semua kombinasi dimensi dari nama metrik dan namespace tersebut diarahkan.

Untuk stream metrik di akun pemantauan, Anda dapat memilih apakah akan memasukkan metrik dari akun sumber yang ditautkan ke akun pemantauan tersebut. Untuk informasi selengkapnya, lihat CloudWatch observabilitas lintas akun.

Untuk membuat dan mengelola aliran metrik, Anda harus masuk ke akun yang memiliki CloudWatchFullAccesskebijakan dan `iam:PassRole` izin, atau akun yang memiliki daftar izin berikut:

- `iam:PassRole`
- `cloudwatch:PutMetricStream`
- `cloudwatch:DeleteMetricStream`
- `cloudwatch:GetMetricStream`
- `cloudwatch:ListMetricStreams`
- `cloudwatch:StartMetricStreams`
- `cloudwatch:StopMetricStreams`

Jika Anda akan CloudWatch menyiapkan peran IAM yang diperlukan untuk aliran metrik, Anda juga harus memiliki izin `iam:CreateRole` dan`iam:PutRolePolicy`.

> **Important**
>
> Seorang pengguna dengan `cloudwatch:PutMetricStream` memiliki akses data CloudWatch metrik yang sedang dialirkan, bahkan jika mereka tidak memiliki `cloudwatch:GetMetricData` izin.

Topik

Penyiapan khusus dengan Firehose

Gunakan metode ini untuk membuat aliran metrik dan mengarahkannya ke aliran pengiriman Amazon Data Firehose yang mengirimkan CloudWatch metrik ke tempat yang Anda inginkan. Anda dapat mengalirkannya ke data lake seperti Amazon S3, atau ke tujuan atau titik akhir apa pun yang didukung oleh Firehose termasuk penyedia pihak ketiga.

Format JSON, OpenTelemetry 1.0.0, dan OpenTelemetry 0.7.0 didukung secara native, atau Anda dapat mengonfigurasi transformasi dalam aliran pengiriman Firehose untuk mengonversi

data ke format lain seperti Parket. Dengan aliran metrik, Anda dapat terus memperbarui data pemantauan, atau menggabungkan data CloudWatch metrik ini dengan data penagihan dan kinerja untuk membuat kumpulan data yang kaya. Anda kemudian dapat menggunakan alat-alat seperti Amazon Athena untuk mendapatkan wawasan optimisasi biaya, performa sumber daya, dan pemanfaatan sumber daya.

Anda dapat menggunakan CloudWatch konsol, AWS CLI, AWS CloudFormation, atau AWS Cloud Development Kit (AWS CDK) untuk mengatur aliran metrik.

Aliran pengiriman Firehose yang Anda gunakan untuk aliran metrik harus berada di akun yang sama dan Wilayah yang sama tempat Anda mengatur aliran metrik. Untuk mencapai fungsionalitas Lintas wilayah, Anda dapat mengonfigurasi aliran pengiriman Firehose untuk melakukan streaming ke tujuan akhir yang ada di akun berbeda atau Wilayah lain.

CloudWatch konsol

Bagian ini menjelaskan cara menggunakan CloudWatch konsol untuk menyiapkan aliran metrik menggunakan Firehose.

Untuk menyiapkan aliran metrik kustom menggunakan Firehose

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik, stream. Kemudian pilih Buat stream metrik.
3. (Opsional) Jika Anda masuk ke akun yang disiapkan sebagai akun pemantauan dalam pengamatan CloudWatch lintas akun, Anda dapat memilih apakah akan menyertakan metrik dari akun sumber tertaut dalam aliran metrik ini. Untuk memasukkan metrik dari akun sumber, pilih Masukkan metrik akun sumber.
4. Pilih Penyiapan khusus dengan Firehose.
5. Untuk Pilih aliran Firehose Data Kinesis Anda, pilih aliran pengiriman Firehose yang akan digunakan. Itu harus di akun yang sama. Format default untuk opsi ini adalah OpenTelemetry 0.7.0, tetapi Anda dapat mengubah formatnya nanti dalam prosedur ini.

   Kemudian pilih aliran pengiriman Firehose untuk digunakan di bawah Pilih aliran pengiriman Firehose Anda.
6. (Opsional) Anda dapat memilih Pilih peran layanan yang ada untuk menggunakan peran IAM yang ada alih-alih CloudWatch membuat yang baru untuk Anda.

7. (Opsional) Untuk mengubah format hasil akhir dari yang sebelumnya format default untuk skenario Anda, pilih Ubah format hasil akhir. Format yang didukung adalah JSON, OpenTelemetry 1.0.0, dan 0.7.0. OpenTelemetry
8. Agar Metrik dapat dialirkan, pilih Semua metrik atau Pilih metrik.

   Jika Anda memilih Semua metrik, semua metrik dari akun ini akan disertakan dalam aliran.

   Pertimbangkan dengan cermat apakah akan melakukan stream semua metrik, karena semakin banyak metrik yang Anda stream semakin tinggi biaya stream metrik Anda.

   Jika Anda memilih Pilih metrik, lakukan salah satu hal berikut:

   - Untuk mengalirkan sebagian besar ruang nama metrik, pilih Kecualikan dan pilih ruang nama atau metrik yang akan dikecualikan. Saat menentukan namespace di Exclude, Anda dapat memilih beberapa metrik tertentu dari namespace tersebut untuk dikecualikan. Jika Anda memilih untuk mengecualikan namespace tetapi tidak memilih metrik di namespace tersebut, semua metrik dari namespace tersebut akan dikecualikan.
   - Untuk menyertakan hanya beberapa ruang nama metrik atau metrik dalam aliran metrik, pilih Sertakan, lalu pilih ruang nama atau metrik yang akan disertakan. Jika Anda memilih untuk menyertakan namespace tetapi tidak memilih metrik di namespace tersebut, semua metrik dari namespace tersebut disertakan.
9. (Opsional) Untuk mengalirkan statistik tambahan untuk beberapa metrik ini di luar Minimum, Maksimum SampleCount, dan Jumlah, pilih Tambahkan statistik tambahan. Atau pilih Tambahkan metrik yang direkomendasikan untuk menambahkan beberapa statistik yang umum digunakan, atau pilih namespace dan nama metrik secara manual untuk melakukan stream statistik tambahan. Berikutnya, pilih statistik tambahan untuk stream.

   Hingga kemudian pilih grup metrik lain untuk stream kumpulan statistik tambahan yang berbeda, pilih Tambahkan statistik tambahan. Setiap metrik dapat mencakup sebanyak 20 statistik tambahan, dan sebanyak 100 metrik dalam sebuah stream metrik yang dapat mencakup statistik tambahan.

   Melakukan stream statistik tambahan akan menimbulkan lebih banyak biaya. Untuk informasi selengkapnya, lihat Statistik yang dapat di-stream.

   Untuk definisi statistik tambahan, silakan lihat CloudWatch definisi statistik.
10. (Opsional) Lakukan kustomisasi pada nama stream metrik yang baru pada Nama stream metrik.
11. Pilih Buat stream metrik.

AWS CLI atau AWS API

Gunakan langkah-langkah berikut untuk membuat aliran CloudWatch metrik.

Untuk menggunakan AWS CLI atau AWS API untuk membuat aliran metrik

1. Jika Anda melakukan stream ke Amazon S3, pertama-tama buatkah bucket-nya. Untuk informasi selengkapnya, silakan lihat Membuat sebuah bucket .
2. Buat aliran pengiriman Firehose. Untuk informasi selengkapnya, lihat Membuat aliran Firehose.
3. Buat peran IAM yang memungkinkan CloudWatch untuk menulis ke aliran pengiriman Firehose. Untuk informasi selengkapnya tentang konten peran ini, silakan lihat Kepercayaan antara CloudWatch dan Firehose.
4. Gunakan perintah `aws cloudwatch put-metric-stream` CLI atau `PutMetricStream` API untuk membuat aliran CloudWatch metrik.

AWS CloudFormation

Anda dapat menggunakan AWS CloudFormation untuk mengatur aliran metrik. Untuk informasi lebih lanjut, lihat AWS::CloudWatch::MetricStream.

Untuk digunakan AWS CloudFormation untuk membuat aliran metrik

1. Jika Anda melakukan stream ke Amazon S3, pertama-tama buatkah bucket-nya. Untuk informasi selengkapnya, silakan lihat Membuat sebuah bucket .
2. Buat aliran pengiriman Firehose. Untuk informasi selengkapnya, lihat Membuat aliran Firehose.
3. Buat peran IAM yang memungkinkan CloudWatch untuk menulis ke aliran pengiriman Firehose. Untuk informasi selengkapnya tentang konten peran ini, silakan lihat Kepercayaan antara CloudWatch dan Firehose.
4. Buat aliran di AWS CloudFormation. Untuk informasi lebih lanjut, lihat AWS::CloudWatch::MetricStream.

AWS Cloud Development Kit (AWS CDK)

Anda dapat menggunakan AWS Cloud Development Kit (AWS CDK) untuk mengatur aliran metrik.

Untuk menggunakan AWS CDK untuk membuat aliran metrik

1. Jika Anda melakukan stream ke Amazon S3, pertama-tama buatkah bucket-nya. Untuk informasi selengkapnya, silakan lihat Membuat sebuah bucket .
2. Buat aliran pengiriman Firehose. Untuk informasi selengkapnya, lihat Membuat Aliran Pengiriman Firehose Data Amazon.
3. Buat peran IAM yang memungkinkan CloudWatch untuk menulis ke aliran pengiriman Firehose. Untuk informasi selengkapnya tentang konten peran ini, silakan lihat Kepercayaan antara CloudWatch dan Firehose.
4. Buat stream metrik. Sumber daya aliran metrik tersedia AWS CDK sebagai Konstruksi Level 1 (L1) bernama. `CfnMetricStream` Untuk informasi selengkapnya, silakan lihat Menggunakan konstruksi L1.

Gunakan pengaturan Amazon S3 Cepat

Metode Quick S3 Setup berfungsi dengan baik jika Anda ingin mengatur aliran dengan cepat ke Amazon S3 dan Anda tidak memerlukan transformasi pemformatan apa pun di luar format JSON OpenTelemetry , 1.0.0, dan 0.7.0 yang didukung. OpenTelemetry CloudWatch akan membuat semua sumber daya yang diperlukan termasuk aliran pengiriman Firehose dan peran IAM yang diperlukan. Format default untuk opsi ini adalah JSON, tetapi Anda dapat mengubah format saat Anda mengatur stream tersebut.

Atau, jika Anda ingin format akhir berupa format Parquet atau Kolom Baris yang Dioptimalkan (ORC), Anda harus mengikuti langkah-langkah berikut di Penyiapan khusus dengan Firehose.

CloudWatch konsol

Bagian ini menjelaskan cara menggunakan CloudWatch konsol untuk menyiapkan aliran metrik Amazon S3 menggunakan penyiapan Quick S3.

Mengatur sebuah stream metrik menggunakan pengaturan S3 Cepat

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik, stream. Kemudian pilih Buat stream metrik.
3. (Opsional) Jika Anda masuk ke akun yang disiapkan sebagai akun pemantauan dalam pengamatan CloudWatch lintas akun, Anda dapat memilih apakah akan menyertakan metrik dari akun sumber tertaut dalam aliran metrik ini. Untuk memasukkan metrik dari akun sumber, pilih Masukkan metrik akun sumber.

4. Pilih Pengaturan S3 Cepat. CloudWatch akan membuat semua sumber daya yang diperlukan termasuk aliran pengiriman Firehose dan peran IAM yang diperlukan. Format default untuk opsi ini adalah JSON, tetapi Anda dapat mengubah format di kemudian hari dalam prosedur ini.
5. (Opsional) Pilih Pilih sumber daya yang ada untuk menggunakan bucket S3 yang ada atau peran IAM yang ada alih-alih CloudWatch membuat yang baru untuk Anda.
6. (Opsional) Untuk mengubah format hasil akhir dari yang sebelumnya format default untuk skenario Anda, pilih Ubah format hasil akhir. Format yang didukung adalah JSON, OpenTelemetry 1.0.0, dan 0.7.0. OpenTelemetry
7. Agar Metrik dapat dialirkan, pilih Semua metrik atau Pilih metrik.

   Jika Anda memilih Semua metrik, semua metrik dari akun ini akan disertakan dalam aliran.

   Pertimbangkan dengan cermat apakah akan melakukan stream semua metrik, karena semakin banyak metrik yang Anda stream semakin tinggi biaya stream metrik Anda.

   Jika Anda memilih Pilih metrik, lakukan salah satu hal berikut:

   - Untuk mengalirkan sebagian besar ruang nama metrik, pilih Kecualikan dan pilih ruang nama atau metrik yang akan dikecualikan. Saat menentukan namespace di Exclude, Anda dapat memilih beberapa metrik tertentu dari namespace tersebut untuk dikecualikan. Jika Anda memilih untuk mengecualikan namespace tetapi tidak memilih metrik di namespace tersebut, semua metrik dari namespace tersebut akan dikecualikan.
   - Untuk menyertakan hanya beberapa ruang nama metrik atau metrik dalam aliran metrik, pilih Sertakan, lalu pilih ruang nama atau metrik yang akan disertakan. Jika Anda memilih untuk menyertakan namespace tetapi tidak memilih metrik di namespace tersebut, semua metrik dari namespace tersebut disertakan.
8. (Opsional) Untuk mengalirkan statistik tambahan untuk beberapa metrik ini di luar Minimum, Maksimum SampleCount, dan Jumlah, pilih Tambahkan statistik tambahan. Atau pilih Tambahkan metrik yang direkomendasikan untuk menambahkan beberapa statistik yang umum digunakan, atau pilih namespace dan nama metrik secara manual untuk melakukan stream statistik tambahan. Berikutnya, pilih statistik tambahan untuk stream.

   Hingga kemudian pilih grup metrik lain untuk stream kumpulan statistik tambahan yang berbeda, pilih Tambahkan statistik tambahan. Setiap metrik dapat mencakup sebanyak 20 statistik tambahan, dan sebanyak 100 metrik dalam sebuah stream metrik yang dapat mencakup statistik tambahan.

Melakukan stream statistik tambahan akan menimbulkan lebih banyak biaya. Untuk informasi selengkapnya, lihat Statistik yang dapat di-stream.

Untuk definisi statistik tambahan, silakan lihat CloudWatch definisi statistik.

9. (Opsional) Lakukan kustomisasi pada nama stream metrik yang baru pada Nama stream metrik.
10. Pilih Buat stream metrik.

Pengaturan mitra cepat

CloudWatch memberikan pengalaman penyiapan cepat untuk mitra pihak ketiga berikut. Untuk menggunakan alur kerja ini, Anda hanya perlu menyediakan URL tujuan dan kunci API untuk tujuan Anda. CloudWatch menangani sisa penyiapan termasuk membuat aliran pengiriman Firehose dan peran IAM yang diperlukan.

> **Important**
>
> Sebelum Anda menggunakan pengaturan mitra cepat untuk membuat sebuah stream metrik, kami sangat menyarankan Anda membaca dokumentasi mitra tersebut, yang ditautkan dalam daftar berikut.

- Datadog
- Dynatrace
- Relik Baru
- Cloud Observabilitas Splunk
- SumoLogic

Saat Anda mengatur sebuah stream metrik ke salah satu mitra ini, stream akan dibuat dengan beberapa pengaturan default, seperti yang tercantum di bagian berikut.

Topik

- Siapkan sebuah stream metrik menggunakan pengaturan mitra cepat
- Stream datadog default
- Stream Dynatrace default

- Stream Relik baru default
- Stream Cloud Observabilitas Splunk default
- Stream Sumo Logic default

Siapkan sebuah stream metrik menggunakan pengaturan mitra cepat

CloudWatch menyediakan opsi pengaturan cepat untuk beberapa mitra pihak ketiga. Sebelum Anda memulai langkah-langkah di bagian ini, Anda harus memiliki informasi tertentu untuk mitra. Informasi ini mungkin termasuk URL tujuan dan/atau kunci API untuk tujuan partner Anda. Anda juga harus membaca dokumentasi di situs web mitra yang ditautkan di bagian sebelumnya, dan bagian default untuk mitra yang tercantum di bagian berikut.

Untuk melakukan streaming ke tujuan pihak ketiga yang tidak didukung oleh penyiapan cepat, Anda dapat mengikuti petunjuk di Ikuti petunjuk Penyiapan khusus dengan Firehose untuk menyiapkan aliran menggunakan Firehose, lalu mengirim metrik tersebut dari Firehose ke tujuan akhir.

Menggunakan pengaturan mitra cepat untuk membuat stream metrik ke penyedia pihak ketiga

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik, stream. Kemudian pilih Buat stream metrik.
3. (Opsional) Jika Anda masuk ke akun yang disiapkan sebagai akun pemantauan dalam pengamatan CloudWatch lintas akun, Anda dapat memilih apakah akan menyertakan metrik dari akun sumber tertaut dalam aliran metrik ini. Untuk memasukkan metrik dari akun sumber, pilih Masukkan metrik akun sumber.
4. Pilih pengaturan mitra Amazon Web Services Cepat
5. Pilih nama mitra yang ingin Anda stream metrik kepadanya.
6. Untuk URL Titik Akhir, masukkan URL tujuan.
7. Untuk Kunci Akses atau Kunci API, masukkan kunci akses untuk mitra. Tidak semua mitra memerlukan kunci akses.
8. Agar Metrik dapat dialirkan, pilih Semua metrik atau Pilih metrik.

   Jika Anda memilih Semua metrik, semua metrik dari akun ini akan disertakan dalam aliran.

   Pertimbangkan dengan cermat apakah akan melakukan stream semua metrik, karena semakin banyak metrik yang Anda stream semakin tinggi biaya stream metrik Anda.

   Jika Anda memilih Pilih metrik, lakukan salah satu hal berikut:

- Untuk mengalirkan sebagian besar ruang nama metrik, pilih Kecualikan dan pilih ruang nama atau metrik yang akan dikecualikan. Saat menentukan namespace di Exclude, Anda dapat memilih beberapa metrik tertentu dari namespace tersebut untuk dikecualikan. Jika Anda memilih untuk mengecualikan namespace tetapi tidak memilih metrik di namespace tersebut, semua metrik dari namespace tersebut akan dikecualikan.
- Untuk menyertakan hanya beberapa ruang nama metrik atau metrik dalam aliran metrik, pilih Sertakan, lalu pilih ruang nama atau metrik yang akan disertakan. Jika Anda memilih untuk menyertakan namespace tetapi tidak memilih metrik di namespace tersebut, semua metrik dari namespace tersebut disertakan.

9. (Opsional) Untuk mengalirkan statistik tambahan untuk beberapa metrik ini di luar Minimum, Maksimum SampleCount, dan Jumlah, pilih Tambahkan statistik tambahan. Atau pilih Tambahkan metrik yang direkomendasikan untuk menambahkan beberapa statistik yang umum digunakan, atau pilih namespace dan nama metrik secara manual untuk melakukan stream statistik tambahan. Berikutnya, pilih statistik tambahan untuk stream.

   Hingga kemudian pilih grup metrik lain untuk stream kumpulan statistik tambahan yang berbeda, pilih Tambahkan statistik tambahan. Setiap metrik dapat mencakup sebanyak 20 statistik tambahan, dan sebanyak 100 metrik dalam sebuah stream metrik yang dapat mencakup statistik tambahan.

   Melakukan stream statistik tambahan akan menimbulkan lebih banyak biaya. Untuk informasi selengkapnya, lihat Statistik yang dapat di-stream.

   Untuk definisi statistik tambahan, silakan lihat CloudWatch definisi statistik.

10. (Opsional) Lakukan kustomisasi pada nama stream metrik yang baru pada Nama stream metrik.
11. Pilih Buat stream metrik.

Stream datadog default

Stream pengaturan mitra cepat ke Datadog menggunakan default berikut:

- Format keluaran: OpenTelemetry 0.7.0
- Firehose streaming konten encoding GZIP
- Opsi buffering aliran Firehose Interval 60 detik, ukuran 4 MB
- Opsi coba lagi aliran Firehose Durasi 60 detik

Bila Anda menggunakan pengaturan mitra cepat untuk membuat sebuah stream metrik ke Datadog dan Anda melakukan stream metrik tertentu, secara default metrik tersebut memasukkan beberapa statistik tambahan. Melakukan stream statistik tambahan dapat menimbulkan lebih banyak biaya tambahan. Untuk informasi selengkapnya tentang statistik dan biayanya, silakan lihat Statistik yang dapat di-stream.

Daftar berikut menunjukkan metrik yang memiliki statistik tambahan yang di-stream secara default, jika Anda memilih untuk melakukan stream pada metrik tersebut. Anda dapat memilih untuk membatalkan pilihan statistik tambahan ini sebelum memulai stream.

- **`Duration`** di **`AWS/Lambda`**: p50, p80, p95, p99, p99.9
- **`PostRuntimeExtensionDuration`** di **`AWS/Lambda`**: p50, p99
- **`FirstByteLatency`** dan **`TotalRequestLatency`** di **`AWS/S3`**: p50, p90, p95, p99, p99.9
- **`ResponseLatency`** di **`AWS/Polly`** dan **`TargetResponseTime`** di AWS/ApplicationELB: p50, p90, p95, p99
- **`Latency`** dan **`IntegrationLatency`** di **`AWS/ApiGateway`**: p90, p95, p99
- **`Latency`** dan **`TargetResponseTime`** di **`AWS/ELB`**: p95, p99
- **`RequestLatency`** di **`AWS/AppRunner`**: p50, p95, p99
- **`ActivityTime`**, **`ExecutionTime`**, **`LambdaFunctionRunTime`**, **`LambdaFunctionScheduleTime`**, **`LambdaFunctionTime`**, **`ActivityRunTime`**, dan **`ActivityScheduleTime`** di **`AWS/States`**: p95, p99
- **`EncoderBitRate`**,**`ConfiguredBitRate`**, dan **`ConfiguredBitRateAvailable`** di **`AWS/MediaLive`**: p90
- **`Latency`** di **`AWS/AppSync`**: p90

Stream Dynatrace default

Stream pengaturan mitra cepat ke Dynatrace menggunakan default berikut:

- Format keluaran: OpenTelemetry 0.7.0
- Firehose streaming konten encoding GZIP
- Opsi buffering aliran Firehose Interval 60 detik, ukuran 5 MB
- Opsi coba lagi aliran Firehose Durasi 600 detik

Stream Relik baru default

Stream pengaturan mitra cepat ke Relik Baru menggunakan default berikut:

- Format keluaran: OpenTelemetry 0.7.0
- Firehose streaming konten encoding GZIP
- Opsi buffering aliran Firehose Interval 60 detik, ukuran 1 MB
- Opsi coba lagi aliran Firehose Durasi 60 detik

Stream Cloud Observabilitas Splunk default

Stream pengaturan mitra cepat ke Cloud Observabilitas Splunk menggunakan default berikut:

- Format keluaran: OpenTelemetry 0.7.0
- Firehose streaming konten encoding GZIP
- Opsi buffering aliran Firehose Interval 60 detik, ukuran 1 MB
- Opsi coba lagi aliran Firehose Durasi 300 detik

Stream Sumo Logic default

Stream pengaturan mitra cepat ke Sumo Logic menggunakan default berikut:

- Format keluaran: OpenTelemetry 0.7.0
- Firehose streaming konten encoding GZIP
- Opsi buffering aliran Firehose Interval 60 detik, ukuran 1 MB
- Opsi coba lagi aliran Firehose Durasi 60 detik

Statistik yang dapat di-stream

stream metrik selalu memasukkan statistik berikut: `Minimum`, `Maximum`, `SampleCount`, dan `Sum`. Anda juga dapat memilih untuk memasukkan statistik tambahan berikut dalam sebuah stream metrik. Pilihan ini berdasarkan per-metrik. Untuk informasi selengkapnya tentang statistik ini, silakan lihat CloudWatch definisi statistik.

- Nilai persentil seperti p95 atau p99 (Untuk aliran dengan JSON atau format) OpenTelemetry

- Rata-rata terpangkas (Hanya untuk stream dengan format JSON)
- Rata-rata winsorisasi (Hanya untuk stream dengan format JSON)
- Hitungan trim (Hanya untuk stream dengan format JSON)
- Jumlah trim (Hanya untuk stream dengan format JSON)
- Peringkat persentil (Hanya untuk stream dengan format JSON)
- Rata-rata interkuartil (Hanya untuk stream dengan format JSON)

Melakukan stream statistik tambahan akan menimbulkan biaya tambahan. Melakukan stream antara satu dan lima dari statistik tambahan ini untuk metrik tertentu akan ditagihkan sebagai sebuah pembaruan metrik tambahan. Setelahnya, setiap set tambahan yang berisikan hingga lima statistik ini akan ditagihkan sebagai pembaruan metrik lainnya.

Sebagai contoh, misalnya untuk satu metrik yang Anda sedang melakukan pengaliran pada enam statistik tambahan berikut: p95, p99, p99.9, Rata-rata terpangkas, Rata-rata terpotong, dan Jumlah terpangkas. Setiap pembaruan metrik ini akan ditagihkan sebagai tiga pembaruan metrik: satu untuk pembaruan metrik yang mencakup statistik default, satu untuk lima statistik tambahan pertama, dan satu untuk statistik tambahan yang keenam. Menambahkan hingga empat statistik tambahan untuk total sepuluh statistik tidak akan meningkatkan biaya penagihan, tetapi statistik tambahan kesebelas akan meningkatkan biaya penagihan.

Saat Anda menentukan kombinasi nama metrik dan namespace untuk melakukan stream statistik tambahan, semua kombinasi dimensi dari nama metrik dan namespace tersebut di-stream dengan statistik tambahan.

CloudWatch aliran metrik menerbitkan metrik baru`TotalMetricUpdate`, yang mencerminkan jumlah dasar pembaruan metrik ditambah pembaruan metrik tambahan yang dikeluarkan oleh streaming statistik tambahan. Untuk informasi selengkapnya, lihat Pantau aliran metrik Anda dengan CloudWatch metrik.

Untuk informasi selengkapnya, lihat CloudWatch Harga Amazon.

> **Note**
>
> Beberapa metrik tidak mendukung persentil. Statistik persentil untuk metrik-metrik ini dikecualikan dari stream dan tidak dikenakan biaya stream metrik. Sebuah contoh dari statistik ini yang tidak mendukung persentil adalah beberapa metrik di namespace `AWS/ECS`.

Statistik tambahan yang Anda konfigurasikan di-stream hanya jika cocok dengan filter untuk stream. Sebagai contoh, jika Anda membuat sebuah stream yang hanya memiliki `EC2` dan `RDS` di filter yang sudah termasuk, dan lalu konfigurasi statistik Anda menyertakan `EC2` dan `Lambda`, maka stream tersebut memasukkan metrik `EC2` dengan statistik tambahan, metrik `RDS` dengan hanya statistik default, dan tidak memasukkan statistik `Lambda` sama sekali.

Operasi dan pemeliharaan stream metrik

Stream metrik selalu berada di salah satu dari dua status, Berjalan atau Berhenti.

- Berjalan — stream metrik berjalan dengan benar. Mungkin tidak akan ada data metrik apa pun yang di-stream ke tujuan karena adanya filter-filter di stream tersebut.
- Terhenti — Stream metrik telah dihentikan secara eksplisit oleh seseorang, dan bukan karena adanya kesalahan. Mungkin berguna bagi Anda untuk menghentikan stream Anda untuk menjeda sejenak kegiatan stream data tanpa menghapus stream-nya.

Jika Anda menghentikan dan memulai ulang aliran metrik, data metrik yang dipublikasikan CloudWatch saat aliran metrik dihentikan tidak akan diisi kembali ke aliran metrik.

Jika Anda mengubah format hasil akhir sebuah stream metrik, dalam kasus tertentu Anda mungkin akan melihat sejumlah kecil data metrik yang dituliskan ke tujuan dalam format lama maupun format yang baru. Untuk menghindari situasi ini, Anda dapat membuat aliran pengiriman Firehose baru dengan konfigurasi yang sama dengan konfigurasi Anda saat ini, lalu mengubah ke aliran pengiriman Firehose baru dan mengubah format output secara bersamaan. Dengan begini, catatan Kinesis dengan format hasil akhir yang berbeda akan disimpan di Amazon S3 dalam objek terpisah. Kemudian, Anda dapat mengarahkan lalu lintas kembali ke aliran pengiriman Firehose asli dan menghapus aliran pengiriman kedua.

Untuk melihat, mengedit, menghentikan, dan memulai stream metrik Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik, Stream.

   Daftar berisikan stream muncul, dan kolom Status menampilkan apakah setiap stream sedang berjalan atau berhenti.
3. Untuk menghentikan atau memulai sebuah stream metrik, pilih stream lalu pilih Berhenti atau Mulai.
4. Untuk melihat detail tentang sebuah stream metrik, pilih stream lalu pilih Lihat detail.

5. Untuk mengubah format keluaran aliran, filter, aliran Firehose tujuan, atau peran, pilih Edit dan buat perubahan yang Anda inginkan.

   Jika Anda mengubah filter, mungkin ada beberapa celah dalam data metrik selama transisi.

Pantau aliran metrik Anda dengan CloudWatch metrik

Aliran metrik memancarkan CloudWatch metrik tentang kesehatan dan operasinya di namespace. `AWS/CloudWatch/MetricStreams` Metrik-metrik berikut dikeluarkan. Metrik-metrik ini dikeluarkan dengan sebuah dimensi `MetricStreamName` dan tanpa dimensi. Anda dapat menggunakan metrik tersebut dengan tanpa dimensi untuk melihat metrik-metrik gabungan untuk semua stream metrik Anda. Anda dapat menggunakan metrik-metrik tersebut dengan dimensi `MetricStreamName` untuk melihat metrik-metrik tersebut hanya tentang stream metrik tersebut.

Untuk semua metrik ini, nilai-nilai dikeluarkan hanya untuk stream metrik yang berada di status Berjalan.

Metrik	Deskripsi
`MetricUpdate`	Jumlah pembaruan metrik dikirimkan ke stream metrik. Jika tidak ada pembaruan metrik yang di-stream selama jangka waktu tertentu, metrik ini tidak muncul selama jangka waktu tersebut. Jika Anda menghentikan stream metrik tersebut, metrik-metrik ini akan berhenti muncul hingga stream metrik dimulai lagi. Statistik valid: `Sum` Satuan: Tidak ada
`TotalMetricUpdate`	Ini dihitung MetricUpdate sebagai+angka berdasarkan statistik tambahan yang sedang dialirkan. Untuk setiap kombinasi namespace dan nama metrik yang unik, lakukan stream 1-5 statistik tambahan akan menambahkan 1 ke `TotalMetricUpdate` , dan melakukan stream 6-10 statistik tambahan akan menambahkan 2 ke `TotalMetricUpdate` , dan seterusnya. Statistik valid: `Sum`

Metrik	Deskripsi
	Satuan: Tidak ada
`PublishErrorRate`	Jumlah kesalahan yang tidak dapat dipulihkan yang terjadi saat memasukkan data ke aliran pengiriman Firehose. Jika tidak ada kesalahan yang terjadi selama jangka waktu, metrik ini tidak akan muncul selama jangka waktu tersebut. Jika Anda menghentikan stream metrik tersebut, metrik-metrik ini akan berhenti muncul hingga stream metrik dimulai lagi. Statistik yang Benar: `Average` untuk melihat tingkat pembaruan metrik tidak dapat dicatat. Nilai ini akan berada di antara 0,0 dan 1,0. Satuan: Tidak ada

Kepercayaan antara CloudWatch dan Firehose

Aliran pengiriman Firehose harus dipercaya CloudWatch melalui peran IAM yang memiliki izin menulis ke Firehose. Izin ini dapat dibatasi pada aliran pengiriman Firehose tunggal yang CloudWatch digunakan aliran metrik. Peran IAM harus memercayai pengguna utama layanan `streams.metrics.cloudwatch.amazonaws.com`.

Jika Anda menggunakan CloudWatch konsol untuk membuat aliran metrik, Anda dapat CloudWatch membuat peran dengan izin yang benar. Jika Anda menggunakan metode lain untuk membuat sebuah stream metrik, atau Anda ingin membuat peran IAM itu sendiri, maka metode itu harus berisikan kebijakan izin dan kebijakan kepercayaan berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:firehose:region:account-id:deliverystream/*"
        }
```

```
    ]
}
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "streams.metrics.cloudwatch.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Data metrik dialirkan CloudWatch ke aliran pengiriman Firehose tujuan atas nama sumber yang memiliki sumber daya aliran metrik.

Format hasil akhir stream metrik

Data dalam aliran CloudWatch metrik dapat dalam format JSON atau OpenTelemetry format. Saat ini, format OpenTelemetry 1.0.0 dan 0.7.0 didukung.

Daftar Isi

Format JSON

Dalam aliran CloudWatch metrik yang menggunakan format JSON, setiap catatan Firehose berisi beberapa objek JSON yang dipisahkan oleh karakter baris baru (\n). Setiap objek mencakup satu titik data tunggal dari satu metrik tunggal.

Format JSON yang digunakan sepenuhnya kompatibel dengan AWS Glue dan dengan Amazon Athena. Jika Anda memiliki aliran pengiriman Firehose dan AWS Glue tabel diformat dengan benar, format dapat secara otomatis diubah menjadi format Parket atau format Optimized Row Columnar (ORC) sebelum disimpan di S3. Untuk informasi selengkapnya tentang mengubah format, lihat Mengonversi Format Rekaman Input Anda di Firehose. Untuk informasi selengkapnya tentang format yang benar AWS Glue, lihatSkema AWS Glue mana yang harus saya gunakan untuk format hasil akhir JSON?.

Pada format JSON, nilai yang valid untuk `unit` adalah sama seperti untuk nilai `unit` dari Struktur API `MetricDatum`. Untuk informasi lebih lanjut, lihat MetricDatum. Nilai untuk bidang `timestamp` dalam jangka waktu milidetik, misalnya `1616004674229`.

Berikut ini adalah contoh dari format tersebut. Pada contoh ini, JSON diformat agar mudah dibaca, tetapi dalam praktiknya keseluruhan format ada pada satu baris tunggal.

```
{
    "metric_stream_name": "MyMetricStream",
    "account_id": "1234567890",
    "region": "us-east-1",
    "namespace": "AWS/EC2",
    "metric_name": "DiskWriteOps",
    "dimensions": {
        "InstanceId": "i-123456789012"
    },
    "timestamp": 1611929698000,
    "value": {
        "count": 3.0,
        "sum": 20.0,
        "max": 18.0,
        "min": 0.0,
        "p99": 17.56,
        "p99.9": 17.8764,
        "TM(25%:75%)": 16.43
    },
    "unit": "Seconds"
```

```
}
```

Skema AWS Glue mana yang harus saya gunakan untuk format hasil akhir JSON?

Berikut ini adalah contoh representasi JSON `StorageDescriptor` untuk AWS Glue tabel, yang kemudian akan digunakan oleh Firehose. Untuk informasi lebih lanjut tentang`StorageDescriptor`, lihat StorageDescriptor.

```
{
  "Columns": [
    {
      "Name": "metric_stream_name",
      "Type": "string"
    },
    {
      "Name": "account_id",
      "Type": "string"
    },
    {
      "Name": "region",
      "Type": "string"
    },
    {
      "Name": "namespace",
      "Type": "string"
    },
    {
      "Name": "metric_name",
      "Type": "string"
    },
    {
      "Name": "timestamp",
      "Type": "timestamp"
    },
    {
      "Name": "dimensions",
      "Type": "map<string,string>"
    },
    {
      "Name": "value",
      "Type":
 "struct<min:double,max:double,count:double,sum:double,p99:double,p99.9:double>"
    },
```

```
    {
      "Name": "unit",
      "Type": "string"
    }
  ],
  "Location": "s3://my-s3-bucket/",
  "InputFormat": "org.apache.hadoop.mapred.TextInputFormat",
  "OutputFormat": "org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat",
  "SerdeInfo": {
    "SerializationLibrary": "org.apache.hive.hcatalog.data.JsonSerDe"
  },
  "Parameters": {
    "classification": "json"
  }
}
```

Contoh sebelumnya adalah untuk data yang dicatat di Amazon S3 dalam format JSON. Gantilah nilai-nilai di bidang-bidang berikut ini dengan nilai-nilai yang sudah terindikasi untuk menyimpan data dalam format Parquet atau format Optimized Row Columnar (ORC).

- Parquet:
  - InputFormat: org.apache.hadoop.hive.ql.io.parquet. MapredParquetInputFormat
  - OutputFormat: org.apache.hadoop.hive.ql.io.parquet. MapredParquetOutputFormat
  - SerDeInfo.SerializationLib: org.apache.hadoop.hive.ql.io.parquet.serde. ParquetHiveSerDe
  - parameters.classification: parquet
- ORC:
  - InputFormat: org.apache.hadoop.hive.ql.io.orc. OrcInputFormat
  - OutputFormat: org.apache.hadoop.hive.ql.io.orc. OrcOutputFormat
  - SerDeInfo.SerializationLib: org.apache.hadoop.hive.ql.io.orc. OrcSerde
  - parameters.classification: orc

OpenTelemetry 1.0.0 format

> **Note**
>
> Dengan format OpenTelemetry 1.0.0, atribut metrik dikodekan sebagai daftar `KeyValue` objek alih-alih `StringKeyValue` jenis yang digunakan dalam format 0.7.0. Sebagai konsumen, ini merupakan satu-satunya perubahan besar yang terjadi antara format 0.7.0

dan 1.0.0. Pengurai yang dihasilkan dari file proto 0.7.0 tidak akan melakukan penguraian terhadap atribut-atribut metrik yang dikodekan dalam format 1.0.0. Hal yang sama berlaku secara terbalik, pengurai yang dihasilkan dari file proto 1.0.0 tidak akan melakukan penguraian terhadap atribut-atribut metrik yang dikodekan dalam format 0.7.0.

OpenTelemetry adalah kumpulan alat, API, dan SDK. Anda dapat menggunakannya untuk instrumen, menghasilkan, mengumpulkan, dan mengekspor data telemetri (metrik, log, dan jejak) untuk analisis. OpenTelemetry adalah bagian dari Cloud Native Computing Foundation. Untuk informasi lebih lanjut, lihat OpenTelemetry.

Untuk informasi tentang spesifikasi OpenTelemetry 1.0.0 lengkap, lihat Rilis versi 1.0.0.

Catatan Kinesis dapat berisi satu atau lebih struktur `ExportMetricsServiceRequest` OpenTelemetry data. Setiap struktur data dimulai dengan header dengan sebuah `UnsignedVarInt32` yang mengindikasikan panjang catatan dalam Byte. Setiap `ExportMetricsServiceRequest` dapat berisikan data dari beberapa metrik sekaligus.

Berikut ini adalah representasi string dari pesan struktur `ExportMetricsServiceRequest` OpenTelemetry data. OpenTelemetry membuat serial protokol biner Google Protocol Buffers, dan ini tidak dapat dibaca manusia.

```
resource_metrics {
  resource {
    attributes {
      key: "cloud.provider"
      value {
        string_value: "aws"
      }
    }
    attributes {
      key: "cloud.account.id"
      value {
        string_value: "123456789012"
      }
    }
    attributes {
      key: "cloud.region"
      value {
        string_value: "us-east-1"
      }
```

```
    }
    attributes {
      key: "aws.exporter.arn"
      value {
        string_value: "arn:aws:cloudwatch:us-east-1:123456789012:metric-stream/
MyMetricStream"
      }
    }
  }
  scope_metrics {
    metrics {
      name: "amazonaws.com/AWS/DynamoDB/ConsumedReadCapacityUnits"
      unit: "NoneTranslated"
      summary {
        data_points {
          start_time_unix_nano: 60000000000
          time_unix_nano: 120000000000
          count: 1
          sum: 1.0
          quantile_values {
            value: 1.0
          }
          quantile_values {
            quantile: 0.95
            value: 1.0
          }
          quantile_values {
            quantile: 0.99
            value: 1.0
          }
          quantile_values {
            quantile: 1.0
            value: 1.0
          }
          attributes {
            key: "Namespace"
            value {
              string_value: "AWS/DynamoDB"
            }
          }
          attributes {
            key: "MetricName"
            value {
              string_value: "ConsumedReadCapacityUnits"
```

```
            }
          }
          attributes {
            key: "Dimensions"
            value {
              kvlist_value {
                values {
                  key: "TableName"
                  value {
                    string_value: "MyTable"
                  }
                }
              }
            }
          }
        }
        data_points {
          start_time_unix_nano: 70000000000
          time_unix_nano: 130000000000
          count: 2
          sum: 5.0
          quantile_values {
            value: 2.0
          }
          quantile_values {
            quantile: 1.0
            value: 3.0
          }
          attributes {
            key: "Namespace"
            value {
              string_value: "AWS/DynamoDB"
            }
          }
          attributes {
            key: "MetricName"
            value {
              string_value: "ConsumedReadCapacityUnits"
            }
          }
          attributes {
            key: "Dimensions"
            value {
              kvlist_value {
```

```
                values {
                  key: "TableName"
                  value {
                    string_value: "MyTable"
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
```

Objek tingkat atas untuk membuat serial data metrik OpenTelemetry

`ExportMetricsServiceRequest`adalah pembungkus tingkat atas untuk membuat serial muatan eksportir. OpenTelemetry Itu berisikan satu atau beberapa `ResourceMetrics`.

```
message ExportMetricsServiceRequest {
  // An array of ResourceMetrics.
  // For data coming from a single resource this array will typically contain one
  // element. Intermediary nodes (such as OpenTelemetry Collector) that receive
  // data from multiple origins typically batch the data before forwarding further and
  // in that case this array will contain multiple elements.
  repeated opentelemetry.proto.metrics.v1.ResourceMetrics resource_metrics = 1;
}
```

`ResourceMetrics`adalah objek tingkat atas untuk mewakili MetricData objek.

```
// A collection of ScopeMetrics from a Resource.
message ResourceMetrics {
  reserved 1000;

  // The resource for the metrics in this message.
  // If this field is not set then no resource info is known.
  opentelemetry.proto.resource.v1.Resource resource = 1;

  // A list of metrics that originate from a resource.
  repeated ScopeMetrics scope_metrics = 2;

  // This schema_url applies to the data in the "resource" field. It does not apply
```

```
  // to the data in the "scope_metrics" field which have their own schema_url field.
  string schema_url = 3;
}
```

objek Sumber Daya

Objek `Resource` adalah sebuah objek pasangan-nilai yang berisikan beberapa informasi tentang sumber daya yang dihasilkan metrik. Untuk metrik yang dibuat oleh AWS, struktur data yang berisikan Amazon Resource Name (ARN) dari sumber daya yang berkaitan dengan metrik, seperti instans EC2 atau bucket S3.

Objek `Resource` berisikan atribut yang disebut `attributes`, yang menyimpan daftar pasangan nilai-kunci.

- `cloud.account.id` berisi ID akun
- `cloud.region` berisi Wilayah
- `aws.exporter.arn` berisi ARN stream metrik
- `cloud.provider` selalu `aws`.

```
// Resource information.
message Resource {
  // Set of attributes that describe the resource.
  // Attribute keys MUST be unique (it is not allowed to have more than one
  // attribute with the same key).
  repeated opentelemetry.proto.common.v1.KeyValue attributes = 1;

  // dropped_attributes_count is the number of dropped attributes. If the value is 0,
 then
  // no attributes were dropped.
  uint32 dropped_attributes_count = 2;
}
```

ScopeMetrics Objeknya

Bidang `scope` tidak akan diisi. Kami hanya akan mengisi bidang metrik yang kami ekspor.

```
// A collection of Metrics produced by an Scope.
message ScopeMetrics {
  // The instrumentation scope information for the metrics in this message.
  // Semantically when InstrumentationScope isn't set, it is equivalent with
```

```
  // an empty instrumentation scope name (unknown).
  opentelemetry.proto.common.v1.InstrumentationScope scope = 1;

  // A list of metrics that originate from an instrumentation library.
  repeated Metric metrics = 2;

  // This schema_url applies to all metrics in the "metrics" field.
  string schema_url = 3;
}
```

Objek metrik

Objek metrik berisikan beberapa metadata dan kolom data `Summary` yang berisi sebuah daftar `SummaryDataPoint`.

Untuk stream metrik, metadatanya adalah sebagai berikut:

- `name` adalah `amazonaws.com/`*`metric_namespace`*`/`*`metric_name`*
- `description` akan menjadi kosong
- `unit` akan diisi dengan pemetaan unit datum metrik ke varian yang sensitif huruf besar-kecil dari kode Terpadu untuk Unit Ukur. Untuk informasi selengkapnya, silakan lihat Terjemahan dengan format OpenTelemetry 1.0.0 dan Kode Terpadu Untuk Unit Ukur.
- `type` adalah `SUMMARY`

```
message Metric {
  reserved 4, 6, 8;

  // name of the metric, including its DNS name prefix. It must be unique.
  string name = 1;

  // description of the metric, which can be used in documentation.
  string description = 2;

  // unit in which the metric value is reported. Follows the format
  // described by http://unitsofmeasure.org/ucum.html.
  string unit = 3;

  // Data determines the aggregation type (if any) of the metric, what is the
  // reported value type for the data points, as well as the relatationship to
  // the time interval over which they are reported.
  oneof data {
```

```
    Gauge gauge = 5;
    Sum sum = 7;
    Histogram histogram = 9;
    ExponentialHistogram exponential_histogram = 10;
    Summary summary = 11;
  }
}

message Summary {
  repeated SummaryDataPoint data_points = 1;
}
```

SummaryDataPoint Objeknya

SummaryDataPoint Objek berisi nilai titik data tunggal dalam deret waktu dalam DoubleSummary metrik.

```
// SummaryDataPoint is a single data point in a timeseries that describes the
// time-varying values of a Summary metric.
message SummaryDataPoint {
  reserved 1;

  // The set of key/value pairs that uniquely identify the timeseries from
  // where this point belongs. The list may be empty (may contain 0 elements).
  // Attribute keys MUST be unique (it is not allowed to have more than one
  // attribute with the same key).
  repeated opentelemetry.proto.common.v1.KeyValue attributes = 7;

  // StartTimeUnixNano is optional but strongly encouraged, see the
  // the detailed comments above Metric.
  //
  // Value is UNIX Epoch time in nanoseconds since 00:00:00 UTC on 1 January
  // 1970.
  fixed64 start_time_unix_nano = 2;

  // TimeUnixNano is required, see the detailed comments above Metric.
  //
  // Value is UNIX Epoch time in nanoseconds since 00:00:00 UTC on 1 January
  // 1970.
  fixed64 time_unix_nano = 3;

  // count is the number of values in the population. Must be non-negative.
  fixed64 count = 4;
```

```
  // sum of the values in the population. If count is zero then this field
  // must be zero.
  //
  // Note: Sum should only be filled out when measuring non-negative discrete
  // events, and is assumed to be monotonic over the values of these events.
  // Negative events *can* be recorded, but sum should not be filled out when
  // doing so.  This is specifically to enforce compatibility w/ OpenMetrics,
  // see: https://github.com/OpenObservability/OpenMetrics/blob/main/specification/
OpenMetrics.md#summary
  double sum = 5;

  // Represents the value at a given quantile of a distribution.
  //
  // To record Min and Max values following conventions are used:
  // - The 1.0 quantile is equivalent to the maximum value observed.
  // - The 0.0 quantile is equivalent to the minimum value observed.
  //
  // See the following issue for more context:
  // https://github.com/open-telemetry/opentelemetry-proto/issues/125
  message ValueAtQuantile {
    // The quantile of a distribution. Must be in the interval
    // [0.0, 1.0].
    double quantile = 1;

    // The value at the given quantile of a distribution.
    //
    // Quantile values must NOT be negative.
    double value = 2;
  }

  // (Optional) list of values at different quantiles of the distribution calculated
  // from the current snapshot. The quantiles must be strictly increasing.
  repeated ValueAtQuantile quantile_values = 6;

  // Flags that apply to this specific data point.  See DataPointFlags
  // for the available flags and their meaning.
  uint32 flags = 8;
}
```

Untuk informasi selengkapnya, lihat Terjemahan dengan format OpenTelemetry 1.0.0.

Terjemahan dengan format OpenTelemetry 1.0.0

CloudWatch melakukan beberapa transformasi untuk memasukkan CloudWatch data ke dalam OpenTelemetry format.

Menerjemahkan namespace, nama metrik, dan dimensi

Atribut ini adalah pasangan nilai-kunci yang dikodekan ke dalam pemetaan.

- Satu atribut memiliki kunci `Namespace` dan nilainya adalah namespace dari metrik tersebut
- Satu atribut memiliki kunci `MetricName` dan nilainya adalah nama dari metrik tersebut
- Satu pasangan memiliki kunci `Dimensions` dan nilainya adalah sebuah daftar pasangan nilai-kunci tersarang. Setiap pasangan dalam daftar ini memetakan ke dimensi CloudWatch metrik, di mana kunci pasangan adalah nama dimensi dan nilainya adalah nilai dimensi.

Menerjemahkan Rata-rata, Jumlah, SampleCount, Min dan Maks

Titik data Ringkasan memungkinkan CloudWatch untuk mengekspor semua statistik ini menggunakan satu titik data.

- `startTimeUnixNano`berisi CloudWatch `startTime`
- `timeUnixNano`berisi CloudWatch `endTime`
- `sum` berisi statistik Jumlah.
- `count`berisi SampleCount statistik.
- `quantile_values` berisi dua objek `valueAtQuantile.value`:
  - `valueAtQuantile.quantile = 0.0` dengan `valueAtQuantile.value = *Min value*`
  - `valueAtQuantile.quantile = 0.99` dengan `valueAtQuantile.value = *p99 value*`
  - `valueAtQuantile.quantile = 0.999` dengan `valueAtQuantile.value = *p99.9 value*`
  - `valueAtQuantile.quantile = 1.0` dengan `valueAtQuantile.value = *Max value*`

Sumber daya yang menggunakan aliran metrik dapat menghitung statistik Rata-rata sebagai Jumlah/SampleCount.

Menerjemahkan unit-unit

CloudWatch unit dipetakan ke varian peka huruf besar/kecil dari kode Terpadu untuk Satuan Ukuran, seperti yang ditunjukkan pada tabel berikut. Untuk informasi selengkapnya, silakan lihat Kode Terpadu Untuk Unit Ukur.

CloudWatch	OpenTelemetry
Detik	d
Detik atau Detik	detik
Mikrodetik	µs
Milidetik	ms
Byte	Oleh
Kilobyte	kBy
Megabyte	MBy
Gigabyte	GBy
Terabyte	TBy
Bit	bit
Kilobit	kbit
Megabit	MBit
Gigabit	GBit
Terabit	TBit
Persen	%
Hitungan	{Count}
Tidak ada	1

Unit yang digabungkan dengan garis miring dipetakan dengan menerapkan OpenTelemetry konversi kedua unit. Sebagai contoh, Byte/Detik dipetakan ke By/s.

Cara mengurai pesan OpenTelemetry 1.0.0

Bagian ini memberikan informasi untuk membantu Anda memulai dengan parsing OpenTelemetry 1.0.0.

Pertama, Anda harus mendapatkan binding khusus bahasa, yang memungkinkan Anda mengurai pesan OpenTelemetry 1.0.0 dalam bahasa pilihan Anda.

Untuk mendapatkan ikatan bahasa tertentu

- Langkah-langkahnya tergantung pada bahasa pilihan Anda.
  - Untuk menggunakan Java, tambahkan dependensi Maven berikut ke proyek Java Anda: Java >> 0.14.1OpenTelemetry .
  - Untuk menggunakan bahasa lain, ikuti langkah-langkah ini:

    a. Pastikan bahasa Anda didukung dengan memeriksa daftar di Menghasilkan Kelas Anda.

    b. Instal alat kompilasi Protobuf dengan mengikuti langkah-langkah di Unduh Buffer Protokol.

    c. Unduh ProtoBuf definisi OpenTelemetry 0.7.0 di Rilis versi 1.0.0.

    d. Konfirmasikan bahwa Anda berada di folder root dari definisi OpenTelemetry 0.7.0 ProtoBuf yang diunduh. Kemudian buatlah folder `src` lalu jalankan perintah untuk menghasilkan ikatan bahasa khusus. Untuk informasi selengkapnya, silakan lihat Menghasilkan Kelas Anda.

    Berikut ini adalah contoh untuk cara menghasilkan ikatan Javascript.

```
protoc --proto_path=./ --js_out=import_style=commonjs,binary:src \
opentelemetry/proto/common/v1/common.proto \
opentelemetry/proto/resource/v1/resource.proto \
opentelemetry/proto/metrics/v1/metrics.proto \
opentelemetry/proto/collector/metrics/v1/metrics_service.proto
```

Bagian berikut mencakup contoh-contoh menggunakan ikatan bahasa khusus yang dapat Anda bangun dengan menggunakan petunjuk sebelumnya.

Java

```
package com.example;

import io.opentelemetry.proto.collector.metrics.v1.ExportMetricsServiceRequest;

import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

public class MyOpenTelemetryParser {

    public List<ExportMetricsServiceRequest> parse(InputStream inputStream) throws
 IOException {
        List<ExportMetricsServiceRequest> result = new ArrayList<>();

        ExportMetricsServiceRequest request;
        /* A Kinesis record can contain multiple `ExportMetricsServiceRequest`
           records, each of them starting with a header with an
           UnsignedVarInt32 indicating the record length in bytes:
            ------ -------------------------- ------ ----------------------
           |UINT32|ExportMetricsServiceRequest|UINT32|ExportMetricsService...
            ------ -------------------------- ------ ----------------------
         */
        while ((request =
 ExportMetricsServiceRequest.parseDelimitedFrom(inputStream)) != null) {
            // Do whatever we want with the parsed message
            result.add(request);
        }

        return result;
    }
}
```

Javascript

Contoh ini mengasumsikan bahwa folder akar dengan ikatan yang dihasilkan adalah `./`

Argumen data dari fungsi `parseRecord` dapat menjadi salah satu dari jenis berikut:

- `Uint8Array` ini adalah optimal
- `Buffer` menjadi optimal di bawah simpul

- `Array.`*`number`* adalah bilangan bulat 8-bit

```
const pb = require('google-protobuf')
const pbMetrics =
    require('./opentelemetry/proto/collector/metrics/v1/metrics_service_pb')

function parseRecord(data) {
    const result = []

    // Loop until we've read all the data from the buffer
    while (data.length) {
        /* A Kinesis record can contain multiple `ExportMetricsServiceRequest`
           records, each of them starting with a header with an
           UnsignedVarInt32 indicating the record length in bytes:
            ------ -------------------------- ------ ----------------------
           |UINT32|ExportMetricsServiceRequest|UINT32|ExportMetricsService...
            ------ -------------------------- ------ ----------------------
         */
        const reader = new pb.BinaryReader(data)
        const messageLength = reader.decoder_.readUnsignedVarint32()
        const messageFrom = reader.decoder_.cursor_
        const messageTo = messageFrom + messageLength

        // Extract the current `ExportMetricsServiceRequest` message to parse
        const message = data.subarray(messageFrom, messageTo)

        // Parse the current message using the ProtoBuf library
        const parsed =
            pbMetrics.ExportMetricsServiceRequest.deserializeBinary(message)

        // Do whatever we want with the parsed message
        result.push(parsed.toObject())

        // Shrink the remaining buffer, removing the already parsed data
        data = data.subarray(messageTo)
    }

    return result
}
```

Python

Anda harus membaca anti pembatas `var-int` oleh Anda sendiri atau gunakan metode internal `_VarintBytes(size)` dan `_DecodeVarint32(buffer, position)`. Hal ini akan mengembalikan posisi dalam buffer langsung menyusul ukuran bita. Bagian baca akan mengonstruksi buffer baru yang terbatas pada pembacaan byte pesan saja.

```
size = my_metric.ByteSize()
f.write(_VarintBytes(size))
f.write(my_metric.SerializeToString())
msg_len, new_pos = _DecodeVarint32(buf, 0)
msg_buf = buf[new_pos:new_pos+msg_len]
request = metrics_service_pb.ExportMetricsServiceRequest()
request.ParseFromString(msg_buf)
```

Kunjungi

Gunakan `Buffer.DecodeMessage()`.

C#

Gunakan `CodedInputStream`. Kelas ini dapat membaca pesan dengan ukuran tak terbatas.

C++

Fungsi yang dijelaskan dalam `google/protobuf/util/delimited_message_util.h` dapat membaca pesan yang dibatasi ukuran.

Bahasa lain

Untuk bahasa lain, silakan lihat Unduh Protokol Buffer.

Saat menerapkan alat pengurai, pertimbangkan bahwa sebuah catatan Kinesis dapat berisikan beberapa pesan Protokol Buffer `ExportMetricsServiceRequest` , masing-masing dimulai dengan sebuah header dengan sebuah `UnsignedVarInt32` yang menunjukkan panjang catatan dalam bita.

OpenTelemetry Format 0.7.0

OpenTelemetry adalah kumpulan alat, API, dan SDK. Anda dapat menggunakannya untuk instrumen, menghasilkan, mengumpulkan, dan mengekspor data telemetri (metrik, log, dan jejak) untuk analisis. OpenTelemetry adalah bagian dari Cloud Native Computing Foundation. Untuk informasi lebih lanjut, lihat OpenTelemetry.

Untuk informasi tentang spesifikasi OpenTelemetry 0.7.0 lengkap, lihat rilis v0.7.0.

Catatan Kinesis dapat berisi satu atau lebih struktur `ExportMetricsServiceRequest` OpenTelemetry data. Setiap struktur data dimulai dengan header dengan sebuah `UnsignedVarInt32` yang mengindikasikan panjang catatan dalam Byte. Setiap `ExportMetricsServiceRequest` dapat berisikan data dari beberapa metrik sekaligus.

Berikut ini adalah representasi string dari pesan struktur `ExportMetricsServiceRequest` OpenTelemetry data. OpenTelemetry membuat serial protokol biner Google Protocol Buffers, dan ini tidak dapat dibaca manusia.

```
resource_metrics {
  resource {
    attributes {
      key: "cloud.provider"
      value {
        string_value: "aws"
      }
    }
    attributes {
      key: "cloud.account.id"
      value {
        string_value: "2345678901"
      }
    }
    attributes {
      key: "cloud.region"
      value {
        string_value: "us-east-1"
      }
    }
    attributes {
      key: "aws.exporter.arn"
      value {
        string_value: "arn:aws:cloudwatch:us-east-1:123456789012:metric-stream/
MyMetricStream"
      }
    }
  }
  instrumentation_library_metrics {
    metrics {
      name: "amazonaws.com/AWS/DynamoDB/ConsumedReadCapacityUnits"
      unit: "1"
      double_summary {
        data_points {
```

```
          labels {
            key: "Namespace"
            value: "AWS/DynamoDB"
          }
          labels {
            key: "MetricName"
            value: "ConsumedReadCapacityUnits"
          }
          labels {
            key: "TableName"
            value: "MyTable"
          }
          start_time_unix_nano: 1604948400000000000
          time_unix_nano: 1604948460000000000
          count: 1
          sum: 1.0
          quantile_values {
            quantile: 0.0
            value: 1.0
          }
          quantile_values {
            quantile: 0.95
            value: 1.0
          }
          quantile_values {
            quantile: 0.99
            value: 1.0
          }
          quantile_values {
            quantile: 1.0
            value: 1.0
          }
        }
        data_points {
          labels {
            key: "Namespace"
            value: "AWS/DynamoDB"
          }
          labels {
            key: "MetricName"
            value: "ConsumedReadCapacityUnits"
          }
          labels {
            key: "TableName"
```

```
            value: "MyTable"
          }
          start_time_unix_nano: 1604948460000000000
          time_unix_nano: 1604948520000000000
          count: 2
          sum: 5.0
          quantile_values {
            quantile: 0.0
            value: 2.0
          }
          quantile_values {
            quantile: 1.0
            value: 3.0
          }
        }
      }
    }
  }
}
```

Objek tingkat atas untuk membuat serial data metrik OpenTelemetry

`ExportMetricsServiceRequest`adalah pembungkus tingkat atas untuk membuat serial muatan eksportir. OpenTelemetry Itu berisikan satu atau beberapa `ResourceMetrics`.

```
message ExportMetricsServiceRequest {
  // An array of ResourceMetrics.
  // For data coming from a single resource this array will typically contain one
  // element. Intermediary nodes (such as OpenTelemetry Collector) that receive
  // data from multiple origins typically batch the data before forwarding further and
  // in that case this array will contain multiple elements.
  repeated opentelemetry.proto.metrics.v1.ResourceMetrics resource_metrics = 1;
}
```

`ResourceMetrics`adalah objek tingkat atas untuk mewakili MetricData objek.

```
// A collection of InstrumentationLibraryMetrics from a Resource.
message ResourceMetrics {
  // The resource for the metrics in this message.
  // If this field is not set then no resource info is known.
  opentelemetry.proto.resource.v1.Resource resource = 1;
```

```
  // A list of metrics that originate from a resource.
  repeated InstrumentationLibraryMetrics instrumentation_library_metrics = 2;
}
```

objek Sumber Daya

Objek `Resource` adalah sebuah objek pasangan-nilai yang berisikan beberapa informasi tentang sumber daya yang dihasilkan metrik. Untuk metrik yang dibuat oleh AWS, struktur data yang berisikan Amazon Resource Name (ARN) dari sumber daya yang berkaitan dengan metrik, seperti instans EC2 atau bucket S3.

Objek `Resource` berisikan atribut yang disebut `attributes`, yang menyimpan daftar pasangan nilai-kunci.

- `cloud.account.id` berisi ID akun
- `cloud.region` berisi Wilayah
- `aws.exporter.arn` berisi ARN stream metrik
- `cloud.provider` selalu `aws`.

```
// Resource information.
message Resource {
  // Set of labels that describe the resource.
  repeated opentelemetry.proto.common.v1.KeyValue attributes = 1;

  // dropped_attributes_count is the number of dropped attributes. If the value is 0,
  // no attributes were dropped.
  uint32 dropped_attributes_count = 2;
}
```

InstrumentationLibraryMetrics Objeknya

Bidang instrumentation_library tidak akan diisi. Kami hanya akan mengisi bidang metrik yang kami ekspor.

```
// A collection of Metrics produced by an InstrumentationLibrary.
message InstrumentationLibraryMetrics {
  // The instrumentation library information for the metrics in this message.
  // If this field is not set then no library info is known.
  opentelemetry.proto.common.v1.InstrumentationLibrary instrumentation_library = 1;
```

```
  // A list of metrics that originate from an instrumentation library.
  repeated Metric metrics = 2;
}
```

objek Metrik

Objek metrik berisikan kolom data `DoubleSummary` yang berisi daftar `DoubleSummaryDataPoint`.

```
message Metric {
  // name of the metric, including its DNS name prefix. It must be unique.
  string name = 1;

  // description of the metric, which can be used in documentation.
  string description = 2;

  // unit in which the metric value is reported. Follows the format
  // described by http://unitsofmeasure.org/ucum.html.
  string unit = 3;

  oneof data {
    IntGauge int_gauge = 4;
    DoubleGauge double_gauge = 5;
    IntSum int_sum = 6;
    DoubleSum double_sum = 7;
    IntHistogram int_histogram = 8;
    DoubleHistogram double_histogram = 9;
    DoubleSummary double_summary = 11;
  }
}

message DoubleSummary {
  repeated DoubleSummaryDataPoint data_points = 1;
}
```

MetricDescriptor Objeknya

MetricDescriptor Objek berisi metadata. Untuk informasi lebih lanjut, lihat metrics.proto di. GitHub

Untuk aliran metrik, MetricDescriptor memiliki konten berikut:

- `name` akan menjadi `amazonaws.com/`*`metric_namespace`*`/`*`metric_name`*
- `description` akan menjadi kosong.

- `unit` akan diisi dengan pemetaan unit datum metrik ke varian yang sensitif huruf besar-kecil dari kode Terpadu untuk Unit Ukur. Untuk informasi selengkapnya, silakan lihat Terjemahan dengan format OpenTelemetry 0.7.0 dan Kode Terpadu Untuk Unit Ukur.
- `type` akan menjadi `SUMMARY`.

DoubleSummaryDataPoint Objeknya

DoubleSummaryDataPoint Objek berisi nilai titik data tunggal dalam deret waktu dalam DoubleSummary metrik.

```
// DoubleSummaryDataPoint is a single data point in a timeseries that describes the
// time-varying values of a Summary metric.
message DoubleSummaryDataPoint {
  // The set of labels that uniquely identify this timeseries.
  repeated opentelemetry.proto.common.v1.StringKeyValue labels = 1;

  // start_time_unix_nano is the last time when the aggregation value was reset
  // to "zero". For some metric types this is ignored, see data types for more
  // details.
  //
  // The aggregation value is over the time interval (start_time_unix_nano,
  // time_unix_nano].
  //
  // Value is UNIX Epoch time in nanoseconds since 00:00:00 UTC on 1 January
  // 1970.
  //
  // Value of 0 indicates that the timestamp is unspecified. In that case the
  // timestamp may be decided by the backend.
  fixed64 start_time_unix_nano = 2;

  // time_unix_nano is the moment when this aggregation value was reported.
  //
  // Value is UNIX Epoch time in nanoseconds since 00:00:00 UTC on 1 January
  // 1970.
  fixed64 time_unix_nano = 3;

  // count is the number of values in the population. Must be non-negative.
  fixed64 count = 4;

  // sum of the values in the population. If count is zero then this field
  // must be zero.
  double sum = 5;
```

```
  // Represents the value at a given quantile of a distribution.
  //
  // To record Min and Max values following conventions are used:
  // - The 1.0 quantile is equivalent to the maximum value observed.
  // - The 0.0 quantile is equivalent to the minimum value observed.
  message ValueAtQuantile {
    // The quantile of a distribution. Must be in the interval
    // [0.0, 1.0].
    double quantile = 1;

    // The value at the given quantile of a distribution.
    double value = 2;
  }

  // (Optional) list of values at different quantiles of the distribution calculated
  // from the current snapshot. The quantiles must be strictly increasing.
  repeated ValueAtQuantile quantile_values = 6;
}
```

Untuk informasi selengkapnya, lihat Terjemahan dengan format OpenTelemetry 0.7.0.

Terjemahan dengan format OpenTelemetry 0.7.0

CloudWatch melakukan beberapa transformasi untuk memasukkan CloudWatch data ke dalam OpenTelemetry format.

Menerjemahkan namespace, nama metrik, dan dimensi

Atribut ini adalah pasangan nilai-kunci yang dikodekan ke dalam pemetaan.

- Satu pasangan berisikan namespace metrik
- Satu pasangan berisikan name metrik
- Untuk setiap dimensi, CloudWatch simpan pasangan berikut: `metricDatum.Dimensions[i].`*`Name`*`, metricDatum.Dimensions[i].`*`Value`*

Menerjemahkan Rata-rata, Jumlah, SampleCount, Min dan Maks

Titik data Ringkasan memungkinkan CloudWatch untuk mengekspor semua statistik ini menggunakan satu titik data.

- `startTimeUnixNano`berisi CloudWatch `startTime`

- `timeUnixNano` berisi CloudWatch `endTime`
- `sum` berisi statistik Jumlah.
- `count` berisi SampleCount statistik.
- `quantile_values` berisi dua objek `valueAtQuantile.value`:
  - `valueAtQuantile.quantile = 0.0` dengan `valueAtQuantile.value = Min value`
  - `valueAtQuantile.quantile = 0.99` dengan `valueAtQuantile.value = p99 value`
  - `valueAtQuantile.quantile = 0.999` dengan `valueAtQuantile.value = p99.9 value`
  - `valueAtQuantile.quantile = 1.0` dengan `valueAtQuantile.value = Max value`

Sumber daya yang menggunakan aliran metrik dapat menghitung statistik Rata-rata sebagai Jumlah/SampleCount.

Menerjemahkan unit-unit

CloudWatch unit dipetakan ke varian peka huruf besar/kecil dari kode Terpadu untuk Satuan Ukuran, seperti yang ditunjukkan pada tabel berikut. Untuk informasi selengkapnya, silakan lihat Kode Terpadu Untuk Unit Ukur.

CloudWatch	OpenTelemetry
Detik	d
Detik atau Detik	d
Mikrodetik	µs
Milidetik	ms
Byte	Oleh
Kilobyte	kBy
Megabyte	MBy
Gigabyte	GBy
Terabyte	TBy

CloudWatch	OpenTelemetry
Bit	bit
Kilobit	kbit
Megabit	MBit
Gigabit	GBit
Terabit	TBit
Persen	%
Hitungan	{Count}
Tidak ada	1

Unit yang digabungkan dengan garis miring dipetakan dengan menerapkan OpenTelemetry konversi kedua unit. Sebagai contoh, Byte/Detik dipetakan ke By/s.

Cara mengurai pesan OpenTelemetry 0.7.0

Bagian ini memberikan informasi untuk membantu Anda memulai dengan parsing OpenTelemetry 0.7.0.

Pertama, Anda harus mendapatkan binding khusus bahasa, yang memungkinkan Anda mengurai pesan OpenTelemetry 0.7.0 dalam bahasa pilihan Anda.

Untuk mendapatkan ikatan bahasa tertentu

- Langkah-langkahnya tergantung pada bahasa pilihan Anda.
  - Untuk menggunakan Java, tambahkan dependensi Maven berikut ke proyek Java Anda: Java >> 0.14.1OpenTelemetry .
  - Untuk menggunakan bahasa lain, ikuti langkah-langkah ini:

    a. Pastikan bahasa Anda didukung dengan memeriksa daftar di Menghasilkan Kelas Anda.

    b. Instal alat kompilasi Protobuf dengan mengikuti langkah-langkah di Unduh Buffer Protokol.

c. Unduh ProtoBuf definisi OpenTelemetry 0.7.0 pada rilis v0.7.0.

d. Konfirmasikan bahwa Anda berada di folder root dari definisi OpenTelemetry 0.7.0 ProtoBuf yang diunduh. Kemudian buatlah folder `src` lalu jalankan perintah untuk menghasilkan ikatan bahasa khusus. Untuk informasi selengkapnya, silakan lihat Menghasilkan Kelas Anda.

   Berikut ini adalah contoh untuk cara menghasilkan ikatan Javascript.

```
protoc --proto_path=./ --js_out=import_style=commonjs,binary:src \
opentelemetry/proto/common/v1/common.proto \
opentelemetry/proto/resource/v1/resource.proto \
opentelemetry/proto/metrics/v1/metrics.proto \
opentelemetry/proto/collector/metrics/v1/metrics_service.proto
```

Bagian berikut mencakup contoh-contoh menggunakan ikatan bahasa khusus yang dapat Anda bangun dengan menggunakan petunjuk sebelumnya.

Java

```
package com.example;

import io.opentelemetry.proto.collector.metrics.v1.ExportMetricsServiceRequest;

import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

public class MyOpenTelemetryParser {

    public List<ExportMetricsServiceRequest> parse(InputStream inputStream) throws
 IOException {
        List<ExportMetricsServiceRequest> result = new ArrayList<>();

        ExportMetricsServiceRequest request;
        /* A Kinesis record can contain multiple `ExportMetricsServiceRequest`
           records, each of them starting with a header with an
           UnsignedVarInt32 indicating the record length in bytes:
            ------ -------------------------- ------ ----------------------
           |UINT32|ExportMetricsServiceRequest|UINT32|ExportMetricsService...
            ------ -------------------------- ------ ----------------------
```

```
         */
        while ((request =
 ExportMetricsServiceRequest.parseDelimitedFrom(inputStream)) != null) {
            // Do whatever we want with the parsed message
            result.add(request);
        }

        return result;
    }
}
```

Javascript

Contoh ini mengasumsikan bahwa folder akar dengan ikatan yang dihasilkan adalah `./`

Argumen data dari fungsi `parseRecord` dapat menjadi salah satu dari jenis berikut:

- `Uint8Array` ini adalah optimal
- `Buffer` menjadi optimal di bawah simpul
- `Array.`*`number`* adalah bilangan bulat 8-bit

```
const pb = require('google-protobuf')
const pbMetrics =
    require('./opentelemetry/proto/collector/metrics/v1/metrics_service_pb')

function parseRecord(data) {
    const result = []

    // Loop until we've read all the data from the buffer
    while (data.length) {
        /* A Kinesis record can contain multiple `ExportMetricsServiceRequest`
           records, each of them starting with a header with an
           UnsignedVarInt32 indicating the record length in bytes:
            ------ -------------------------- ------ ----------------------
           |UINT32|ExportMetricsServiceRequest|UINT32|ExportMetricsService...
            ------ -------------------------- ------ ----------------------
         */
        const reader = new pb.BinaryReader(data)
        const messageLength = reader.decoder_.readUnsignedVarint32()
        const messageFrom = reader.decoder_.cursor_
        const messageTo = messageFrom + messageLength
```

```
        // Extract the current `ExportMetricsServiceRequest` message to parse
        const message = data.subarray(messageFrom, messageTo)

        // Parse the current message using the ProtoBuf library
        const parsed =
            pbMetrics.ExportMetricsServiceRequest.deserializeBinary(message)

        // Do whatever we want with the parsed message
        result.push(parsed.toObject())

        // Shrink the remaining buffer, removing the already parsed data
        data = data.subarray(messageTo)
    }

    return result
}
```

Python

Anda harus membaca anti pembatas `var-int` oleh Anda sendiri atau gunakan metode internal `_VarintBytes(size)` dan `_DecodeVarint32(buffer, position)`. Hal ini akan mengembalikan posisi dalam buffer langsung menyusul ukuran bita. Bagian baca akan mengonstruksi buffer baru yang terbatas pada pembacaan byte pesan saja.

```
size = my_metric.ByteSize()
f.write(_VarintBytes(size))
f.write(my_metric.SerializeToString())
msg_len, new_pos = _DecodeVarint32(buf, 0)
msg_buf = buf[new_pos:new_pos+msg_len]
request = metrics_service_pb.ExportMetricsServiceRequest()
request.ParseFromString(msg_buf)
```

Kunjungi

Gunakan `Buffer.DecodeMessage()`.

C#

Gunakan `CodedInputStream`. Kelas ini dapat membaca pesan dengan ukuran tak terbatas.

C++

Fungsi yang dijelaskan dalam `google/protobuf/util/delimited_message_util.h` dapat membaca pesan yang dibatasi ukuran.

Bahasa lain

Untuk bahasa lain, silakan lihat Unduh Protokol Buffer.

Saat menerapkan alat pengurai, pertimbangkan bahwa sebuah catatan Kinesis dapat berisikan beberapa pesan Protokol Buffer `ExportMetricsServiceRequest` , masing-masing dimulai dengan sebuah header dengan sebuah `UnsignedVarInt32` yang menunjukkan panjang catatan dalam bita.

Pemecahan Masalah

Jika Anda tidak melihat data metrik di tujuan akhir Anda, periksa hal berikut:

- Pastikan stream metrik dalam status berjalan. Untuk langkah-langkah tentang cara menggunakan CloudWatch konsol untuk melakukan ini, lihatOperasi dan pemeliharaan stream metrik.
- Metrik yang diterbitkan lebih dari dua hari di masa lalu tidak dialirkan. Untuk menentukan apakah metrik tertentu akan dialirkan, buat grafik metrik di CloudWatch konsol dan periksa berapa umur titik data terakhir yang terlihat. Jika lebih dari dua hari di masa lalu, maka itu tidak akan diambil oleh aliran metrik.
- Periksa metrik yang dihasilkan oleh stream metrik. Di CloudWatch konsol, di bawah Metrik, lihat MetricStreams namespace AWS/CloudWatch/untuk metrik,, dan MetricUpdate. TotalMetricUpdatePublishErrorRate
- Jika PublishErrorRatemetrik tinggi, konfirmasikan bahwa tujuan yang digunakan oleh aliran pengiriman Firehose ada dan peran IAM yang ditentukan dalam konfigurasi aliran metrik memberikan izin utama `CloudWatch` layanan untuk menulis ke metrik. Untuk informasi selengkapnya, lihat Kepercayaan antara CloudWatch dan Firehose.
- Periksa apakah aliran pengiriman Firehose memiliki izin untuk menulis ke tujuan akhir.
- Di Firehose console, lihat aliran pengiriman Firehose yang digunakan untuk aliran metrik dan periksa tab Monitoring untuk melihat apakah aliran pengiriman Firehose menerima data.
- Konfirmasikan bahwa Anda telah mengonfigurasi aliran pengiriman Firehose dengan detail yang benar.
- Periksa log atau metrik yang tersedia untuk tujuan akhir yang dituliskan oleh aliran pengiriman Firehose.

- Untuk mendapatkan informasi lebih rinci, aktifkan CloudWatch log kesalahan Log pada aliran pengiriman Firehose. Untuk informasi selengkapnya, lihat Memantau Amazon Data Firehose Menggunakan CloudWatch Log.

Lihat metrik yang tersedia

Metrik dikelompokkan berdasarkan namespace terlebih dahulu, lalu berdasarkan kombinasi dimensi dalam setiap namespace. Misalnya, Anda dapat melihat semua metrik EC2, metrik EC2 yang dikelompokkan berdasarkan instans, atau metrik EC2 yang dikelompokkan berdasarkan grup Auto Scaling.

Hanya AWS layanan yang Anda gunakan mengirim metrik ke Amazon CloudWatch.

Untuk daftar AWS layanan yang mengirim metrik CloudWatch, lihatAWS layanan yang mempublikasikan CloudWatch metrik. Dari halaman ini, Anda juga dapat melihat metrik dan dimensi yang dipublikasikan oleh masing-masing layanan tersebut.

> **Note**
>
> Metrik yang tidak memiliki titik data baru dalam dua minggu terakhir tidak muncul di konsol. Mereka juga tidak muncul ketika Anda mengetik nama metrik atau nama dimensi mereka di kotak pencarian dalam tab Semua metrik di konsol, dan tidak dikembalikan di dalam hasil dari perintah list-metrics. Cara terbaik untuk mengambil metrik ini adalah dengan get-metric-statisticsperintah get-metric-dataatau di. AWS CLI
> Jika metrik lama yang ingin Anda lihat memiliki metrik saat ini dengan dimensi serupa, Anda dapat melihat metrik serupa saat ini dan kemudian memilih tab Sumber, dan ubah nama metrik dan bidang dimensi ke yang diinginkan, dan juga ubah rentang waktu menjadi waktu ketika metrik dilaporkan.

Langkah-langkah berikut membantu Anda menelusuri namespace metrik untuk menemukan dan melihat metrik. Anda juga dapat mencari metrik menggunakan istilah pencarian bertarget. Untuk informasi selengkapnya, lihat Cari metrik yang tersedia.

Jika Anda menjelajah di akun yang disiapkan sebagai akun pemantauan dalam pengamatan CloudWatch lintas akun, Anda dapat melihat metrik dari akun sumber yang ditautkan ke akun pemantauan ini. Saat metrik dari akun sumber ditampilkan, ID atau label akun tempat mereka berasal juga ditampilkan. Untuk informasi selengkapnya, lihat CloudWatch observabilitas lintas akun.

Untuk melihat metrik yang tersedia berdasarkan namespace dan dimensi menggunakan konsol

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik, Semua metrik.
3. Pilih namespace metrik (misalnya, EC2 atau Lambda).
4. Pilih dimensi metrik (misalnya, Metrik Per-Instans) atau Berdasarkan Nama Fungsi).
5. Tab Jelajahi menampilkan semua metrik untuk dimensi pada namespace. Dengan setiap nama metrik adalah tombol informasi Anda dapat memilih untuk melihat popup dengan definisi metrik.

   Jika ini adalah akun pemantauan dalam pengamatan CloudWatch lintas akun, Anda juga melihat metrik dari akun sumber yang ditautkan ke akun pemantauan ini. Label Akun dan kolom id Akun dalam tabel menampilkan akun mana setiap metrik berasal.

   Anda dapat melakukan hal berikut:

   a. Untuk menyortir tabel, gunakan judul kolomnya.
   b. Untuk membuat grafik sebuah metrik, pilih kotak centang di sebelah metrik. Untuk memilih semua metrik, pilih kotak centang di baris judul tabel.
   c. Untuk memfilter berdasarkan akun, pilih label akun atau ID akun, lalu pilih Tambahkan ke pencarian.
   d. Untuk menyaring berdasarkan sumber daya, pilih ID sumber daya, kemudian pilih Tambahkan ke pencarian.
   e. Untuk menyaring berdasarkan metrik, pilih nama metrik, kemudian pilih Tambahkan ke pencarian.
6. (Opsional) Untuk menambahkan grafik ini ke CloudWatch dasbor, pilih Tindakan, Tambahkan ke dasbor.

Untuk melihat metrik yang tersedia berdasarkan namespace akun, dimensi, atau metrik menggunakan AWS CLI

Gunakan perintah list-metrics untuk membuat daftar metrik. CloudWatch Untuk daftar namespace, metrik, dan dimensi pada semua layanan yang menerbitkan metrik, silakan lihat AWS layanan yang mempublikasikan CloudWatch metrik.

Contoh perintah berikut mencantumkan semua metrik untuk Amazon EC2.

```
aws cloudwatch list-metrics --namespace AWS/EC2
```

Berikut ini adalah output contoh.

```
{
  "Metrics" : [
    ...
    {
        "Namespace": "AWS/EC2",
        "Dimensions": [
            {
                "Name": "InstanceId",
                "Value": "i-1234567890abcdef0"
            }
        ],
        "MetricName": "NetworkOut"
    },
    {
        "Namespace": "AWS/EC2",
        "Dimensions": [
            {
                "Name": "InstanceId",
                "Value": "i-1234567890abcdef0"
            }
        ],
        "MetricName": "CPUUtilization"
    },
    {
        "Namespace": "AWS/EC2",
        "Dimensions": [
            {
                "Name": "InstanceId",
                "Value": "i-1234567890abcdef0"
            }
        ],
        "MetricName": "NetworkIn"
    },
    ...
  ]
}
```

Menyebutkan semua metrik yang tersedia untuk sumber daya tertentu

Contoh berikut menentukan namespace `AWS/EC2` dan dimensi `InstanceId` untuk melihat hasil hanya pada instans yang ditentukan.

```
aws cloudwatch list-metrics --namespace AWS/EC2 --dimensions
 Name=InstanceId,Value=i-1234567890abcdef0
```

Untuk mencantumkan metrik pada semua sumber daya

Contoh berikut menentukan namespace `AWS/EC2` dan nama metrik untuk melihat hasil hanya pada metrik yang ditentukan.

```
aws cloudwatch list-metrics --namespace AWS/EC2 --metric-name CPUUtilization
```

Untuk mengambil metrik dari akun sumber tertaut dalam CloudWatch observabilitas lintas akun

Contoh berikut dijalankan di akun pemantauan untuk mengambil metrik dari akun pemantauan dan semua akun sumber terkait. Jika Anda tidak menambahkan `--include-linked-accounts`, perintah hanya mengembalikan metrik akun pemantauan.

```
aws cloudwatch list-metrics --include-linked-accounts
```

Untuk mengambil metrik dari akun sumber dalam CloudWatch observabilitas lintas akun

Contoh berikut dijalankan di akun pemantauan untuk mengambil metrik dari akun sumber dengan ID 111122223333.

```
aws cloudwatch list-metrics --include-linked-accounts --owning-account "111122223333"
```

Cari metrik yang tersedia

Anda dapat mencari di dalam semua metrik pada akun Anda menggunakan istilah pencarian yang ditargetkan. Metrik dikembalikan dengan hasil yang sesuai dalam namespace, nama metrik, atau dimensi mereka.

Jika ini adalah akun pemantauan dalam pengamatan CloudWatch lintas akun, Anda juga mencari metrik dari akun sumber yang ditautkan ke akun pemantauan ini.

> **Note**
>
> Metrik yang tidak memiliki titik data baru dalam dua minggu terakhir tidak muncul di konsol. Mereka juga tidak muncul ketika Anda mengetik nama metrik atau nama dimensi mereka di kotak pencarian dalam tab Semua metrik di konsol, dan tidak dikembalikan di dalam hasil dari perintah list-metrics. Cara terbaik untuk mengambil metrik ini adalah dengan get-metric-statisticsperintah get-metric-dataatau di. AWS CLI

Untuk mencari metrik yang tersedia di CloudWatch

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik.
3. Di bidang pencarian pada tab Semua metrik, masukkan istilah pencarian, seperti nama metrik, namespace, ID akun, label akun, nama atau nilai dimensi, atau nama sumber. Hal ini menampilkan semua namespace dengan metrik dengan istilah pencarian ini.

   Sebagai contoh, jika Anda mencari **`volume`**, ini menunjukkan namespace yang memuat metrik dengan istilah ini di namanya.

   Untuk informasi selengkapnya tentang pencarian, silakan lihat Gunakan ekspresi pencarian pada grafik
4. Untuk membuat grafik semua hasil pencarian, pilih Buat grafik pencarian

   atau

   Pilih namespace untuk melihat metrik dari namespace tersebut. Anda kemudian dapat melakukan hal berikut:

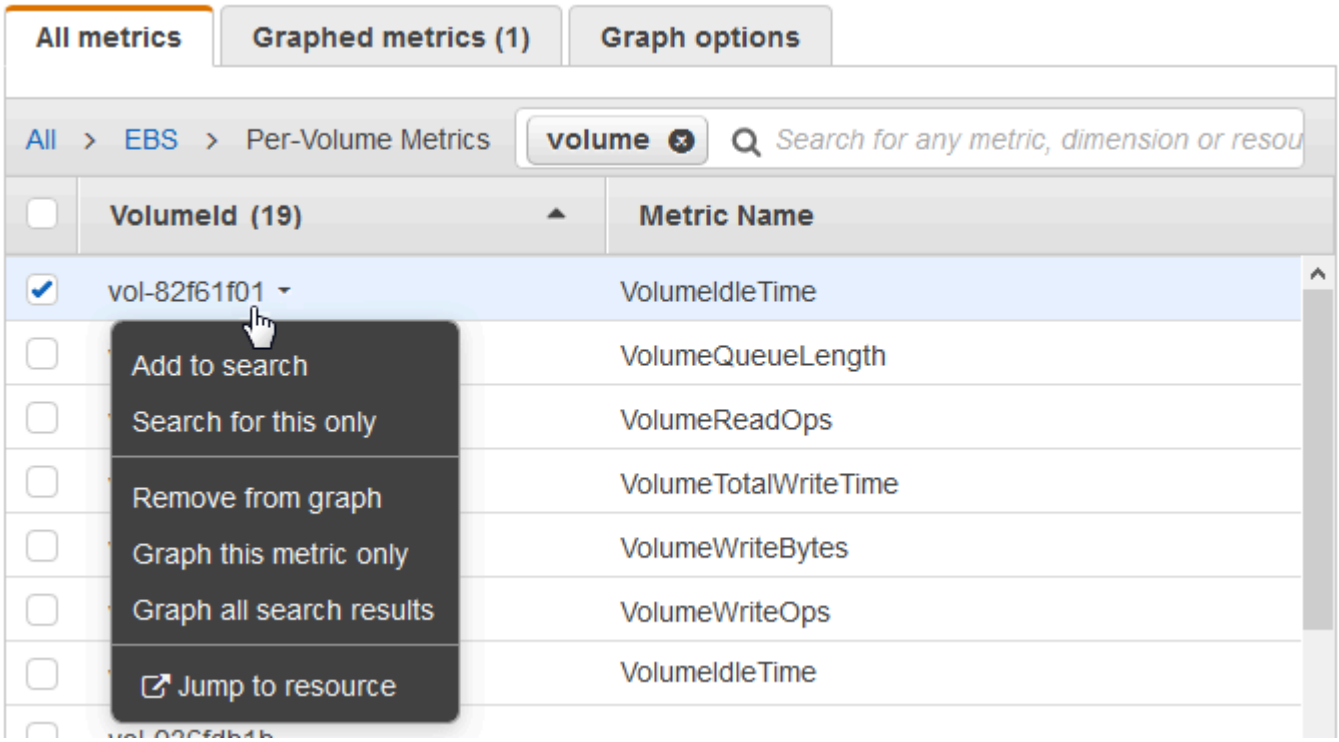
   a. Untuk membuat grafik satu metrik atau lebih, pilih kotak centang di sebelah setiap metrik. Untuk memilih semua metrik, pilih kotak centang di baris judul tabel.

   b. Untuk menyempurnakan pencarian Anda, arahkan kursor ke nama metrik dan pilih Tambahkan ke pencarian atau Cari ini saja.

   c. Untuk melihat salah satu sumber daya pada konsolnya, pilih ID sumber daya dan kemudian pilih Lompat ke sumber daya.

   d. Untuk melihat bantuan metrik, pilih nama metrik dan pilih Apa ini?.

Metrik yang dipilih muncul di grafik.



5. (Opsional) Pilih salah satu tombol di bilah pencarian untuk menyunting bagian dari istilah pencarian tersebut.

Membuat grafik metrik

Gunakan CloudWatch konsol untuk membuat grafik data metrik yang dihasilkan oleh AWS layanan lain. Hal ini membuatnya lebih efisien untuk melihat aktivitas metrik pada layanan Anda. Prosedur berikut menjelaskan cara membuat grafik metrik. CloudWatch

Daftar Isi

Membuat sebuah grafik metrik

Anda dapat memilih metrik dan membuat grafik data metrik menggunakan konsol. CloudWatch

CloudWatch mendukung statistik metrik berikut:`Average`,,`Minimum`, `MaximumSum`, dan`SampleCount`. Untuk informasi selengkapnya, lihat Statistik.

Anda dapat melihat data pada berbagai tingkat detail. Misalnya, Anda dapat memilih tampilan satu menit, yang dapat berguna ketika pemecahan masalah. Atau, pilih tampilan satu jam yang kurang terperinci. Hal ini dapat berguna ketika melihat rentang waktu yang lebih luas (misalnya, 3 hari) sehingga Anda dapat melihat tren dari waktu ke waktu. Untuk informasi selengkapnya, lihat Periode.

Jika Anda menggunakan akun yang disiapkan sebagai akun pemantauan dalam pengamatan CloudWatch lintas akun, Anda dapat membuat grafik metrik dari akun sumber yang ditautkan ke akun pemantauan ini. Untuk informasi selengkapnya, lihat CloudWatch observabilitas lintas akun.

Membuat grafik

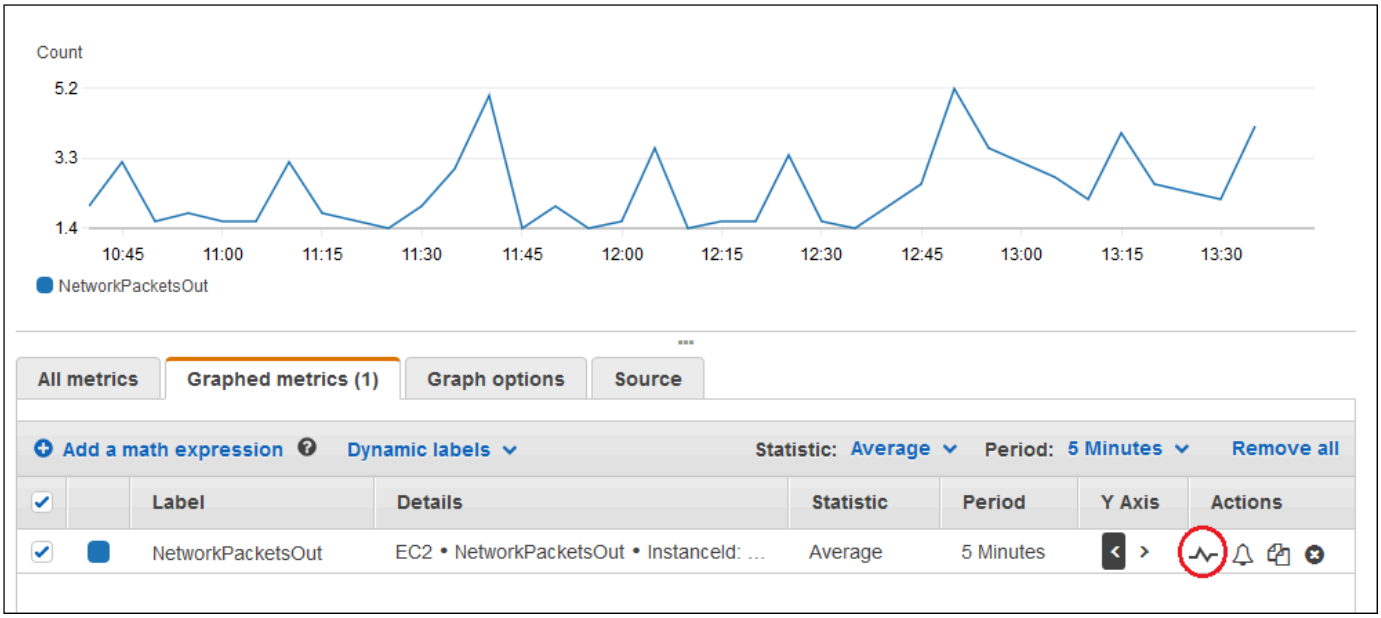
Cara membuat sebuah grafik metrik

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik, Semua metrik.
3. Di tab Jelajahi, masukkan istilah pencarian di bidang pencarian, seperti nama metrik, ID akun, atau nama sumber daya.

   Sebagai contoh, jika mencari metrik `CPUUtilization`, Anda melihat namespace dan dimensi dengan metrik ini.
4. Pilih salah satu hasil pencarian Anda untuk melihat metrik.
5. Untuk membuat grafik satu metrik atau lebih, pilih kotak centang di sebelah setiap metrik. Untuk memilih semua metrik, pilih kotak centang di baris judul tabel.
6. (Opsional) Untuk mengubah jenis grafik, pilih tab Opsi. Kemudian, Anda dapat memilih antara grafik garis, diagram wilayah bertumpuk, tampilan angka, pengukur, diagram batang, atau diagram lingkaran.
7. Pilih tab Metrik bergrafik.
8. (Opsional) Untuk mengubah statistik yang digunakan dalam grafik, pilih statistik baru di kolom Statistik di sebelah nama metrik.

Untuk informasi lebih lanjut tentang CloudWatch statistik, lihatCloudWatch definisi statistik. Untuk informasi selengkapnya tentang statistik persentil pxx, silakan lihat Persentil.

9. (Opsional) Untuk menambahkan pita deteksi anomali yang menunjukkan nilai yang diharapkan untuk metrik, pilih ikon deteksi anomali di bawah Tindakan di sebelah metrik.



CloudWatch menggunakan hingga dua minggu dari data historis metrik terbaru untuk menghitung model untuk nilai yang diharapkan. Kemudian menampilkan rentang nilai yang diharapkan ini sebagai pita pada grafik. CloudWatch menambahkan baris baru di bawah metrik untuk menampilkan ekspresi matematika pita deteksi anomali, berlabel ANOMALY_DETECTION_BAND. Jika ada data historis terbaru, Anda segera melihat pita deteksi anomali pratinjau, yang merupakan perkiraan pita deteksi anomali yang dihasilkan oleh model. Diperlukan waktu hingga 15 menit agar pita deteksi anomali aktual muncul.

Secara default, CloudWatch membuat batas atas dan bawah pita nilai yang diharapkan dengan nilai default 2 untuk ambang batas band. Untuk mengubah angka ini, ubah nilai di akhir rumus di bawah Detail untuk pita.

- (Opsional) Pilih Sunting model untuk mengubah cara model deteksi anomali dihitung. Anda dapat mengecualikan periode waktu di masa lalu dan mendatang dari penggunaan dalam pelatihan untuk menghitung model. Penting untuk mengecualikan peristiwa-peristiwa tidak biasa seperti gangguan sistem, deployment, dan hari libur dari data pelatihan. Anda juga dapat menentukan zona waktu yang akan digunakan oleh model untuk perubahan waktu musim panas.

Untuk informasi selengkapnya, lihat Menggunakan CloudWatch deteksi anomali.

Untuk menyembunyikan model dari grafik, hapus tanda centang dari garis dengan fungsi `ANOMALY_DETECTION_BAND` atau pilih ikon `X`. Untuk menghapus model sepenuhnya, pilih Sunting model, Hapus model.

10. (Opsional) Ketika Anda memilih metrik untuk dibuat grafik, tentukan label dinamis agar muncul pada legenda grafik untuk setiap metrik. Label dinamis menampilkan statistik tentang metrik, dan secara otomatis diperbarui ketika dasbor atau grafik disegarkan. Untuk menambahkan label dinamis, pilih Metrik yang digambarkan, Tambahkan label dinamis.

    Secara bawaan, nilai dinamis yang Anda tambahkan ke label muncul di awal label. Anda kemudian dapat memilih nilai Label bagi metrik untuk menyunting label. Untuk informasi selengkapnya, lihat Gunakan label dinamis.

11. Untuk melihat informasi lebih lanjut tentang metrik yang sedang digambarkan, tahan mouse di atas legenda.

12. Keterangan horizontal dapat membantu pengguna grafik untuk melihat secara lebih efisien jika metrik telah meningkat ke tingkat tertentu, atau apakah metrik berada dalam rentang yang telah ditentukan sebelumnya. Untuk menambahkan keterangan horizontal, pilih tab Opsi dan kemudian Tambahkan keterangan horizontal:

    a. Untuk Label, masukkan label untuk keterangan.

    b. Untuk Nilai, masukkan nilai metrik tempat keterangan horizontal muncul.

    c. Untuk Isi, tentukan apakah akan menggunakan bayangan pengisian dengan keterangan ini. Sebagai contoh, pilih `Above` atau `Below` untuk area terkait yang akan diisi. Jika Anda menentukan `Between`, bidang `Value` lainnya muncul, dan area grafik di antara kedua nilai diisi.

    d. Untuk Sumbu, tentukan apakah angka di `Value` mengacu pada metrik terkait dengan sumbu Y kiri atau sumbu Y kanan, jika grafik mencakup beberapa metrik.

    Anda dapat mengubah warna pengisian keterangan dengan memilih kotak warna di kolom kiri keterangan.

Ulangi langkah-langkah ini untuk menambahkan beberapa keterangan horizontal ke grafik yang sama.

Untuk menyembunyikan keterangan, kosongkan kotak centang di kolom kiri untuk keterangan tersebut.

Untuk menghapus keterangan, pilih x dalam kolom Tindakan.

13. Untuk mendapatkan URL grafik, pilih Tindakan, Bagikan. Salin URL untuk menyimpan atau berbagi.
14. Untuk menambahkan grafik Anda ke dasbor, pilih Tindakan, Tambahkan ke dasbor.

Membuat sebuah grafik metrik dari sumber data lain

Anda dapat membuat grafik yang menampilkan sumber daya dari sumber data selain CloudWatch. Untuk informasi selengkapnya mengenai cara membuat koneksi ke sumber data lainnya, silakan lihat Metrik kueri dari sumber data lain.

Cara membuat sebuah grafik metrik dari sumber data lain

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik, Semua metrik.
3. Pilih tab Kueri multi sumber.
4. Untuk Sumber data, silakan pilih sumber data yang ingin Anda gunakan.

   Jika Anda belum membuat koneksi ke sumber data yang Anda inginkan, pilih Buat dan kelola sumber data, lalu pilih Buat dan kelola sumber data. Untuk informasi tentang sisa proses pembuatan sumber data ini, silakan lihat Hubungkan ke sumber data default dengan sebuah pemandu.
5. Pemandu atau editor kueri meminta kepada Anda informasi yang diperlukan untuk kueri. Alur kerjanya akan berbeda untuk masing-masing sumber data, dan akan disesuaikan dengan sumber data. Sebagai contoh, untuk Layanan Terkelola Amazon untuk Prometheus; dan sumber data Prometheus, kotak editor kueri PromQL dengan pembantu kueri akan ditampilkan.
6. Setelah Anda selesai membuat konsep kueri tersebut, silakan pilih Buat grafik kueri.

   Grafik diisi dengan metrik dari kueri.
7. (Opsional) Keterangan horizontal dapat membantu pengguna grafik untuk melihat secara lebih efisien jika metrik telah meningkat ke tingkat tertentu, atau apakah metrik berada dalam rentang yang telah ditentukan sebelumnya. Untuk menambahkan keterangan horizontal, pilih tab Opsi dan kemudian Tambahkan keterangan horizontal:

a. Untuk Label, masukkan label untuk keterangan.

b. Untuk Nilai, masukkan nilai metrik tempat keterangan horizontal muncul.

c. Untuk Isi, tentukan apakah akan menggunakan bayangan pengisian dengan keterangan ini. Sebagai contoh, pilih `Above` atau `Below` untuk area terkait yang akan diisi. Jika Anda menentukan `Between`, bidang `Value` lainnya muncul, dan area grafik di antara kedua nilai diisi.

d. Untuk Sumbu, tentukan apakah angka di `Value` mengacu pada metrik terkait dengan sumbu Y kiri atau sumbu Y kanan, jika grafik mencakup beberapa metrik.

   Anda dapat mengubah warna pengisian keterangan dengan memilih kotak warna di kolom kiri keterangan.

Ulangi langkah-langkah ini untuk menambahkan beberapa keterangan horizontal ke grafik yang sama.

Untuk menyembunyikan keterangan, kosongkan kotak centang di kolom kiri untuk keterangan tersebut.

Untuk menghapus keterangan, pilih x dalam kolom Tindakan.

8. (Opsional) Untuk menambahkan grafik ini ke dasbor, pilih Tindakan, Tambahkan ke dasbor.

Memperbarui grafik

Untuk memperbarui grafik Anda

1. Untuk mengubah nama grafik, pilih ikon pensil.
2. Untuk mengubah rentang waktu, pilih salah satu nilai yang telah ditentukan sebelumnya atau pilih kustom. Untuk informasi selengkapnya, lihat Modifikasi deret waktu atau format zona waktu untuk grafik.
3. Untuk mengubah statistik, pilih tab Metrik bergrafik. Pilih judul kolom atau nilai individu dan kemudian pilih salah satu statistik atau persentil yang telah ditentukan sebelumnya, atau tentukan persentil kustom (misalnya, **`p95.45`**).
4. Untuk mengubah periode, pilih tab Metrik bergrafik. Pilih judul kolom atau nilai individu, lalu pilih nilai yang berbeda.

5. Untuk menambahkan keterangan horizontal, pilih tab Opsi grafik dan kemudian Tambahkan keterangan horizontal:

   a. Untuk Label, masukkan label untuk keterangan.

   b. Untuk Nilai, masukkan nilai metrik tempat keterangan horizontal muncul.

   c. Untuk Isi, tentukan apakah akan menggunakan bayangan pengisian dengan keterangan ini. Sebagai contoh, pilih `Above` atau `Below` untuk area terkait yang akan diisi. Jika Anda menentukan `Between`, bidang `Value` lainnya muncul, dan area grafik di antara kedua nilai diisi.

   d. Untuk Sumbu, tentukan apakah angka di `Value` mengacu pada metrik yang terkait dengan sumbu y kiri atau sumbu y kanan, jika grafik mencakup beberapa metrik.

      Anda dapat mengubah warna pengisian keterangan dengan memilih kotak warna di kolom kiri keterangan.

   Ulangi langkah-langkah ini untuk menambahkan beberapa keterangan horizontal ke grafik yang sama.

   Untuk menyembunyikan keterangan, kosongkan kotak centang di kolom kiri untuk keterangan tersebut.

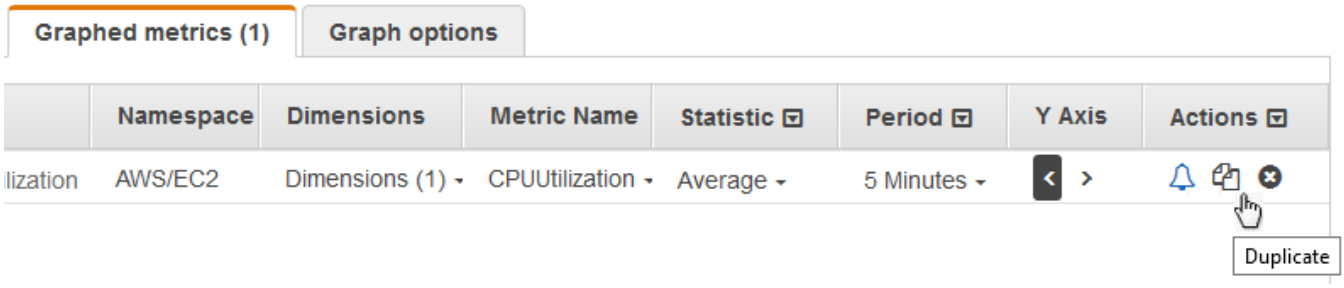
   Untuk menghapus keterangan, pilih x dalam kolom Tindakan.

6. Untuk mengubah interval penyegaran, pilih Segarkan pilihan dan kemudian pilih Segarkan otomatis atau pilih 1 Menit, 2 Menit, 5 Menit, atau 15 Menit.

Duplikasi metrik

Untuk menduplikasi metrik

1. Pilih tab Metrik bergrafik.
2. Untuk Tindakan, pilih ikon Duplikat.



3. Perbarui metrik duplikat sesuai kebutuhan.

Gabungkan dua grafik menjadi satu

Anda dapat menggabungkan dua grafik yang berbeda menjadi satu, dan kemudian grafik yang dihasilkan menunjukkan kedua metrik. Ini dapat berguna jika Anda sudah memiliki metrik berbeda yang ditampilkan dalam grafik yang berbeda dan ingin menggabungkannya, atau Anda ingin membuat grafik tunggal dengan mudah dengan metrik dari Wilayah yang berbeda.

Untuk menggabungkan grafik ke grafik lain, Anda menggunakan URL atau sumber JSON dari grafik yang ingin Anda gabungkan.

Untuk menggabungkan dua grafik menjadi satu

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Buka grafik yang ingin Anda gabungkan ke grafik lain. Untuk melakukan hal itu, Anda dapat memilih Metrik, Semua metrik, dan kemudian memilih metrik untuk grafik. Atau Anda dapat membuka dasbor dan kemudian membuka salah satu grafik di dasbor dengan memilih grafik dan memilih Buka dalam metrik dari menu di kanan atas grafik.
3. Setelah grafik Anda terbuka, lakukan salah satu hal berikut:.
   - Salin URL dari bilah browser.
   - Pilih tab Sumber dan kemudian pilih Salin.
4. Buka grafik yang ingin Anda gabungkan ke grafik sebelumnya.
5. Saat grafik kedua terbuka di tampilan Metrik, pilih Tindakan, Gabungkan grafik.
6. Masukkan URL atau JSON yang sebelumnya Anda salin, dan pilih Gabung.
7. Grafik gabungan muncul. Sumbu y di sebelah kiri untuk grafik asli, dan sumbu y di sebelah kanan untuk grafik yang Anda gabungkan ke dalamnya.

> **Note**
>
> Jika grafik yang Anda gabungkan menggunakan fungsi METRICS (), metrik dalam grafik yang digabungkan tidak termasuk dalam penghitungan METRICS () dalam grafik gabungan.

8. Untuk menyimpan grafik yang digabungkan ke dasbor, pilih Tindakan, Tambahkan ke dasbor.

Gunakan label dinamis

Anda dapat menggunakan label dinamis dengan grafik. Label dinamis menambahkan nilai yang diperbarui secara dinamis ke label untuk metrik yang dipilih. Anda dapat menambahkan berbagai nilai pada label, seperti yang ditunjukkan pada tabel berikut.

Nilai dinamis yang ditunjukkan pada label berasal dari rentang waktu yang saat ini ditunjukkan pada grafik. Bagian dinamis dari label secara otomatis diperbarui ketika dasbor atau grafik disegarkan.

Jika Anda menggunakan label dinamis dengan ekspresi pencarian, label dinamis berlaku untuk setiap metrik yang dikembalikan oleh pencarian.

Anda dapat menggunakan CloudWatch konsol untuk menambahkan nilai dinamis ke label, mengedit label, mengubah posisi nilai dinamis dalam kolom label, dan membuat penyesuaian lainnya.

Label dinamis

Dalam label dinamis, Anda dapat menggunakan nilai berikut yang berkaitan dengan sifat metrik:

Nilai hidup label dinamis	Deskripsi
${AVG}	Nilai rata-rata dalam rentang waktu yang saat ini ditunjukkan dalam grafik.
${DATAPOINT_COUNT}	Jumlah titik data dalam rentang waktu yang saat ini ditunjukkan dalam grafik.
${FIRST}	Nilai metrik tertua dalam rentang waktu yang saat ini ditampilkan dalam grafik.

Nilai hidup label dinamis	Deskripsi
${FIRST_LAST_RANGE}	Perbedaan antara nilai metrik titik data tertua dan terbaru yang saat ini ditampilkan dalam grafik.
${FIRST_LAST_TIME_ RANGE}	Rentang waktu absolut antara titik data tertua dan terbaru yang saat ini ditampilkan dalam grafik.
${FIRST_TIME}	Timestamp titik data tertua dalam rentang waktu yang saat ini ditampilkan dalam grafik.
${FIRST_TIME_RELATIVE}	Perbedaan waktu absolut antara sekarang dan timestamp titik data tertua dalam rentang waktu yang saat ini ditampilkan dalam grafik.
${LABEL}	Representasi label bawaan untuk metrik.
${LAST}	Nilai metrik paling akhir dalam rentang waktu yang saat ini ditampilkan dalam grafik.
$ {LAST_TIME}	Timestamp titik data terbaru dalam rentang waktu yang saat ini ditampilkan dalam grafik.
${LAST_TIME_RELATIVE}	Perbedaan waktu absolut antara sekarang dan timestamp titik data terbaru dalam rentang waktu yang saat ini ditampilkan dalam grafik.
${MAX}	Nilai maksimum dalam rentang waktu yang saat ini ditunjukkan dalam grafik.
${MAX_TIME}	Timestamp titik data yang memiliki nilai metrik tertinggi, titik data yang saat ini ditampilkan dalam grafik.
${MAX_TIME_RELATIVE}	Perbedaan waktu mutlak antara sekarang dan timestamp titik data dengan nilai tertinggi, dari titik data yang saat ini ditampilk an dalam grafik.
${MIN}	Nilai minimum rentang waktu yang saat ini ditunjukkan dalam grafik.

Nilai hidup label dinamis	Deskripsi
${MIN_MAX_RANGE}	Perbedaan nilai metrik antara titik data dengan nilai metrik tertinggi dan terendah, dari titik data yang saat ini ditampilkan dalam grafik.
${MIN_MAX_TIME_RANGE}	Rentang waktu mutlak antara titik data dengan nilai metrik tertinggi dan terendah, dari titik data yang saat ini ditampilkan dalam grafik.
${MIN_TIME}	Timestamp titik data yang memiliki nilai metrik terendah, titik data yang saat ini ditampilkan dalam grafik.
${MIN_TIME_RELATIVE}	Perbedaan waktu mutlak antara sekarang dan timestamp titik data dengan nilai terendah, dari titik data yang saat ini ditampilk an dalam grafik.
$ {PROP ('AccountId')}	ID AWS akun metrik.
$ {PROP ('AccountLabel')}	Label yang ditentukan untuk akun sumber yang memiliki metrik ini, dalam observabiilitas CloudWatch lintas akun.
${PROP('Dim.*dimension_name* ')}	Nilai dimensi yang ditentukan. Ganti *dimension_name* dengan nama yang peka huruf besar dan kecil untuk dimensi Anda.
$ {PROP ('MetricName')}	Nama metrik.
${PROP('Namespace')}	Namespace metrik.
${PROP('Period')}	Periode metrik, dalam detik.
${PROP('Region')}	AWS Wilayah tempat metrik diterbitkan.
${PROP('Stat')}	Statistik metrik yang sedang digambarkan.
${SUM}	Jumlah dari nilai-nilai dalam rentang waktu yang saat ini ditampilkan dalam grafik.

Sebagai contoh, anggap Anda memiliki ekspresi pencarian **`SEARCH(' {AWS/Lambda, FunctionName} Errors ', 'Sum')`**, yang menemukan `Errors` untuk setiap fungsi Lambda. Jika Anda mengatur label menjadi `[max: ${MAX} Errors for Function Name ${LABEL}]`, label untuk setiap metrik adalah [maks: ***nomor*** Kesalahan untuk Nama Fungsi ***Nama***].

Anda dapat menambahkan sebanyak enam nilai dinamis ke label. Anda hanya dapat menggunakan pengganti `${LABEL}` sekali dalam setiap label.

Modifikasi deret waktu atau format zona waktu untuk grafik

Bagian ini menjelaskan bagaimana Anda dapat mengubah format tanggal, waktu, dan zona waktu pada grafik CloudWatch metrik. Ini juga menjelaskan bagaimana Anda dapat memperbesar grafik untuk menerapkan rentang waktu tertentu. Untuk informasi tentang membuat grafik, silakan lihat Membuat sebuah grafik metrik.

> **Note**
>
> Jika rentang waktu dasbor lebih pendek daripada periode yang digunakan untuk grafik di dasbor, hal berikut ini terjadi:
>
> - Grafik dimodifikasi untuk menampilkan jumlah data yang sesuai dengan satu periode lengkap untuk widget tersebut, meskipun ini lebih lama daripada rentang waktu dasbor. Hal ini memastikan bahwa setidaknya ada satu titik data pada grafik.
> - Waktu mulai periode untuk titik data ini akan disesuaikan secara mundur untuk memastikan bahwa setidaknya satu titik data dapat ditampilkan.

Atur rentang waktu relatif

New interface

Untuk menentukan rentang waktu relatif pada grafik

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik, dan kemudian pilih Semua metrik. Di sudut kanan atas layar, Anda dapat memilih salah satu rentang waktu yang telah ditentukan sebelumnya, yang berkisar dari 1 jam hingga 1 minggu (1j, 3j, 12j, 1h, 3h, atau 1m). Atau, Anda dapat memilih Kustom untuk mengatur rentang waktu Anda sendiri.

3. Pilih Kustom, dan kemudian pilih tab Relatif di sudut kiri atas kotak. Anda dapat menentukan rentang waktu dalam Menit, Jam, Hari, Minggu, Bulan.
4. Setelah Anda menentukan rentang waktu, pilih Terapkan.

Original interface

Untuk menentukan rentang waktu relatif pada grafik

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik, dan kemudian pilih Semua metrik. Di sudut kanan atas layar, Anda dapat memilih salah satu rentang waktu yang telah ditentukan sebelumnya, yang berkisar dari 1 jam hingga 1 minggu (1j, 3j, 12j, 1h, 3h, atau 1m). Atau, Anda dapat memilih kustom untuk mengatur rentang waktu Anda sendiri.
3. Pilih Kustom, dan kemudian pilih tab Relatif di sudut kiri atas kotak. Anda dapat menentukan rentang waktu dalam Menit, Jam, Hari, Minggu, atau Bulan.

Atur rentang waktu mutlak

New interface

Untuk menentukan rentang waktu mutlak untuk sebuah grafik

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik, dan kemudian pilih Semua metrik. Di sudut kanan atas layar, Anda dapat memilih salah satu rentang waktu yang telah ditentukan sebelumnya, yang berkisar dari 1 jam hingga 1 minggu (1j, 3j, 12j, 1h, 3h, atau 1m). Atau, Anda dapat memilih Kustom untuk mengatur rentang waktu Anda sendiri.
3. Pilih Kustom, dan kemudian pilih tab Mutlak di sudut kiri atas kotak. Gunakan pemilih kalender atau kotak bidang teks untuk menentukan rentang waktu.
4. Setelah Anda menentukan rentang waktu, pilih Terapkan.

Original interface

Untuk menentukan rentang waktu mutlak untuk sebuah grafik

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.

2. Pada panel navigasi, silakan pilih Metrik, dan kemudian pilih Semua metrik. Di sudut kanan atas layar, Anda dapat memilih salah satu rentang waktu yang telah ditentukan sebelumnya, yang berkisar dari 1 jam hingga 1 minggu (1j, 3j, 12j, 1h, 3h, atau 1m). Atau, Anda dapat memilih kustom untuk mengatur rentang waktu Anda sendiri.
3. Pilih kustom, dan kemudian pilih Mutlak di sudut kiri atas kotak. Gunakan pemilih kalender atau kotak bidang teks untuk menentukan rentang waktu.
4. Setelah Anda menentukan rentang waktu, pilih Terapkan.

Mengatur format zona waktu

New interface

Untuk menentukan zona waktu untuk grafik

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik, dan kemudian pilih Semua metrik. Di sudut kanan atas layar, Anda dapat memilih salah satu rentang waktu yang telah ditentukan sebelumnya, yang berkisar dari 1 jam hingga 1 minggu (1j, 3j, 12j, 1h, 3h, atau 1m). Atau, Anda dapat memilih Kustom untuk mengatur rentang waktu Anda sendiri.
3. Pilih Kustom, dan kemudian pilih dropdown di sudut kanan atas kotak. Anda dapat mengubah zona waktu ke UTC atau zona waktu lokal.
4. Setelah Anda membuat perubahan, pilih Terapkan.

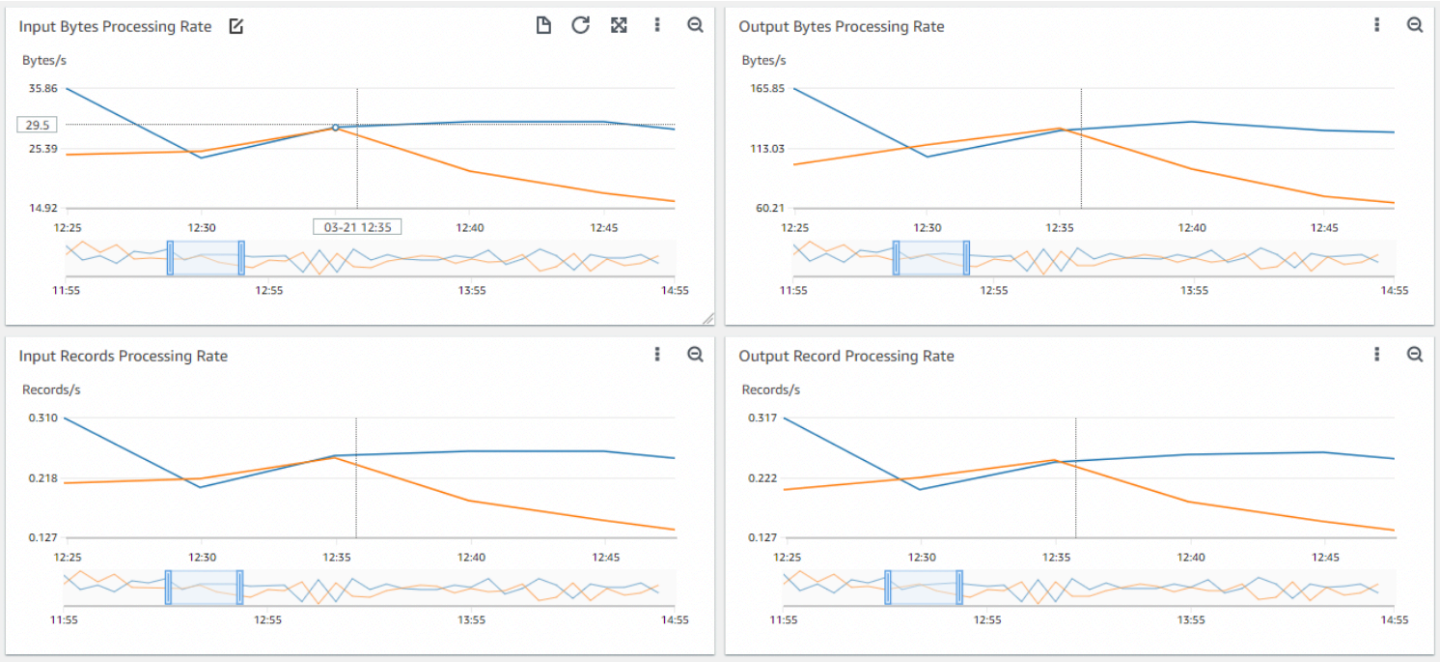
Original interface

Untuk menentukan zona waktu untuk grafik

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik, dan kemudian pilih Semua metrik. Di sudut kanan atas layar, Anda dapat memilih salah satu rentang waktu yang telah ditentukan sebelumnya, yang berkisar dari 1 jam hingga 1 minggu (1j, 3j, 12j, 1h, 3h, atau 1m). Atau, Anda dapat memilih kustom untuk mengatur rentang waktu Anda sendiri.
3. Pilih kustom, dan kemudian pilih dropdown di sudut kanan atas kotak. Anda dapat mengubah zona waktu ke UTC atau zona waktu lokal.

Memperbesar grafik garis atau grafik area bertumpuk

Di CloudWatch konsol, Anda dapat menggunakan fitur zoom peta mini untuk fokus pada bagian grafik garis dan grafik area bertumpuk tanpa mengubah antara tampilan yang diperbesar dan diperbesar. Sebagai contoh, Anda dapat menggunakan fitur zoom peta kecil untuk memfokuskan pada puncak pada grafik garis, sehingga Anda dapat membandingkan lonjakan tersebut dengan metrik lain di dasbor Anda dari rentang waktu yang sama. Prosedur di bagian ini menjelaskan cara menggunakan fitur zoom.



Pada gambar sebelumnya, fitur zoom fokus pada lonjakan grafik garis yang terkait dengan laju pemrosesan byte input saat menampilkan grafik garis lain di dasbor yang juga fokus pada bagian-bagian dari rentang waktu yang sama.

New interface

Untuk memperbesar grafik

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik, dan kemudian pilih Semua metrik.
3. Pilih Jelajahi, dan kemudian pilih satu metrik atau beberapa metrik untuk grafik.
4. Pilih Opsi, dan pilih Garis di bawah Tipe widget.
5. Pilih dan seret pada area grafik yang ingin Anda fokuskan, dan kemudian lepaskan seretannya.

6. Untuk mengatur ulang zoom, pilih ikon Atur ulang zoom, yang terlihat seperti kaca pembesar dengan simbol minus (-) di dalamnya.

Original interface

Untuk memperbesar grafik

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik, dan kemudian pilih Semua metrik.
3. Pilih Semua metrik, dan kemudian pilih satu metrik untuk grafik.
4. Pilih Opsi grafik. Di bawah Tipe widget pilih Garis.
5. Pilih dan seret pada area grafik yang ingin Anda fokuskan, dan kemudian lepaskan seretannya.
6. Untuk mengatur ulang zoom, pilih ikon Atur ulang zoom, yang terlihat seperti kaca pembesar dengan simbol minus (-) di dalamnya.

> **Tip**
>
> Jika sudah membuat dasbor yang berisi grafik garis atau grafik area bertumpuk, Anda dapat pergi ke dasbor dan mulai menggunakan fitur zoom.

Modifikasi sumbu y untuk grafik

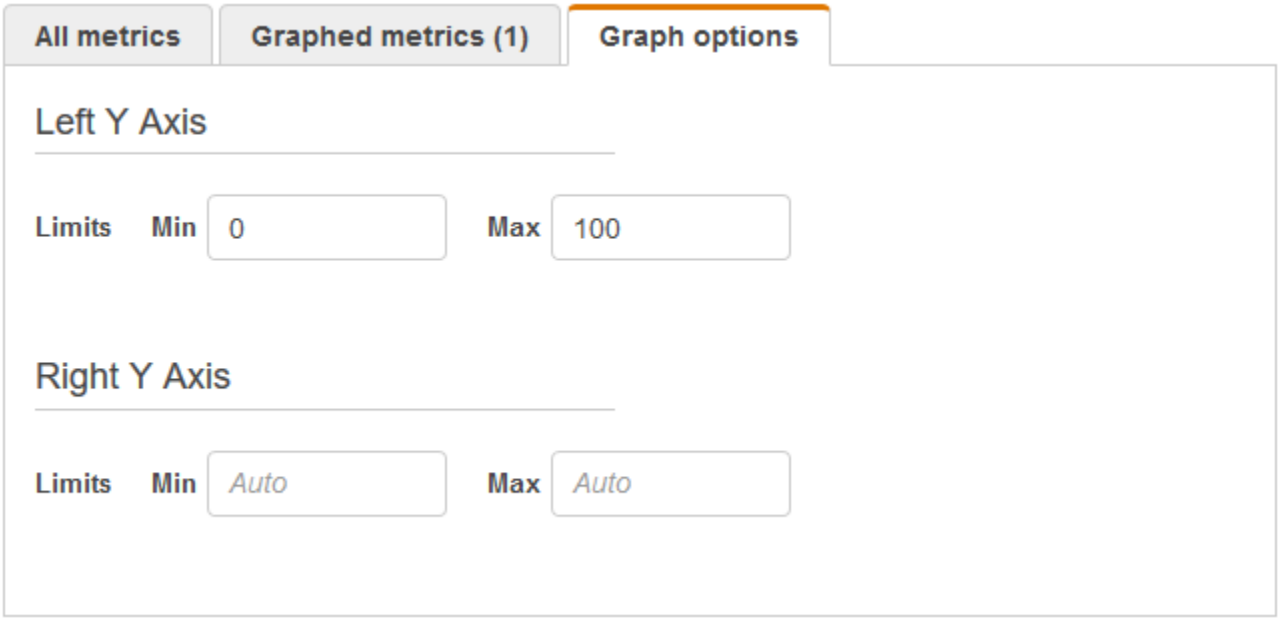
Anda dapat mengatur batas kustom untuk sumbu y pada grafik untuk membantu melihat data dengan lebih baik. Sebagai contoh, Anda dapat mengubah batasan pada grafik `CPUUtilization` hingga 100 persen sehingga mudah untuk melihat apakah CPU rendah (garis plot berada di dekat bagian bawah grafik) atau tinggi (garis plot berada di dekat bagian atas grafik).

Anda dapat beralih antara dua sumbu y yang berbeda untuk grafik. Ini berguna jika grafik memuat metrik yang memiliki unit yang berbeda atau yang sangat berbeda dalam rentang nilai.

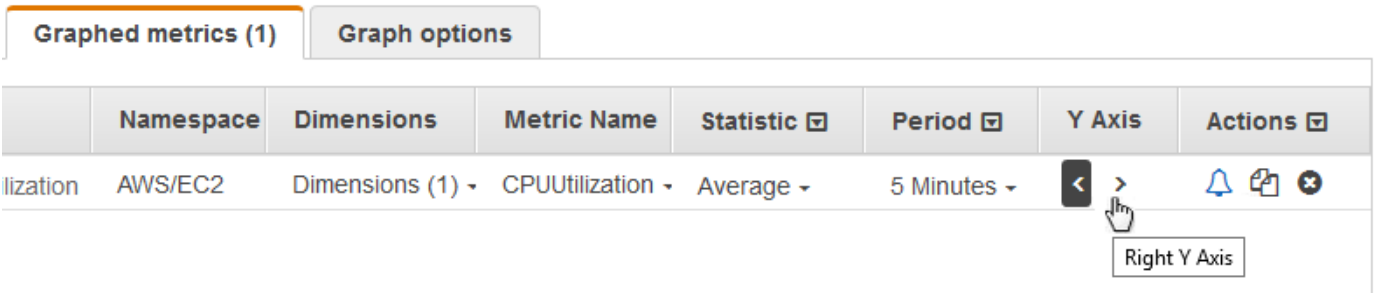
Untuk memodifikasi sumbu y pada grafik

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik.

3. Pilih namespace metrik (misalnya, EC2) dan kemudian dimensi metrik (misalnya, Metrik Per-Instans).
4. Tab Semua metrik menampilkan semua metrik untuk dimensi di namespace tersebut. Untuk membuat grafik sebuah metrik, pilih kotak centang di sebelah metrik.
5. Di tab Opsi grafik, tentukan nilai Min dan Maks untuk Sumbu Y Kiri. Nilai Min tidak boleh lebih besar dari nilai Maks.



6. Untuk membuat sumbu y kedua, tentukan nilai Min dan Maks untuk Sumbu Y Kanan.
7. Untuk beralih antara dua sumbu y, pilih tab Metrik bergrafik. Untuk Sumbu Y, pilih Sumbu Y Kiri atau Sumbu Y Kanan.

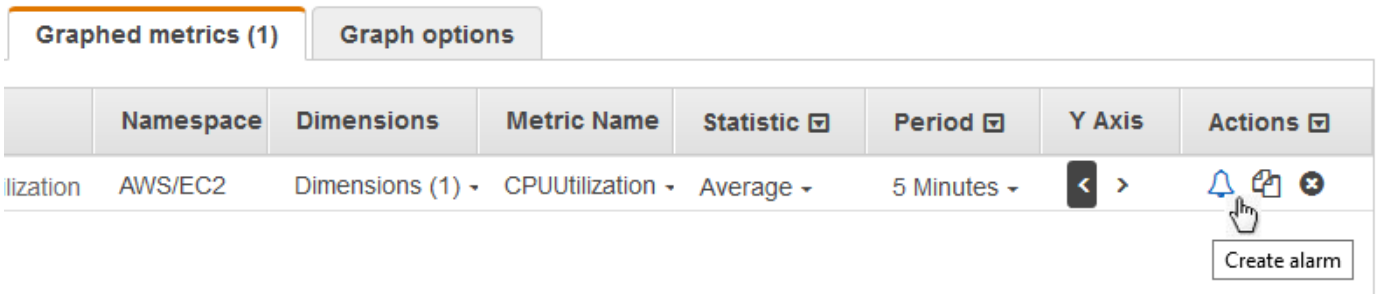


Buat sebuah alarm dari metrik pada grafik

Anda dapat membuat grafik metrik dan kemudian membuat alarm dari metrik pada grafik, yang memiliki manfaat mengisi banyak bidang alarm untuk Anda.

Buat alarm dari metrik pada grafik

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik.
3. Pilih namespace metrik (misalnya, EC2) dan kemudian dimensi metrik (misalnya, Metrik Per-Instans).
4. Tab Semua metrik menampilkan semua metrik untuk dimensi di namespace tersebut. Untuk membuat grafik sebuah metrik, pilih kotak centang di sebelah metrik.
5. Untuk membuat alarm pada metrik, pilih tab Metrik bergrafik. Untuk Tindakan, pilih ikon alarm.



6. Berdasarkan Kondisi, pilih Statis atau Deteksi anomali untuk menentukan apakah akan menggunakan ambang batas statis atau model deteksi anomali untuk alarm.

   Tergantung pada pilihan Anda, masukkan data lainnya untuk kondisi alarm.
7. Pilih Konfigurasi tambahan. Untuk Titik data alarm, tentukan berapa banyak periode evaluasi (titik data) yang harus ada dalam status `ALARM` untuk memicu alarm. Jika kedua nilai di sini cocok, Anda membuat alarm yang beralih ke status `ALARM` jika terjadi pelanggaran selama sebanyak itu dalam periode berturut-turut.

   Untuk membuat alarm M dari N, tentukan angka yang lebih rendah untuk nilai pertama dibandingkan dengan nilai yang Anda tentukan untuk nilai kedua. Untuk informasi selengkapnya, lihat Melakukan evaluasi alarm.
8. Untuk Perlakuan data yang hilang, pilih cara alarm berperilaku ketika beberapa titik data hilang. Untuk informasi selengkapnya, lihat Mengkonfigurasi bagaimana CloudWatch alarm memperlakukan data yang hilang.
9. Pilih Berikutnya.
10. Pada Notifikasi, silakan pilih topik SNS yang akan diberitahukan sebagai notifikasi ketika alarm berada dalam status `ALARM`, status `OK`, atau status `INSUFFICIENT_DATA`.

Agar alarm mengirimkan beberapa notifikasi untuk status alarm yang sama atau untuk status alarm yang berbeda, silakan pilih Tambahkan notifikasi.

Agar alarm tidak mengirim notifikasi, silakan pilih Hapus.

11. Agar alarm dapat melakukan tindakan penskalaan otomatis (Auto Scaling) atau EC2, silakan pilih tombol yang sesuai dan pilih status alarm serta tindakan yang harus dilakukan.
12. Setelah selesai, silakan pilih Berikutnya.
13. Masukkan sebuah nama dan deskripsi untuk alarm tersebut. Nama harus menggunakan karakter ASCII saja. Lalu pilih Berikutnya.
14. Pada Pratinjau dan buat, konfirmasikan bahwa informasi dan ketentuannya sudah sesuai keinginan Anda, kemudian pilih Buat alarm.

Menggunakan CloudWatch deteksi anomali

Saat Anda mengaktifkan deteksi anomali untuk metrik, CloudWatch terapkan algoritma statistik dan pembelajaran mesin. Algoritma ini secara terus-menerus menganalisis metrik sistem dan aplikasi, menentukan garis dasar normal, serta anomali permukaan dengan intervensi pengguna dalam level minimal.

Algoritma menghasilkan model deteksi anomali. Model tersebut menghasilkan serangkaian nilai yang diharapkan yang mewakili perilaku metrik normal.

Anda dapat mengaktifkan deteksi anomali menggunakan AWS Management Console,, AWS CLI AWS CloudFormation, atau SDK AWS . Anda dapat mengaktifkan deteksi anomali pada metrik yang dijual oleh AWS dan juga pada metrik khusus. Di akun yang disiapkan sebagai akun pemantauan untuk pengamatan CloudWatch lintas akun, Anda dapat membuat detektor anomali pada metrik di akun sumber selain metrik di akun pemantauan.

Anda dapat menggunakan model nilai yang diharapkan dengan dua cara:

- Membuat alarm deteksi anomali dengan berdasar nilai yang diharapkan dari metrik. Jenis-jenis alarm ini tidak memiliki ambang statis untuk menentukan status alarm. Alih-alih, jenis-jenis alarm tersebut membandingkan nilai metrik dengan nilai yang diharapkan berdasar model deteksi anomali.

  Anda dapat memilih apakah alarm terpicu ketika nilai metrik berada di atas pita nilai, di bawah pita yang diharapkan, atau keduanya.

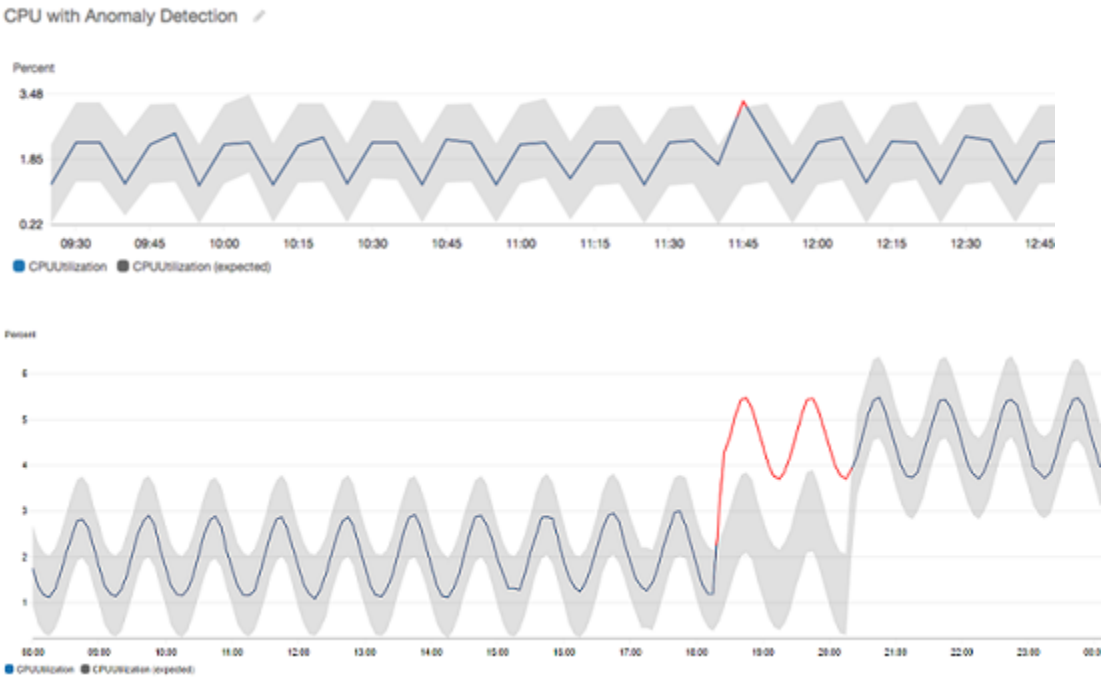
Untuk informasi selengkapnya, lihat Buat CloudWatch alarm berdasarkan deteksi anomali.

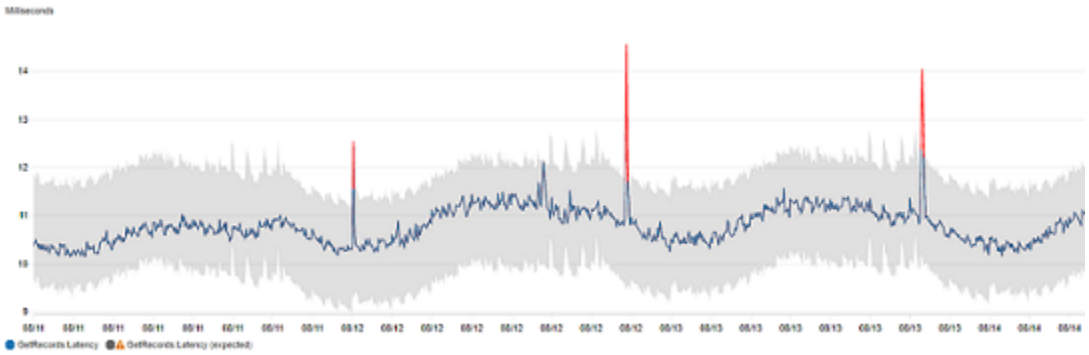
- Ketika melihat grafik data metrik, hamparkan nilai yang diharapkan di grafik sebagai pita. Pita ini secara visual menjelaskan nilai-nilai yang ada di grafik di luar rentang normal. Untuk informasi selengkapnya, lihat Membuat grafik.

  Anda juga dapat mengambil nilai atas dan bawah pita model menggunakan Permintaan API `GetMetricData` dengan `ANOMALY_DETECTION_BAND` fungsi matematika metrik. Untuk informasi lebih lanjut, lihat GetMetricData.

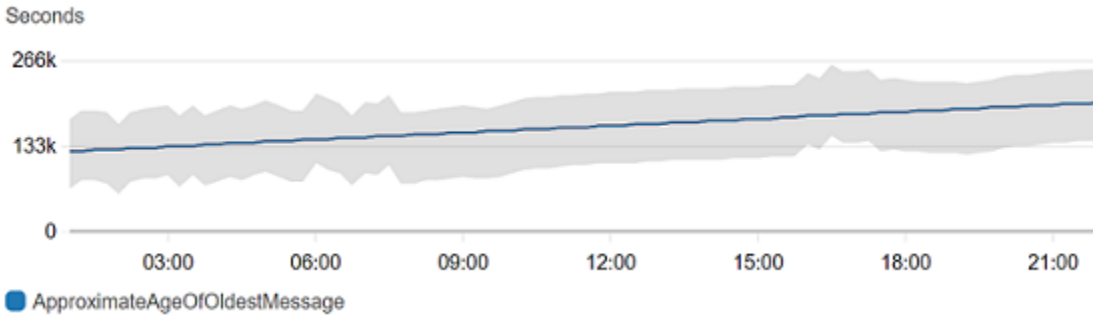
Dalam grafik dengan deteksi anomali, rentang nilai yang diharapkan ditampilkan sebagai pita abu-abu. Jika nilai aktual metrik melampaui pita ini, nilai tersebut akan ditampilkan berwarna merah selama waktu tersebut.

Algoritma deteksi anomali bertanggung jawab atas perubahan-perubahan musiman dan tren metrik. Perubahan musiman dapat dilakukan setiap jam, setiap hari, atau setiap minggu, sebagaimana ditunjukkan dalam contoh-contoh berikut.

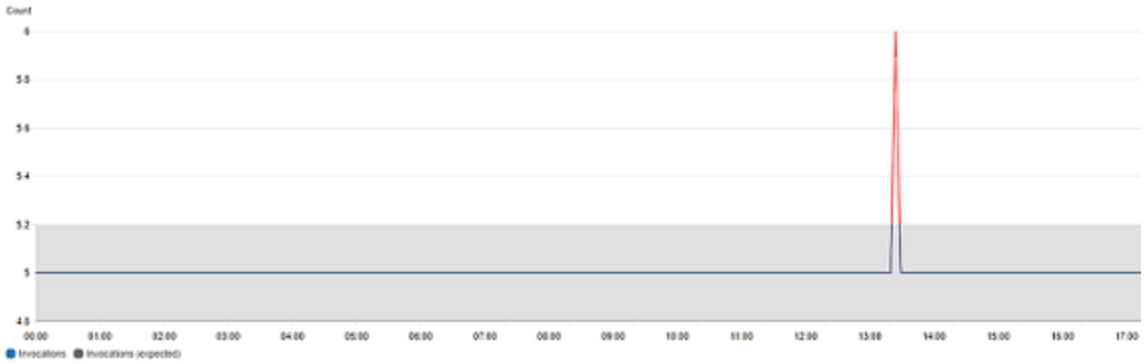




Tren dengan rentang lebih panjang dapat turun atau naik.



Deteksi anomali juga bekerja baik dengan metrik berpola datar.



Bagaimana CloudWatch deteksi anomali bekerja

Saat Anda mengaktifkan deteksi anomali untuk metrik, CloudWatch terapkan algoritme pembelajaran mesin ke data masa lalu metrik untuk membuat model nilai yang diharapkan metrik. Model ini menilai tren maupun pola metrik setiap jam, setiap hari, dan setiap minggu. Algoritma menentukan data metrik hingga dua minggu, tetapi Anda dapat mengaktifkan deteksi anomali pada metrik bahkan jika metrik tidak memiliki data dua minggu penuh.

Anda menentukan nilai untuk ambang deteksi anomali yang CloudWatch digunakan bersama dengan model untuk menentukan rentang nilai “normal” untuk metrik. Nilai yang lebih tinggi untuk ambang batas deteksi anomali menghasilkan pita yang lebih tebal daripada nilai "normal".

Model machine learning spesifik untuk metrik dan statistik. Sebagai contoh, jika Anda mengaktifkan deteksi anomali untuk metrik menggunakan statistik `AVG`, model spesifik untuk statistik `AVG`.

Saat CloudWatch membuat model untuk banyak metrik umum dari AWS layanan, ini memastikan bahwa band tidak meluas di luar nilai logis. Misalnya, band untuk `MemoryUtilization` instans EC2 akan tetap antara 0 dan 100, dan pelacakan pita CloudFront `Requests`, yang tidak bisa negatif, tidak akan pernah memanjang di bawah nol.

Setelah Anda membuat model, deteksi CloudWatch anomali terus mengevaluasi model dan membuat penyesuaian untuk memastikan bahwa itu seakurat mungkin. Penyesuaian ini termasuk penentuan ulang model untuk menyesuaikan jika nilai metrik berkembang seiring waktu atau mengalami perubahan mendadak, dan juga menyertakan prediktor untuk meningkatkan model metrik yang bersifat musiman, runcing, atau jarang.

Setelah Anda mengaktifkan deteksi anomali pada metrik, Anda dapat memilih untuk mengecualikan periode waktu tertentu dari metrik yang digunakan untuk melatih model tersebut. Dengan cara ini, Anda dapat mengecualikan penyebaran atau peristiwa tidak lazim lainnya dari penggunaan untuk penyesuaian model, yang memastikan model yang paling akurat telah dibuat.

Menggunakan model deteksi anomali untuk alarm menimbulkan biaya pada akun Anda. AWS Untuk informasi lebih lanjut, lihat Amazon CloudWatch Harga.

Deteksi anomali pada matematika metrik

Deteksi anomali pada matematika metrik adalah sebuah fitur yang dapat Anda gunakan untuk membuat alarm deteksi anomali pada hasil ekspresi matematika metrik. Anda dapat menggunakan ekspresi-ekspresi ini untuk membuat grafik yang menggambarkan pita deteksi anomali. Fitur ini mendukung fungsi aritmatika dasar, operator perbandingan dan logika, dan sebagian besar fungsi lainnya. Untuk informasi tentang fungsi yang tidak didukung, lihat Menggunakan matematika metrik di Panduan CloudWatch Pengguna Amazon.

Anda dapat membuat model deteksi anomali berdasarkan ekspresi matematika metrik yang sama dengan cara Anda membuat model deteksi anomali. Dari CloudWatch konsol, Anda dapat menerapkan deteksi anomali ke ekspresi matematika metrik dan memilih deteksi anomali sebagai tipe ambang batas untuk ekspresi ini.

> **Note**
>
> Deteksi anomali pada matematika metrik hanya dapat diaktifkan dan diedit di antarmuka pengguna metrik versi terbaru. Ketika Anda membuat detektor anomali berdasarkan ekspresi matematika metrik di versi antarmuka yang baru, Anda dapat melihatnya di versi lama, tetapi tidak dapat mengeditnya.

Untuk informasi tentang cara membuat alarm dan model untuk deteksi anomali dan matematika metrik, silakan lihat bagian-bagian berikut:

- Membuat CloudWatch alarm berdasarkan deteksi anomali
- Membuat CloudWatch alarm berdasarkan ekspresi matematika metrik

Anda juga dapat membuat, menghapus, dan menemukan model deteksi anomali berdasarkan ekspresi matematika metrik menggunakan CloudWatch API dengan`PutAnomalyDetector`,`DeleteAnomalyDetector`, dan. `DescribeAnomalyDetectors` Untuk informasi tentang tindakan API ini, lihat bagian berikut di Referensi Amazon CloudWatch API.

- PutAnomalyDetector
- DeleteAnomalyDetector
- DescribeAnomalyDetectors

Untuk informasi tentang harga alarm deteksi anomali, lihat harga Amazon. CloudWatch

Gunakan matematika metrik

Matematika metrik memungkinkan Anda untuk menanyakan beberapa CloudWatch metrik dan menggunakan ekspresi matematika untuk membuat deret waktu baru berdasarkan metrik ini. Anda dapat memvisualisasikan deret waktu yang dihasilkan di CloudWatch konsol dan menambahkannya ke dasbor. Dengan menggunakan AWS Lambda metrik sebagai contoh, Anda dapat membagi `Errors` metrik dengan `Invocations` metrik untuk mendapatkan tingkat kesalahan. Kemudian tambahkan deret waktu yang dihasilkan ke grafik di CloudWatch dasbor Anda.

Anda juga dapat melakukan matematika metrik secara terprogram, menggunakan Operasi API `GetMetricData`. Untuk informasi lebih lanjut, lihat GetMetricData.

Tambahkan ekspresi matematika ke CloudWatch grafik

Anda dapat menambahkan ekspresi matematika ke grafik di CloudWatch dasbor Anda. Setiap grafik dibatasi menggunakan maksimum 500 metrik dan ekspresi, sehingga Anda dapat menambahkan ekspresi matematika hanya jika grafik memiliki 499 metrik atau lebih sedikit. Ini berlaku meskipun tidak semua metrik ditampilkan pada grafik.

Untuk menambahkan ekspresi matematika ke grafik

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Buat atau sunting grafik. Setidaknya harus ada satu metrik dalam grafik.
3. Pilih Metrik bergrafik.
4. Pilih Ekspresi matematika, Mulai dengan ekspresi kosong. Garis baru muncul untuk ekspresi.
5. Di garis baru, berdasarkan kolom Detail, masukkan ekspresi matematika. Tabel di bagian Sintaksis dan Fungsi Matematika Metrik mencantumkan fungsi yang dapat Anda gunakan dalam ekspresi.

   Untuk menggunakan metrik atau hasil ekspresi lain sebagai bagian dari rumus untuk ekspresi ini, gunakan nilai yang ditunjukkan di kolom Id: misalnya, m1+m2 atau e1-MIN(e1).

   Anda dapat mengubah nilai Id. Ini dapat mencakup angka, huruf, garis bawah, dan harus dimulai dengan huruf kecil. Mengubah nilai Id menjadi nama yang lebih bermakna juga dapat membuat grafik lebih mudah dipahami; misalnya, mengubah m1 dan m2 menjadi kesalahan dan permintaan.

   > **Tip**
   >
   > Pilih panah turun di sebelah Ekspresi Matematika untuk melihat daftar fungsi yang didukung, yang dapat Anda gunakan ketika membuat ekspresi.

6. Untuk kolom Label dari ekspresi, masukkan nama yang menjelaskan penghitungan ekspresi.

   Jika hasil dari ekspresi adalah susunan deret waktu, masing-masing rangkaian waktu tersebut ditampilkan pada grafik dengan garis terpisah, dengan warna berbeda. Tepat di bawah grafik terdapat legenda untuk setiap garis dalam grafik. Untuk ekspresi tunggal yang menghasilkan beberapa rangkaian waktu, keterangan legenda untuk deret waktu tersebut berada dalam format ***Expression-Label Metric-Label***. Sebagai contoh, jika grafik mencakup metrik dengan label Kesalahan dan ekspresiFILL(METRICS(), 0) yang memiliki label Filled With 0:, satu garis

dalam legenda akan Filled With 0: Errors. Agar legenda hanya menampilkan label metrik asli, atur *`Expression-Label`* menjadi kosong.

Ketika satu ekspresi menghasilkan susunan deret waktu pada grafik, Anda tidak dapat mengubah warna yang digunakan untuk setiap deret waktu tersebut.

7. Setelah menambahkan ekspresi yang diinginkan, Anda dapat menyederhanakan grafik dengan menyembunyikan beberapa metrik asli. Untuk menyembunyikan metrik atau ekspresi, hapus kotak centang di sebelah kiri bidang Id.

Sintaks dan fungsi matematika metrik

Bagian berikut menjelaskan fungsi yang tersedia untuk matematika metrik. Semua fungsi harus ditulis dalam huruf besar (seperti AVG), dan bidang Id untuk semua metrik dan ekspresi matematika harus dimulai dengan huruf kecil.

Hasil akhir ekspresi matematika harus berupa satu deret waktu atau susunan deret waktu. Beberapa fungsi menghasilkan angka skalar. Anda dapat menggunakan fungsi ini dalam fungsi yang lebih besar sehingga pada akhirnya menghasilkan satu deret waktu. Misalnya, mengambil AVG dari satu deret waktu menghasilkan angka skalar, sehingga ini tidak dapat menjadi hasil ekspresi akhir. Namun demikian, Anda dapat menggunakannya dalam fungsi m1-AVG(m1) untuk menampilkan deret waktu dari perbedaan antara setiap titik data individu dan nilai rata-rata dalam deret waktu.

Singkatan tipe data

Beberapa fungsi hanya berlaku untuk jenis data tertentu. Singkatan dalam daftar berikut digunakan dalam tabel fungsi untuk mewakili jenis data yang didukung pada setiap fungsi:

- S mewakili angka skalar, seperti 2, -5, atau 50,25.
- TS adalah deret waktu (serangkaian nilai untuk satu CloudWatch metrik dari waktu ke waktu): misalnya, `CPUUtilization` metrik misalnya `i-1234567890abcdef0` selama tiga hari terakhir.
- TS[] adalah susunan deret waktu, seperti deret waktu untuk beberapa metrik.
- String[] adalah susunan string.

Fungsi METRICS()

Fungsi METRICS() mengembalikan semua metrik dalam permintaan. Ekspresi matematika tidak disertakan.

Anda dapat menggunakan METRICS() dalam ekspresi yang lebih besar yang menghasilkan satu deret waktu atau susunan deret waktu. Sebagai contoh, ekspresi SUM(METRICS())mengembalikan deret waktu (TS) yang merupakan jumlah dari nilai semua metrik yang digambarkan. METRICS()/100 mengembalikan susunan deret waktu, masing-masing merupakan deret waktu yang menunjukkan setiap titik data dari salah satu metrik dibagi 100.

Anda dapat menggunakan fungsi METRICS() dengan sebuah string untuk mengembalikan hanya metrik yang digambarkan yang mengandung string tersebut dalam kolom Id mereka. Sebagai contoh, ekspresi SUM(METRICS("error")) mengembalikan deret waktu yang merupakan jumlah nilai semua metrik yang digambarkan yang memiliki ‘kesalahan’ dalam kolom Id mereka. Anda juga dapat menggunakan SUM([METRICS (“4xx”), METRICS (“5xx”)]) untuk mencocokkan beberapa string.

Fungsi aritmetika dasar

Tabel berikut mencantumkan daftar fungsi aritmetika dasar yang didukung. Nilai yang hilang dalam deret waktu dianggap 0. Jika nilai titik data menyebabkan fungsi mencoba membagi dengan nol, maka titik data diabaikan.

Operasi	Argumen	Contoh
Operator aritmetika: + - * / ^	S, S	PERIOD(m1)/60
	S, TS	5 * m1
	TS, TS	m1 - m2
	S, TS[]	SUM(100/[m1, m2])
	TS, TS[]	AVG(METRICS())
		METRICS()*100
Pengurangan uner -	S	-5*m1
	TS	-m1
	TS[]	SUM(-[m1, m2])

Operator perbandingan dan logis

Anda dapat menggunakan operator perbandingan dan logika baik dengan sepasang deret waktu maupun sepasang nilai skalar tunggal. Ketika Anda menggunakan operator perbandingan dengan sepasang deret waktu, operator mengembalikan deret waktu di mana setiap titik data adalah 0 (salah) atau 1 (benar). Jika Anda menggunakan operator perbandingan pada sepasang nilai skalar, satu nilai skalar akan dikembalikan, baik 0 maupun 1.

Ketika operator perbandingan digunakan antara dua deret waktu, dan hanya satu deret waktu yang memiliki nilai untuk time stamp tertentu, fungsi menganggap nilai yang hilang dalam deret waktu lain sebagai 0.

Anda dapat menggunakan operator logis bersama operator perbandingan, untuk membuat fungsi yang lebih kompleks.

Tabel berikut mencantumkan operator yang didukung.

Jenis operator	Operator yang didukung
Operator Perbandingan	== != <= >= < >
Operator Logis	AND atau && OR atau ||

Untuk melihat bagaimana operator ini digunakan, misalkan kita memiliki dua deret waktu: metric1 memiliki nilai `[30, 20, 0, 0]` dan metric2 memiliki nilai `[20, -, 20, -]` di mana `-` menunjukkan bahwa tidak ada nilai untuk timestamp tersebut.

Ekspresi	Output
(metric1 < metric2)	0, 0, 1, 0
(metric1 >= 30)	1, 0, 0, 0
(metric1 > 15 AND metric2 > 15)	1, 0, 0, 0

Fungsi yang didukung untuk matematika metrik

Tabel berikut menjelaskan fungsi yang dapat Anda gunakan dalam ekspresi matematika. Masukkan semua fungsi dalam huruf besar.

Hasil akhir ekspresi matematika harus berupa satu deret waktu atau susunan deret waktu. Beberapa fungsi dalam tabel di bagian berikut menghasilkan angka skalar. Anda dapat menggunakan fungsi ini dalam fungsi yang lebih besar sehingga pada akhirnya menghasilkan satu deret waktu. Misalnya, mengambil AVG dari satu deret waktu menghasilkan angka skalar, sehingga ini tidak dapat menjadi hasil ekspresi akhir. Namun demikian, Anda dapat menggunakannya dalam fungsi m1-AVG(m1) untuk menampilkan deret waktu dari perbedaan antara setiap titik data individu dan nilai rata-rata dari titik data itu.

Dalam tabel berikut, setiap contoh dalam kolom Contoh adalah ekspresi yang menghasilkan satu deret waktu atau susunan deret waktu. Contoh ini menunjukkan cara fungsi yang mengembalikan angka skalar dapat digunakan sebagai bagian dari ekspresi yang benar guna menghasilkan satu deret waktu.

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
ABS	TS TS[]	TS TS[]	Mengembalikan nilai mutlak dari setiap titik data.	ABS(m1-m2) MIN(ABS([m1, m2])) ABS(METRICS())	✓

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
ANOMALY_D ETECTION_ BAND	TS TS, S	TS[]	Mengembalikan pita deteksi anomali untuk metrik tertentu. Pita ini terdiri atas dua deret waktu, satu mewakili batas atas nilai yang diharapkan "normal" dari metrik, dan satu lagi mewakili batas bawah. Fungsi dapat menerima dua argumen. Yang pertama adalah ID metrik untuk membuat pita. Argumen kedua adalah jumlah penyimpangan baku yang digunakan untuk pita. Jika Anda tidak menentukan argumen ini, default 2 digunakan. Untuk informasi selengkap nya, lihat Menggunak an CloudWatch deteksi anomali.	ANOMALY_D ETECTION_BAND(m1) ANOMALY_D ETECTION_BAND(m1,4)	

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
AVG	TS TS[]	S TS	AVG dari satu deret waktu mengembalikan skalar yang mewakili rata-rata semua titik data dalam metrik. AVG dari susunan deret waktu mengembal ikan satu rangkaian waktu. Nilai yang hilang dianggap sebagai 0. **Note** Kami menyarankan Anda untuk tidak menggunak an fungsi ini di CloudWatch alarm jika Anda ingin fungsi mengembalikan skalar. Misalnya, `AVG(m2)`. Setiap kali alarm mengevaluasi apakah akan mengubah status, CloudWatch mencoba untuk mengambil	SUM([m1,m2])/AVG(m2) AVG(METRICS())	✓

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
			jumlah titik data yang lebih tinggi daripada nomor yang ditentukan sebagai Periode Evaluasi. Fungsi ini berperilaku berbeda ketika data tambahan diminta. Untuk menggunak an fungsi ini dengan alarm, terutama alarm yang memiliki tindakan Penskalaa n Otomatis, kami sarankan Anda mengatur alarm untuk menggunakan M dari titik data N, di mana M < N.		

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
CEIL	TS TS[]	TS TS[]	Mengembalikan batas tertinggi setiap metrik. Batas tertinggi adalah bilangan bulat terkecil yang lebih besar dari atau sama dengan setiap nilai.	CEIL(m1) CEIL(METRICS()) SUM(CEIL(METRICS()))	✓

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
DATAPOINT _COUNT	TS TS[]	S TS	Mengembalikan hitungan titik data yang melaporkan nilai-nil ai. Ini berguna untuk menghitung rata-rata metrik yang jarang. **Note** Kami menyarankan Anda untuk tidak menggunak an fungsi ini di CloudWatch alarm. Setiap kali alarm mengevaluasi apakah akan mengubah status, CloudWatch mencoba untuk mengambil jumlah titik data yang lebih tinggi daripada nomor yang ditentukan sebagai Periode Evaluasi. Fungsi ini berperilaku berbeda ketika	SUM(m1) / DATAPOINT _COUNT(m1) DATAPOINT _COUNT(METRICS())	✓

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
			data tambahan diminta.		

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
DB_PERF_INSIGHTS	String, String, String String, String, String[]	TS (jika diberi satu string) TS[] (jika diberi susunar string)	Mengembalikan metrik Wawasan Performa Counter untuk basis data seperti Amazon Relational Database Service dan Amazon DocumentDB (dengan kompatibilitas MongoDB). Fungsi ini mengembalikan jumlah data yang sama yang bisa Anda dapatkan dengan langsung menanyakan API Wawasan Performa. Anda dapat menggunakan metrik ini CloudWatch untuk membuat grafik dan membuat alarm. **Important** Bila Anda menggunakan fungsi ini, Anda harus menentukan ID Sumber Daya Basis Data Unik dari basis data tersebut. Hal ini berbeda dengan	DB_PERF_INSIGHTS('RDS', 'db-ABCDEFGHIJKLMNOPQRSTUVWXY1', 'os.cpuUtilization.user.avg') DB_PERF_INSIGHTS('DOCDB, 'db-ABCDEFGHIJKLMNOPQRSTUVWXY1', ['os.cpuUtilization.idle.avg', 'os.cpuUtilization.user.max'])	

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
			pengident ifikasi basis data. Untuk menemukan ID sumber daya basis data di konsol Amazon RDS, pilih instans DB untuk melihat detailnya. Kemudian, pilih tab Konfigura si. ID Sumber Daya ditampilk an di bagian Konfigurasi. DB_PERF_INSIGHTS juga akan membawa metrik DBLoad pada interval sub-menit. Metrik Performance Insights yang diambil dengan fungsi ini tidak disimpan. CloudWatc h Oleh karena itu, beberapa CloudWatch fitur seperti observabi litas lintas akun, deteksi		

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
			anomali, aliran metrik, penjelajah metrik, dan Metric Insights tidak berfungsi dengan metrik Performance Insights yang Anda ambil dengan DB_PERF_INSIGHTS. Permintaan tunggal menggunakan fungsi DB_PERF_INSIGHTS dapat mengambil nomor titik data berikut. • 1080 titik data untuk periode resolusi tinggi (1 detik, 10 detik, 30-an) • 1440 titik data untuk periode resolusi standar (1m, 5m, 1 jam, 1d) Fungsi DB_PERF_INSIGHTS hanya mendukung panjang periode berikut: • 1 detik • 10 detik • 30 detik		

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
			• 1 menit • 5 Menit • 1 Jam • 1 hari Untuk informasi selengkapnya tentang metrik penghitung Wawasan Performa Amazon RDS, silakan lihat Metrik penghitung Wawasan Performa. Untuk informasi selengkapnya tentang metrik penghitung Wawasan Performa Amazon DocumentDB, silakan lihat Metrik penghitung Wawasan Performa. **Note** Metrik-metrik resolusi tinggi dengan pedetail sub-menit yang diambil oleh DB_PERF_INSIGHTS hanya berlaku		

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
			untuk metrik DBLoad, atau untuk metrik sistem operasi jika Anda telah mengaktifkan Pemantauan yang Ditingkat kan dengan resolusi yang lebih tinggi. Untuk informasi selengkap nya tentang pemantauan yang ditingkat kan Amazon RDS, silakan lihat Memantau metrik OS dengan Pemantauan yang Ditingkat kan. . Anda dapat membuat sebuah alarm resolusi tinggi dengan menggunak an fungsi DB_PERF_I		

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
			NSIGHTS untuk rentang waktu maksimum selama tiga jam. Anda dapat menggunakan CloudWatch konsol untuk membuat grafik metrik yang diambil dengan fungsi DB_PERF_I NSIGHTS untuk rentang waktu apa pun.		
DIFF	TS TS[]	TS TS[]	Mengembalikan perbedaan antara setiap nilai dalam deret waktu dan nilai sebelumny a dari deret waktu tersebut.	DIFF(m1)	✓
DIFF_TIME	TS TS[]	TS TS[]	Mengembalikan perbedaan dalam detik antara timestamp setiap nilai dalam deret waktu dan timestamp nilai sebelumnya dari deret waktu tersebut.	DIFF_TIME(METRICS())	✓

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
FILL	TS, [S | REPEA | LINEAR TS[], [TS | S | REPEA | LINEAR	TS TS[]	Mengisi nilai-nilai yang hilang dari deret waktu. Ada beberapa pilihan untuk nilai yang akan digunakan sebagai pengisi untuk nilai-nilai yang hilang: • Anda dapat menentukan sebuah nilai yang akan digunakan sebagai nilai pengisi. • Anda dapat menentukan sebuah metrik yang akan digunakan sebagai nilai pengisi. • Anda dapat menggunakan kata kunci REPEAT untuk mengisi nilai-nilai yang hilang dengan nilai aktual metrik terbaru sebelum nilai yang hilang tersebut. • Anda dapat menggunakan kata kunci LINEAR untuk mengisi nilai-nilai yang hilang dengan	FILL(m1,10) FILL(METRICS(), 0) FILL(METRICS(), m1) FILL(m1, MIN(m1)) FILL(m1, REPEAT) FILL(METRICS(), LINEAR)	✓

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
			nilai-nilai yang menciptakan interpola si linear antara nilai-nilai di awal dan akhir kesenjangan. **Note** Saat menggunak an fungsi ini dalam alarm, Anda dapat mengalami masalah jika metrik Anda dipublikasikan dengan sedikit penundaan, dan menit terakhir tidak pernah memiliki data. Dalam hal ini, FILL mengganti kan titik data yang hilang dengan nilai yang diminta. Ini menyebabkan titik data terbaru untuk metrik		

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
			selalu menjadi nilai FILL, yang dapat mengakiba tkan alarm macet dalam status OK atau status ALARM. Anda dapat mengatasi ini menggunakan alarm M dari N. Untuk informasi selengkapnya, lihat Melakukan evaluasi alarm.		

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
FIRST LAST	TS[]	TS	Mengembalikan deret waktu pertama atau terakhir dari susunan deret waktu. Ini berguna ketika digunakan dengan fungsi SORT. Ini juga dapat digunakan untuk mendapatk an ambang batas tinggi dan rendah dari fungsi ANOMALY_D ETECTION_BAND.	IF(FIRST(SORT(METR ICS(), AVG, DESC))>10 0, 1, 0) Melihat metrik teratas dari susunan, yang diurutkan oleh AVG. Ini kemudian akan mengembalikan 1 atau 0 untuk setiap titik data, tergantung pada apakah nilai titik data tersebut lebih dari 100. LAST(ANOM ALY_DETEC TION_BAND(m1)) mengembalikan batas atas pita prediksi anomali.	✓
FLOOR	TS TS[]	TS TS[]	Mengembalikan pangkat setiap metrik. Pangkat adalah bilangan bulat terbesar yang kurang dari atau sama dengan setiap nilai.	FLOOR(m1) FLOOR(METRICS())	✓

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
IF	Ekspres IF	TS	Gunakan IF bersama operator perbandingan untuk memfilter titik data dari deret waktu, atau membuat deret waktu campuran yang terdiri atas beberapa deret waktu gabungan. Untuk informasi selengkap nya, lihat Menggunakan ekspresi IF.	Sebagai contoh, silakan lihat Menggunakan ekspresi IF.	✓
INSIGHT_R ULE_METRIC	INSIGH ULE_MI C(ruleN e, metricN e)	TS	Gunakan INSIGHT_R ULE_METRIC untuk mengekstrak statistik dari aturan di Contribut or Insightsr. Untuk informasi selengkap nya, lihat Membuat grafik metrik-metrik yang dihasilkan oleh aturan.		
LAMBDA	LAMBD (Lambd nctionN e [, opsiona arg] *)	TS TS{}	Memanggil fungsi Lambda untuk menanyakan metrik dari sumber data yang tidak. CloudWatch Untuk informasi selengkapnya, lihat Cara meneruska n argumen ke fungsi Lambda Anda.		

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
LOG	TS TS[]	TS TS[]	LOG dari deret waktu mengembalikan nilai logaritma alami dari setiap nilai dalam deret waktu.	LOG(METRICS())	✓
LOG10	TS TS[]	TS TS[]	LOG dari deret waktu mengembalikan nilai logaritma dasar-10 dari setiap nilai dalam deret waktu.	LOG10(m1)	✓

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
MAX	TS TS[]	S TS	MAX dari satu deret waktu mengembalikan skalar yang mewakili nilai maksimum semua titik data dalam metrik. Nilai MAX dari susunan deret waktu mengembalikan satu deret waktu. **Note** Kami menyarankan Anda untuk tidak menggunakan fungsi ini di CloudWatch alarm jika Anda ingin fungsi mengembalikan skalar. Misalnya, `MAX(m2)` Setiap kali alarm mengevaluasi apakah akan mengubah status, CloudWatch mencoba untuk mengambil jumlah titik data yang lebih tinggi	MAX(m1)/m1 MAX(METRICS())	✓

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
			daripada nomor yang ditentukan sebagai Periode Evaluasi. Fungsi ini berperilaku berbeda ketika data tambahan diminta.		
METRIC_CO UNT	TS[]	S	Mengembalikan jumlah metrik dalam susunan deret waktu.	m1/METRIC _COUNT(METRICS())	✓

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
METRICS	kosong string	TS[]	Fungsi METRICS () mengembalikan semua CloudWatch metrik dalam permintaan. Ekspresi matematika tidak disertakan. Anda dapat menggunak an METRICS() dalam ekspresi yang lebih besar yang menghasil kan satu deret waktu atau susunan deret waktu. Anda dapat menggunak an fungsi METRICS() dengan sebuah string untuk mengembalikan hanya metrik yang digambarkan yang mengandung string tersebut dalam kolom Id mereka. Sebagai contoh, ekspresi SUM(METRICS("error ")) mengembalikan deret waktu yang merupakan jumlah nilai semua metrik yang digambark an yang memiliki ‘kesalahan’ dalam kolom	AVG(METRICS()) SUM(METRICS("error s"))	✓

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
			Id mereka. Anda juga dapat menggunakan SUM([METRICS (“4xx”), METRICS (“5xx”)]) untuk mencocokkan beberapa string.		

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
MIN	TS TS[]	S TS	MIN dari satu deret waktu tunggal mengembalikan skalar yang mewakili nilai minimum semua titik data dalam metrik. MIN dari susunan deret waktu mengembalikan satu rangkaian waktu. **Note** Kami menyarankan Anda untuk tidak menggunak an fungsi ini di CloudWatch alarm jika Anda ingin fungsi mengembalikan skalar. Misalnya, `MIN(m2)` Setiap kali alarm mengevaluasi apakah akan mengubah status, CloudWatch mencoba untuk mengambil jumlah titik data	m1-MIN(m1) MIN(METRICS())	✓

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
			yang lebih tinggi daripada nomor yang ditentukan sebagai Periode Evaluasi. Fungsi ini berperilaku berbeda ketika data tambahan diminta.		

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?

MINUTE
HOUR
DAY
DATE
MONTH
YEAR
EPOCH
	TS	TS	Fungsi-fungsi ini mengambil periode dan rentang dari deret waktu dan mengembalikan deret waktu baru yang tidak jarang di mana setiap nilai didasarkan pada timestampnya.
			• MINUTE mengembal ikan deret waktu yang tidak jarang dari bilangan bulat antara 0 dan 59 yang mewakili menit UTC dari setiap timestamp dalam deret waktu asli.
			• HOUR mengembal ikan deret waktu yang tidak jarang dari bilangan bulat antara 0 dan 23 yang mewakili menit UTC dari setiap timestamp dalam deret waktu asli.
			• DAY mengembal ikan deret waktu yang tidak jarang dari bilangan bulat antara
				MINUTE(m1)
				IF(DAY(m1)<6,m1) mengembalikan metrik hanya dari hari kerja, Senin sampai Jumat.
				IF(MONTH(m1) == 4,m1) mengembalikan hanya metrik yang diterbitkan pada bulan April.
					✓

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
			1 dan 7 yang mewakili hari UTC dalam seminggu dari setiap timestamp dalam deret waktu asli. 1 mewakili Senin dan 7 mewakili Minggu. • DATE mengembal ikan deret waktu yang tidak jarang dari bilangan bulat antara 1 dan 31 yang mewakili hari UTC dalam sebulan dari setiap timestamp dalam deret waktu asli. • BULAN mengembal ikan deret waktu yang tidak jarang dari bilangan bulat antara 1 dan 12 yang mewakili bulan UTC dari setiap timestamp dalam deret waktu asli. 1 mewakili Januari dan 12 mewakili Desember. • TAHUN mengembal ikan deret waktu		

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
			yang tidak jarang dari bilangan bulat yang mewakili tahun UTC dari setiap timestamp dalam deret waktu asli. • EPOCH mengembal ikan deret waktu yang tidak jarang dari bilangan bulat yang mewakili waktu UTC dalam detik sejak Epoch dari setiap timestamp dalam deret waktu asli. Epoch adalah 1 Januari 1970.		
PERIOD	TS	S	Mengembalikan periode metrik dalam detik. Input yang benar adalah metrik, bukan hasil ekspresi lainnya.	m1/PERIOD(m1)	✓

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
RATE	TS TS[]	TS TS[]	Mengembalikan tingkat perubahan metrik per detik. Ini dihitung sebagai selisih antara nilai titik data terbaru dan nilai titik data sebelumnya, dibagi dengan perbedaan waktu dalam detik antara dua nilai. **Important** Menyetel alarm pada ekspresi yang menggunakan fungsi RATE pada metrik dengan data jarang dapat berperilaku tidak terduga, karena rentang titik data yang diambil saat mengevalu asi alarm dapat bervariasi berdasarkan kapan titik data	RATE(m1) RATE(METRICS())	✓

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
			terakhir diterbitk an.		

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
REMOVE_EM PTY	TS[]	TS[]	Menghapus setiap deret waktu yang tidak memiliki titik data dari susunan deret waktu. Hasilnya adalah susunan deret waktu di mana setiap rangkaian waktu memuat paling sedikit satu titik data. **Note** Kami menyarankan Anda untuk tidak menggunak an fungsi ini di CloudWatch alarm. Setiap kali alarm mengevaluasi apakah akan mengubah status, CloudWatch mencoba untuk mengambil jumlah titik data yang lebih tinggi daripada nomor yang ditentukan sebagai Periode	REMOVE_EM PTY(METRICS())	✓

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
			Evaluasi. Fungsi ini berperilaku berbeda ketika data tambahan diminta.		

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
RUNNING_SUM	TS TS[]	TS TS[]	Mengembalikan deret waktu dengan jumlah berjalan dari nilai dalam deret waktu asli. **Note** Kami menyarankan Anda untuk tidak menggunakan fungsi ini di CloudWatch alarm. Setiap kali alarm mengevaluasi apakah akan mengubah status, CloudWatch mencoba untuk mengambil jumlah titik data yang lebih tinggi daripada nomor yang ditentukan sebagai Periode Evaluasi. Fungsi ini berperilaku berbeda ketika	RUNNING_SUM([m1,m2])	✓

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
			data tambahan diminta.		

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
SEARCH	Ekspres pencari	Satu TS atau lebih	Mengembalikan satu deret waktu atau lebih yang sesuai dengan kriteria pencarian yang Anda tentukan. Fungsi SEARCH memungkin kan Anda menambahk an beberapa deret waktu terkait ke grafik dengan satu ekspresi. Grafik diperbarui secara dinamis untuk menyertakan metrik baru yang ditambahk an kemudian dan cocok dengan kriteria pencarian. Untuk informasi selengkapnya, lihat Gunakan ekspresi pencarian pada grafik. Anda tidak dapat membuat alarm berdasarkan ekspresi SEARCH. Hal ini karena ekspresi pencarian mengembalikan beberapa deret waktu, dan sebuah alarm yang dibuat berdasarkan ekspresi matematika		✓

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
			hanya dapat mengawasi satu deret waktu. Jika Anda masuk ke akun pemantauan dalam observabilitas CloudWatch lintas akun, fungsi SEARCH akan menemukan metrik di akun sumber dan akses pemantauan.		
SERVICE_QUOTA	TS yang merupa metrik penggu n	TS	Mengembalikan kuota layanan untuk metrik penggunaan yang diberikan. Anda dapat menggunakan ini untuk memvisualisasikan cara penggunaan saat ini dibandingkan dengan kuota, dan mengatur alarm yang mempering atkan Anda ketika mendekati kuota. Untuk informasi selengkap nya, lihat AWS metrik penggunaan.		✓

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
SLICE	(TS[], S, S) atau (TS[], S)	TS[] TS	Mengambil sebagian dari susunan deret waktu. Ini sangat berguna jika digabungk an dengan SORT. Sebagai contoh, Anda dapat mengecual ikan hasil teratas dari susunan deret waktu. Anda dapat menggunak an dua pendapat skalar untuk menentukan seperangkat deret waktu yang ingin dikembalikan. Kedua skalar menetapkan awal (inklusif) dan akhir (eksklusif) dari susunan untuk kembali. Susunan diberi indeks nol, sehingga deret waktu pertama dalam susunan adalah deret waktu 0. Atau, Anda dapat menentukan hanya satu nilai, dan CloudWatch mengembal ikan semua deret waktu dimulai dengan nilai itu.	SLICE(SORT(METRICS), SUM, DESC), 0, 10) mengembalikan 10 metrik dari susunan metrik dalam permintaa n yang memiliki nilai SUM tertinggi. SLICE(SORT(METRICS (), AVG, ASC), 5) mengurutkan susunan metrik berdasarkan statistik AVG, kemudian mengembalikan semua deret waktu kecuali untuk 5 dengan AVG terendah.	✓

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
SORT	(TS[], FUNCT SORT_(R) (TS[], FUNCT SORT_(R, S)	TS[]	Mengurutkan susunan deret waktu sesuai fungsi yang Anda tentukan. Fungsi yang Anda gunakan dapat berupa AVG, , MAX, atau SUM. Urutan pengurutan dapat berupa ASC untuk naik (nilai terendah terlebih dahulu) atau DESC untuk mengurutk an nilai lebih tinggi terlebih dahulu. Anda secara opsional dapat menentukan sebuah angka setelah urutan pengurutan yang bertindak sebagai batas. Sebagai contoh, menentukan batas 5 pengembalian hanya 5 deret waktu teratas dari pengurutan. Ketika fungsi matematik a ini ditampilkan pada grafik, label untuk setiap metrik dalam grafik juga diurutkan dan diberi nomor.	SORT(METRICS(), AVG, DESC, 10) menghitung nilai rata-rata setiap deret waktu, mengurutkan deret waktu dengan nilai tertinggi di awal pengurutan, dan mengembalikan hanya 10 deret waktu dengan rata-rata tertinggi. SORT(METRICS(), MAX, ASC) mengurutk an susunan metrik dengan statistik MAX, kemudian mengembal ikan semuanya dalam urutan naik.	✓

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
STDDEV	TS TS[]	S TS	STDDEV dari satu deret waktu mengembalikan skalar yang mewakili standar deviasi semua titik data dalam metrik. STDDEV dari susunan deret waktu mengembalikan satu rangkaian waktu. **Note** Kami menyarankan Anda untuk tidak menggunakan fungsi ini di CloudWatch alarm jika Anda ingin fungsi mengembalikan skalar. Misalnya, `STDDEV(m2)` Setiap kali alarm mengevaluasi apakah akan mengubah status, CloudWatch mencoba untuk mengambil jumlah titik data	m1/STDDEV(m1) STDDEV(METRICS())	✓

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
			yang lebih tinggi daripada nomor yang ditentukan sebagai Periode Evaluasi. Fungsi ini berperilaku berbeda ketika data tambahan diminta.		

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
SUM	TS TS[]	S TS	SUM dari satu deret waktu mengembalikan skalar yang mewakili jumlah nilai dari semua titik data dalam metrik. AVG dari susunan deret waktu mengembalikan satu deret waktu. **Note** Kami menyarankan Anda untuk tidak menggunak an fungsi ini di CloudWatch alarm jika Anda ingin fungsi mengembalikan skalar. Misalnya, `SUM(m1)`. Setiap kali alarm mengevaluasi apakah akan mengubah status, CloudWatch mencoba untuk mengambil jumlah titik data yang lebih tinggi	SUM(METRICS())/SUM (m1) SUM([m1, m2]) SUM(METRICS("error s"))/SUM(METRICS(" requests"))*100	✓

Fungsi	Argume	Jenis pengem ian*	Deskripsi	Contoh-contoh	Didukung untuk lintas akun?
			daripada nomor yang ditentukan sebagai Periode Evaluasi. Fungsi ini berperilaku berbeda ketika data tambahan diminta.		
TIME_SERIES	S	TS	Mengembalikan deret waktu non-jarang di mana setiap nilai diatur ke argumen skalar.	TIME_SERIES(MAX(m1)) TIME_SERIES(5*AVG(m1)) TIME_SERIES(10)	✓

*Menggunakan fungsi yang hanya mengembalikan angka skalar tidak benar, karena semua hasil akhir ekspresi harus berupa satu deret waktu atau susunan deret waktu. Alih-alih, gunakan fungsi ini sebagai bagian dari ekspresi yang lebih besar agar mengembalikan deret waktu.

Menggunakan ekspresi IF

Gunakan IF bersama operator perbandingan untuk memfilter titik data dari deret waktu, atau membuat deret waktu campuran yang terdiri atas beberapa deret waktu gabungan.

IF menggunakan argumen berikut:

```
IF(condition, trueValue, falseValue)
```

Ketentuan tersebut mengevaluasi menjadi FALSE jika nilai titik data ketentuan 0, dan menjadi TRUE jika nilai ketentuan kondisi adalah nilai lainnya, baik nilai tersebut positif maupun negatif. Jika ketentuannya adalah deret waktu, maka akan dievaluasi secara terpisah untuk setiap timestamp.

Berikut ini daftar sintaksis yang benar. Untuk setiap sintaksis ini, output adalah satu deret waktu.

- IF(TS ***Comparison Operator*** S, S | TS, S | TS)

  > **Note**
  >
  > Jika TRUE tetapi `metric2` tidak memiliki titik data yang sesuai, outputnya akan menjadi 0.
  > `TS comparison operator S`

- IF(TS, TS, TS)
- IF(TS, S, TS)
- IF(TS, TS, S)
- IF(TS, S, S)
- IF(S, TS, TS)

Bagian berikut memberikan detail lebih lanjut dan contoh untuk sintaksis ini.

IF(TS ***Comparison Operator*** S, scalar2 | metric2, scalar3 | metric3)

Nilai deret waktu output yang sesuai:

- memiliki nilai scalar2 atau metric2, jika TS ***Comparison Operator*** S TRUE
- memiliki nilai scalar3 atau metric3, jika TS ***Comparison Operator*** S FALSE
- memiliki nilai 0 jika ***Operator Perbandingan*** TS TRUE dan titik data yang sesuai dalam metrik2 tidak ada.
- memiliki nilai 0 jika ***Operator Perbandingan*** TS adalah FALSE dan titik data yang sesuai dalam metrik3 tidak ada.
- adalah deret waktu kosong, jika titik data yang sesuai tidak ada dalam metric3, atau jika skalar3/metric3 dihilangkan dari ekspresi

IF(metric1, metric2, metric3)

Untuk setiap titik data metric1, nilai deret waktu output yang sesuai:

- memiliki nilai metric2, jika titik data yang sesuai dari metric1 TRUE.
- memiliki nilai metric3, jika titik data yang sesuai dari metric1 FALSE.

- memiliki nilai 0, jika titik data yang sesuai dari metric1 TRUE dan titik data yang sesuai tidak ada dalam metric2.
- dijatuhkan, jika titik data metrik1 yang sesuai adalah FALSE dan titik data yang sesuai tidak ada dalam metrik3
- dijatuhkan, jika titik data yang sesuai dari metric1 adalah FALSE dan metric3 dihilangkan dari ekspresi.
- dijatuhkan, jika titik data metrik1 yang sesuai hilang.

Tabel berikut menunjukkan contoh untuk sintaks ini.

Metrik atau fungsi	Nilai-nilai
(metric1)	[1, 1, 0, 0, -]
(metric2)	[30, -, 0, 0, 30]
(metric3)	[0, 0, 20, -, 20]
IF(metric1, metric2, metric3)	[30, 0, 20, 0, -]

IF(metric1, scalar2, metric3)

Untuk setiap titik data metric1, nilai deret waktu output yang sesuai:

- memiliki nilai scalar2, jika titik data yang sesuai dari metric1 TRUE.
- memiliki nilai metric3, jika titik data yang sesuai dari metric1 FALSE.
- diabaikan, jika titik data yang sesuai dari metric1 FALSE dan titik data yang sesuai tidak ada pada metric3, atau jika metric3 dihilangkan dari ekspresi.

Metrik atau fungsi	Nilai-nilai
(metric1)	[1, 1, 0, 0, -]
scalar2	5
(metric3)	[0, 0, 20, -, 20]

Metrik atau fungsi	Nilai-nilai
IF(metric1, scalar2, metric3)	[5, 5, 20, -, -]

IF(metric1, metric2, scalar3)

Untuk setiap titik data metric1, nilai deret waktu output yang sesuai:

- memiliki nilai metric2, jika titik data yang sesuai dari metric1 TRUE.
- memiliki nilai scalar3, jika titik data yang sesuai dari metric1 FALSE.
- memiliki nilai 0, jika titik data yang sesuai dari metric1 TRUE dan titik data yang sesuai tidak ada dalam metric2.
- diabaikan jika titik data yang sesuai pada metric1 tidak ada.

Metrik atau fungsi	Nilai-nilai
(metric1)	[1, 1, 0, 0, -]
(metric2)	[30, -, 0, 0, 30]
scalar3	5
IF(metric1, metric2, scalar3)	[30, 0, 5, 5, -]

IF(scalar1, metric2, metric3)

Nilai deret waktu output yang sesuai:

- memiliki nilai metric2, jika scalar1 TRUE.
- memiliki nilai metric3, jika scalar1 FALSE.
- adalah deret waktu kosong, jika metrik3 dihilangkan dari ekspresi.

Gunakan contoh kasus untuk ekspresi IF

Contoh berikut menggambarkan kemungkinan penggunaan fungsi IF.

- Untuk menampilkan hanya nilai metrik rendah:

  IF(metric1<400, metric1)

- Untuk mengubah setiap titik data dalam metrik menjadi satu dari dua nilai, untuk menunjukkan tinggi dan rendah relatif dari metrik awal:

  IF(metric1<400, 10, 2)

- Agar menampilkan 1 untuk setiap timestamp di mana latensi berada di atas ambang batas, dan menampilkan 0 untuk semua titik data lainnya:

  IF(latency>threshold, 1, 0)

Gunakan matematika metrik dengan operasi GetMetricData API

Anda dapat menggunakan `GetMetricData` untuk melakukan penghitungan menggunakan ekspresi matematika, dan juga mengambil kumpulan besar data metrik dalam satu panggilan API. Untuk informasi lebih lanjut, lihat GetMetricData.

Deteksi anomali pada matematika metrik

Deteksi anomali pada matematika metrik adalah fitur yang dapat Anda gunakan untuk membuat alarm deteksi anomali pada metrik tunggal dan hasil ekspresi matematika metrik. Anda dapat menggunakan ekspresi-ekspresi ini untuk membuat grafik yang menggambarkan pita deteksi anomali. Fitur ini mendukung fungsi aritmatika dasar, operator perbandingan dan logika, dan sebagian besar fungsi lainnya.

Deteksi anomali pada matematika metrik tidak mendukung fungsi berikut:

- Ekspresi yang berisi lebih dari satu ANOMALY_DETECTION_BAND di garis yang sama.
- Ekspresi yang berisi lebih dari 10 metrik atau ekspresi matematika.
- Ekspresi yang berisi ekspresi METRIK.
- Ekspresi yang berisi fungsi SEARCH.
- Ekspresi yang menggunakan fungsi DP_PERF_INSIGHTS.
- Ekspresi yang menggunakan metrik dengan periode yang berbeda.
- Detektor anomali matematika metrik yang menggunakan metrik resolusi tinggi sebagai input.

Untuk informasi selengkapnya tentang fitur ini, lihat Menggunakan deteksi CloudWatch anomali di CloudWatch Panduan Pengguna Amazon.

Gunakan ekspresi pencarian pada grafik

Ekspresi pencarian adalah jenis ekspresi matematika yang dapat Anda tambahkan ke CloudWatch grafik. Ekspresi penelusuran memungkinkan Anda untuk menambahkan beberapa metrik terkait ke grafik dengan cepat. Ini juga memungkinkan Anda untuk membuat grafik dinamis yang secara otomatis menambahkan metrik yang sesuai ke tampilan mereka, meskipun metrik tersebut tidak ada ketika Anda pertama kali membuat grafik.

Sebagai contoh, Anda dapat membuat ekspresi pencarian yang menampilkan `AWS/EC2` `CPUUtilization` metrik untuk semua instans di Wilayah. Jika nanti Anda meluncurkan instans yang baru, `CPUUtilization` dari instans yang baru secara otomatis ditambahkan ke grafik.

Ketika Anda menggunakan ekspresi pencarian pada grafik, pencarian menemukan ekspresi pencarian dalam nama metrik, namespace, nama dimensi, dan nilai dimensi. Anda dapat menggunakan operator Boolean untuk pencarian yang lebih rumit dan kuat. Ekspresi pencarian hanya dapat menemukan metrik yang telah melaporkan data dalam dua minggu terakhir.

Anda tidak dapat membuat sebuah alarm berdasarkan ekspresi SEARCH. Hal ini karena ekspresi pencarian mengembalikan beberapa deret waktu, dan sebuah alarm yang dibuat berdasarkan ekspresi matematika hanya dapat mengawasi satu deret waktu.

Jika Anda menggunakan akun pemantauan dalam pengamatan CloudWatch lintas akun, ekspresi penelusuran Anda dapat menemukan metrik di akun sumber yang ditautkan ke akun pemantauan tersebut.

Topik

- CloudWatch sintaks ekspresi pencarian
- CloudWatch contoh ekspresi pencarian
- Buat CloudWatch grafik dengan ekspresi pencarian

CloudWatch sintaks ekspresi pencarian

Ekspresi pencarian yang benar memiliki format berikut.

```
SEARCH(' {Namespace, DimensionName1, DimensionName2, ...} SearchTerm', 'Statistic')
```

Sebagai contoh:

```
SEARCH('{AWS/EC2,InstanceId} MetricName="CPUUtilization"', 'Average')
```

- Bagian pertama kueri setelah kata SEARCH, yang diapik dengan rangka melengkung, adalah skema metrik yang akan dicari. Skema metrik memuat namespace metrik dan satu nama dimensi atau lebih. Termasuk skema metrik dalam kueri pencarian yang bersifat opsional. Jika ditentukan, skema metrik harus memuat namespace dan secara opsional dapat memuat satu nama dimensi atau lebih yang benar di namespace tersebut.

  Anda tidak perlu menggunakan tanda kutip di dalam skema metrik kecuali jika namespace atau nama dimensi mencakup spasi atau karakter bukan alfanumerik. Dalam hal ini, Anda harus mengapit nama yang memuat karakter tersebut dengan tanda kutip ganda.
- SearchTerm juga bersifat opsional, tetapi pencarian yang benar harus memuat skema metrik, SearchTerm, atau keduanya. SearchTerm biasanya berisi satu ID akun atau lebih, nama metrik, atau nilai dimensi. SearchTerm dapat menyertakan beberapa istilah untuk dicari, baik cocok sebagian maupun yang benar-benar cocok. Juga dapat memuat operator Boolean.

  Menggunakan ID akun hanya SearchTerm berfungsi di akun yang disiapkan sebagai akun pemantauan untuk pengamatan CloudWatch lintas akun. Sintaks untuk ID akun SearchTerm adalah :aws.AccountId = "*444455556666*". Anda juga dapat menggunakan 'LOCAL' untuk menentukan akun pemantauan itu sendiri: :aws.AccountId = 'LOCAL'

  Untuk informasi selengkapnya, lihat CloudWatch observabilitas lintas akun.

  SearchTerm dapat mencakup satu penanda atau lebih, seperti MetricName= sebagaimana dalam contoh ini, tetapi menggunakan penanda yang tidak diperlukan.

  Skema metrik dan SearchTerm harus diapit bersama-sama dalam sepasang tanda kutip tunggal.
- StatisticIni adalah nama CloudWatch statistik yang valid. Ini harus diapit dengan satu tanda kutip. Untuk informasi selengkapnya, lihat Statistik.

Contoh sebelumnya mencari namespace AWS/EC2 untuk metrik yang memiliki InstanceId sebagai nama dimensi. Ini mengembalikan semua metrik CPUUtilization yang ditemukan, dengan grafik yang menunjukkan statistik Average.

Ekspresi pencarian hanya dapat menemukan metrik yang telah melaporkan data dalam dua minggu terakhir.

Batas ekspresi pencarian

Ukuran kueri ekspresi pencarian maksimum 1024 karakter. Anda dapat memiliki sebanyak 100 ekspresi pencarian pada satu grafik. Grafik dapat menampilkan sebanyak 500 deret waktu.

CloudWatch ekspresi pencarian: Tokenisasi

Saat Anda menentukan`SearchTerm`, fungsi pencarian akan mencari token, yang merupakan substring yang secara CloudWatch otomatis dihasilkan dari nama metrik lengkap, nama dimensi, nilai dimensi, dan ruang nama. CloudWatch menghasilkan token yang dibedakan dengan kapitalisasi kasus unta dalam string asli. Karakter numerik juga berfungsi sebagai awal token baru, dan karakter bukan alfanumerik berfungsi sebagai pembatas, yang membuat token sebelum dan sesudah karakter bukan alfanumerik.

String berkelanjutan dengan jenis karakter pembatas token yang sama menghasilkan satu token.

Semua token dibuat dalam huruf kecil. Tabel berikut menunjukkan beberapa contoh token yang dibuat.

String awal	Token yang dihasilkan
CustomCount1	`customcount1` , `custom`, `count`, `1`
SDBFailure	`sdbfailure` , `sdb`, `failure`
Project2-trial333	`project2trial333` , `project`, `2`, `trial`, `333`

CloudWatch ekspresi pencarian: Kecocokan sebagian

Saat Anda menentukan a`SearchTerm`, istilah pencarian juga diberi token. CloudWatchmenemukan metrik berdasarkan kecocokan sebagian, yang merupakan kecocokan dari satu token yang dihasilkan dari istilah pencarian ke token tunggal yang dihasilkan dari nama metrik, namespace, nama dimensi, atau nilai dimensi.

Pencarian cocok sebagian untuk mencocokkan satu token bersifat tidak peka huruf besar dan kecil. Sebagai contoh, menggunakan salah satu istilah pencarian berikut dapat menampilkan metrik `CustomCount1`:

- `count`

- `Count`
- `COUNT`

Namun demikian, menggunakan `couNT` sebagai istilah pencarian tidak menemukan `CustomCount1` karena penulisan huruf besar dalam istilah pencarian `couNT` diberi token menjadi `cou` dan `NT`.

Pencarian juga dapat mencocokkan token gabungan, yang merupakan beberapa token yang muncul secara berurutan dalam nama aslinya. Untuk mencocokkan token gabungan, pencarian peka huruf besar dan kecil. Sebagai contoh, jika istilah awal `CustomCount1`, mencari `CustomCount` atau `Count1` berhasil, tetapi mencari `customcount` atau `count1` tidak berhasil.

CloudWatch ekspresi pencarian: Pencocokan yang tepat

Anda dapat menentukan pencarian untuk menemukan istilah yang benar-benar cocok saja dari istilah pencarian menggunakan tanda kutip ganda di bagian dari istilah pencarian yang memerlukan istilah yang benar-benar cocok. . Tanda kutip ganda ini diapit dalam tanda kutip tunggal yang digunakan di sekitar seluruh istilah pencarian. Sebagai contoh, **`SEARCH(' {MyNamespace}, "CustomCount1" ', 'Maximum')`** menemukan string `CustomCount1` yang benar-benar sama jika ada sebagai nama metrik, nama dimensi, atau nilai dimensi di namespace yang disebut `MyNamespace`. Namun demikian, pencarian **`SEARCH(' {MyNamespace}, "customcount1" ', 'Maximum')`** atau **`SEARCH(' {MyNamespace}, "Custom" ', 'Maximum')`** tidak menemukan string ini.

Anda dapat menggabungkan istilah yang cocok sebagian dan istilah yang benar-benar cocok dalam satu ekspresi pencarian. Sebagai contoh, **`SEARCH(' {AWS/NetworkELB, LoadBalancer} "ConsumedLCUs" OR flow ', 'Maximum')`** mengembalikan metrik Penyeimbang Beban Elastis yang disebut `ConsumedLCUs` serta semua metrik atau dimensi Penyeimbang Beban Elastis yang memuat token `flow`.

Penggunaan istilah yang benar-benar cocok juga merupakan cara yang baik untuk menemukan nama dengan karakter khusus, seperti karakter bukan alfanumerik atau spasi, seperti dalam contoh berikut.

```
SEARCH(' {"My Namespace", "Dimension@Name"}, "Custom:Name[Special_Characters" ',
 'Maximum')
```

CloudWatch ekspresi pencarian: Tidak termasuk skema metrik

Semua contoh yang ditunjukkan sejauh ini mencakup skema metrik, dalam rangka melengkung. Pencarian yang menghilangkan skema metrik juga benar.

Sebagai contoh, **`SEARCH(' "CPUUtilization" ', 'Average')`** mengembalikan semua nama metrik, nama dimensi, nilai dimensi, dan namespace yang benar-benar cocok untuk string `CPUUtilization`. Di ruang nama AWS metrik, ini dapat mencakup metrik dari beberapa layanan termasuk Amazon EC2, Amazon ECS, dan lainnya. SageMaker

Untuk mempersempit pencarian ini menjadi hanya satu AWS layanan, praktik terbaik adalah menentukan namespace dan dimensi apa pun yang diperlukan dalam skema metrik, seperti pada contoh berikut. Meskipun mempersempit pencarian menjadi namespace `AWS/EC2`, ini masih akan mengembalikan hasil metrik lain jika Anda telah menetapkan `CPUUtilization` sebagai nilai dimensi untuk metrik tersebut.

```
SEARCH(' {AWS/EC2, InstanceType} "CPUUtilization" ', 'Average')
```

Atau, Anda dapat menambahkan namespace di `SearchTerm` dalam contoh berikut. Namun dalam contoh ini, pencarian akan cocok dengan string `AWS/EC2`, meskipun ini merupakan nama atau nilai dimensi kustom.

```
SEARCH(' "AWS/EC2" MetricName="CPUUtilization" ', 'Average')
```

CloudWatch ekspresi pencarian: Menentukan nama properti dalam pencarian

Pencarian yang benar-benar cocok berikut untuk `"CustomCount1"` mengembalikan semua metrik dengan tepat namanya.

```
SEARCH(' "CustomCount1" ', 'Maximum')
```

Tapi itu juga mengembalikan metrik dengan nama dimensi, nilai dimensi, atau namespace dari `CustomCount1`. Untuk menyusun pencarian lebih lanjut, Anda dapat menetapkan nama properti dari jenis objek yang ingin Anda temukan dalam pencarian Anda. Contoh berikut mencari semua namespace dan mengembalikan metrik yang disebut `CustomCount1`.

```
SEARCH(' MetricName="CustomCount1" ', 'Maximum')
```

Anda juga dapat menggunakan namespace dan nama dimensi/pasangan nilai sebagai nama properti, seperti dalam contoh berikut. Contoh pertama juga menggambarkan bahwa Anda juga dapat menggunakan nama properti dengan pencarian yang cocok sebagian.

```
SEARCH(' InstanceType=micro ', 'Average')
```

```
SEARCH(' InstanceType="t2.micro" Namespace="AWS/EC2" ', 'Average')
```

CloudWatch ekspresi pencarian: Karakter non-alfanumerik

Karakter non-alfanumerik berfungsi sebagai pembatas, dan menandai di mana nama metrik, dimensi, namespace, dan istilah pencarian akan dipisahkan menjadi token. Ketika istilah diberi token, karakter non-alfanumerik dihapus dan tidak muncul dalam token. Sebagai contoh, `Network-Errors_2` menghasilkan token `network`, `errors`, dan 2.

Istilah pencarian Anda dapat mencakup karakter non-alfanumerik. Jika karakter ini muncul dalam istilah pencarian Anda, maka karakter tersebut dapat menentukan token gabungan dalam karakter yang cocok sebagian. Sebagai contoh, semua pencarian berikut akan menemukan metrik yang disebut `Network-Errors-2` atau `NetworkErrors2`.

```
network/errors
network+errors
network-errors
Network_Errors
```

Ketika Anda melakukan pencarian nilai yang benar-benar sama, karakter non-alfanumerik yang digunakan dalam pencarian yang benar-benar sama harus merupakan karakter yang benar yang muncul dalam string yang dicari. Sebagai contoh, jika Anda ingin menemukan `Network-Errors-2`, yang mencari `"Network-Errors-2"` berhasil, tetapi pencarian `"Network_Errors_2"` tidak berhasil.

Ketika Anda melakukan pencarian yang benar-benar sama, karakter berikut harus hilang dengan garis miring terbalik.

```
" \ ( )
```

Sebagai contoh, untuk menemukan nama metrik `Europe\France Traffic(Network)` berdasarkan karakter yang benar-benar sama, gunakan istilah pencarian **`"Europe\\France Traffic\(Network\)"`**

CloudWatch ekspresi pencarian: Operator Boolean

Pencarian mendukung penggunaan operator Boolean `AND`, `OR`, dan `NOT` dalam `SearchTerm`. Operator Boolean diapit dalam tanda kutip tunggal yang Anda gunakan untuk mengapit seluruh istilah

pencarian. Operator boolean peka huruf besar dan kecil, sehingga `and`, `or`, dan `not` tidak benar sebagai operator Boolean.

Anda dapat menggunakan AND secara eksplisit dalam pencarian Anda, seperti **`SEARCH('{AWS/EC2,InstanceId} network AND packets', 'Average')`**. Tidak menggunakan operator Boolean di antara istilah pencarian secara implisit mencari seolah-olah ada operator AND, sehingga **`SEARCH(' {AWS/EC2,InstanceId} network packets ', 'Average')`** memberikan hasil pencarian yang sama.

Gunakan NOT untuk mengecualikan subset data dari hasil. Sebagai contoh, **`SEARCH(' {AWS/EC2,InstanceId} MetricName="CPUUtilization" NOT i-1234567890123456 ', 'Average')`** mengembalikan `CPUUtilization` untuk semua instans Anda, kecuali untuk instans `i-1234567890123456`. Anda juga dapat menggunakan klausa NOT sebagai satu-satunya istilah pencarian. Sebagai contoh, **`SEARCH( 'NOT Namespace=AWS ', 'Maximum')`** menghasilkan semua metrik kustom Anda (metrik dengan namespace yang tidak mencakup AWS).

Anda dapat menggunakan beberapa frasa NOT dalam kueri. Sebagai contoh, **`SEARCH(' {AWS/EC2,InstanceId} MetricName="CPUUtilization" NOT "ProjectA" NOT "ProjectB" ', 'Average')`** mengembalikan `CPUUtilization` dari semua instans di Wilayah, kecuali untuk contoh dengan nilai dimensi `ProjectA` atau `ProjectB`.

Anda dapat menggabungkan operator Boolean untuk pencarian yang lebih kuat dan detail, seperti dalam contoh berikut. Gunakan tanda kurung untuk mengelompokkan operator.

Kedua contoh berikutnya mengembalikan semua nama metrik yang memuat `ReadOps` dari namespace EC2 dan EBS.

```
SEARCH(' (EC2 OR EBS) AND MetricName=ReadOps ', 'Maximum')
```

```
SEARCH(' (EC2 OR EBS) MetricName=ReadOps ', 'Maximum')
```

Contoh berikut mempersempit pencarian sebelumnya menjadi hasil yang hanya mencakup `ProjectA`, yang dapat berupa nilai dimensi.

```
SEARCH(' (EC2 OR EBS) AND ReadOps AND ProjectA ', 'Maximum')
```

Contoh berikut menggunakan pengelompokan bersarang. Ini mengembalikan metrik Lambda untuk `Errors` dari semua fungsi, dan `Invocations` dari fungsi dengan nama yang menyertakan string `ProjectA` atau `ProjectB`.

```
SEARCH(' {AWS/Lambda,FunctionName} MetricName="Errors" OR (MetricName="Invocations" AND
 (ProjectA OR ProjectB)) ', 'Average')
```

CloudWatch ekspresi pencarian: Menggunakan ekspresi matematika

Anda dapat menggunakan ekspresi pencarian di dalam ekspresi matematika dalam grafik.

Sebagai contoh, **`SUM(SEARCH(' {AWS/Lambda, FunctionName} MetricName="Errors" ', 'Sum'))`** mengembalikan jumlah metrik `Errors` dari semua fungsi Lambda Anda.

Menggunakan baris terpisah untuk ekspresi pencarian dan ekspresi matematika Anda dapat memberikan hasil yang lebih berguna. Sebagai contoh, misal Anda menggunakan dua ekspresi berikut dalam grafik. Baris pertama menampilkan baris `Errors` secara terpisah untuk setiap fungsi Lambda Anda. ID dari ekspresi ini adalah `e1`. Baris kedua menambahkan baris lain yang menunjukkan jumlah kesalahan dari semua fungsi.

```
SEARCH(' {AWS/Lambda, FunctionName}, MetricName="Errors" ', 'Sum')
SUM(e1)
```

CloudWatch contoh ekspresi pencarian

Contoh berikut menggambarkan penggunaan ekspresi pencarian dan sintaks yang lebih banyak. Mari kita mulai dengan pencarian untuk `CPUUtilization` di seluruh instans di Wilayah dan kemudian lihat variasi.

Contoh ini menampilkan satu baris untuk setiap instans di Wilayah, yang menampilkan metrik `CPUUtilization` dari namespace `AWS/EC2`.

```
SEARCH(' {AWS/EC2,InstanceId} MetricName="CPUUtilization" ', 'Average')
```

Mengubah `InstanceId` menjadi `InstanceType` akan mengubah grafik guna menampilkan satu baris untuk masing-masing tipe instans yang digunakan di Wilayah. Data dari semua instans setiap jenis digabungkan ke dalam satu baris untuk tipe instans tersebut.

```
SEARCH(' {AWS/EC2,InstanceType} MetricName="CPUUtilization" ', 'Average')
```

Contoh berikut menggabungkan `CPUUtilization` menurut tipe instans dan menampilkan satu baris untuk setiap tipe instans yang mencakup string `micro`.

```
SEARCH('{AWS/EC2,InstanceType} InstanceType=micro MetricName="CPUUtilization" ',
 'Average')
```

Contoh ini mempersempit contoh sebelumnya, mengubah `InstanceType` pada pencarian yang benar-benar sama untuk instans t2.micro.

```
SEARCH('{AWS/EC2,InstanceType} InstanceType="t2.micro" MetricName="CPUUtilization" ',
 'Average')
```

Pencarian berikut menghapus bagian `{metric schema}` dari kueri, sehingga metrik `CPUUtilization` dari semua namespace muncul di grafik. Ini dapat mengembalikan beberapa hasil karena grafik mencakup beberapa baris untuk `CPUUtilization` metrik dari setiap AWS layanan, digabungkan sepanjang dimensi yang berbeda.

```
SEARCH('MetricName="CPUUtilization" ', 'Average')
```

Untuk sedikit mempersempit hasil ini, Anda dapat menentukan dua namespace metrik tertentu.

```
SEARCH('MetricName="CPUUtilization" AND ("AWS/ECS" OR "AWS/ES") ', 'Average')
```

Contoh sebelumnya merupakan satu-satunya cara untuk melakukan pencarian beberapa namespace khusus dengan satu kueri pencarian, karena Anda hanya dapat menentukan satu skema metrik dalam setiap kueri. Namun demikian, untuk menambahkan struktur lain, Anda dapat menggunakan dua kueri di grafik, seperti dalam contoh berikut. Contoh ini juga menambahkan struktur lain dengan menentukan dimensi untuk digunakan guna menggabungkan data pada Amazon ECS.

```
SEARCH('{AWS/ECS ClusterName}, MetricName="CPUUtilization" ', 'Average')
SEARCH(' {AWS/EBS} MetricName="CPUUtilization" ', 'Average')
```

Contoh berikut mengembalikan metrik Penyeimbang Beban Elastis yang disebut `ConsumedLCUs`, serta semua metrik atau dimensi Penyeimbang Beban Elastis yang memuat token `flow`.

```
SEARCH('{AWS/NetworkELB, LoadBalancer} "ConsumedLCUs" OR flow ', 'Maximum')
```

Contoh berikut menggunakan pengelompokan bersarang. Ini mengembalikan metrik Lambda untuk `Errors` dari semua fungsi dan `Invocations` dari fungsi dengan nama yang menyertakan string `ProjectA` atau `ProjectB`.

```
SEARCH('{AWS/Lambda,FunctionName} MetricName="Errors" OR (MetricName="Invocations" AND
 (ProjectA OR ProjectB)) ', 'Average')
```

Contoh berikut menampilkan semua metrik kustom Anda, tidak termasuk metrik yang dihasilkan oleh layanan AWS .

```
SEARCH('NOT Namespace=AWS ', 'Average')
```

Contoh berikut menampilkan metrik dengan nama metrik, namespace, nama dimensi, dan nilai dimensi yang memuat string `Errors` sebagai bagian dari nama mereka.

```
SEARCH('Errors', 'Average')
```

Contoh berikut mempersempit pencarian hingga benar-benar cocok. Sebagai contoh, pencarian ini menemukan nama metrik `Errors` tetapi bukan metrik yang disebut `ConnectionErrors` atau `errors`.

```
SEARCH(' "Errors" ', 'Average')
```

Contoh berikut menunjukkan cara menentukan nama yang memuat spasi atau karakter khusus di bagian skema metrik istilah pencarian.

```
SEARCH('{"Custom-Namespace", "Dimension Name With Spaces"}, ErrorCount ', 'Maximum')
```

CloudWatch contoh ekspresi pencarian observabilitas lintas akun

CloudWatch contoh observabilitas lintas akun

Jika Anda masuk ke akun yang disiapkan sebagai akun pemantauan dalam pengamatan CloudWatch lintas akun, Anda dapat menggunakan fungsi SEARCH untuk mengembalikan metrik dari akun sumber tertentu. Untuk informasi selengkapnya, lihat CloudWatch observabilitas lintas akun.

Contoh berikut mengambil semua metrik Lambda dari akun dengan ID akun 111122223333.

```
SEARCH(' AWS/Lambda :aws.AccountId = "111122223333" ', 'Average')
```

Contoh berikut mengambil semua metrik `AWS/EC2` dari dua akun: 111122223333 dan 777788889999.

```
SEARCH(' AWS/EC2 :aws.AccountId = ("111122223333" OR "777788889999") ', 'Average')
```

Contoh berikut mengambil semua metrik `AWS/EC2` dari akun sumber 111122223333 dan dari akun pemantauan itu sendiri.

```
SEARCH(' AWS/EC2 :aws.AccountId = ("111122223333" OR 'LOCAL') ', 'Average')
```

Contoh berikut mengambil `SUM` metrik `MetaDataToken` dari akun `444455556666` dengan dimensi `InstanceId`.

```
SEARCH('{AWS/EC2,InstanceId} :aws.AccountId=444455556666 MetricName=\"MetadataNoToken
\"','Sum')
```

Buat CloudWatch grafik dengan ekspresi pencarian

Di CloudWatch konsol, Anda dapat mengakses kemampuan penelusuran saat menambahkan grafik ke dasbor, atau dengan menggunakan tampilan Metrik.

Anda tidak dapat membuat alarm berdasarkan ekspresi SEARCH. Hal ini karena ekspresi pencarian mengembalikan beberapa deret waktu, dan sebuah alarm yang dibuat berdasarkan ekspresi matematika hanya dapat mengawasi satu deret waktu.

Untuk menambahkan grafik dengan ekspresi pencarian ke dasbor yang sudah ada

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor dan pilih dasbor.
3. Pilih Tambahkan widget.
4. Pilih salah satu Garis atau Wilayah yang ditumpuk dan pilih Konfigurasi.
5. Di tab Metrik grafik, pilih Tambahkan ekspresi matematika.
6. Untuk Detail, masukkan ekspresi pencarian yang Anda inginkan. Sebagai contoh, **`SEARCH('{AWS/EC2,InstanceId} MetricName="CPUUtilization"', 'Average')`**.
7. (Opsional) Untuk menambahkan ekspresi pencarian atau ekspresi matematika lainnya ke grafik, pilih Tambahkan ekspresi matematika
8. (Opsional) Setelah menambahkan ekspresi pencarian, Anda dapat menentukan label dinamis agar muncul di legenda grafik untuk setiap metrik. Label dinamis menampilkan statistik tentang metrik, dan secara otomatis diperbarui ketika dasbor atau grafik disegarkan. Untuk menambahkan label dinamis, pilih Metrik grafik dan kemudian Label dinamis.

Secara bawaan, nilai dinamis yang Anda tambahkan ke label muncul di awal label. Anda kemudian dapat mengklik nilai Label untuk metrik untuk menyunting label. Untuk informasi selengkapnya, lihat Gunakan label dinamis.

9. (Opsional) Untuk menambahkan metrik tunggal ke grafik, pilih tab Semua metrik dan telusuri ke dalam metrik yang Anda inginkan.
10. (Opsional) Untuk mengubah rentang waktu yang ditunjukkan pada grafik, pilih salah satu kustom di bagian atas grafik atau salah satu periode waktu di sebelah kiri kustom.
11. (Opional) Keterangan horizontal membantu pengguna dasbor dengan cepat melihat kapan suatu metrik naik ke tingkat tertentu atau apakah metrik berada dalam rentang yang telah ditentukan. Untuk menambahkan keterangan horizontal, pilih tab Opsi grafik dan kemudian Tambahkan keterangan horizontal:

    a. Untuk Label, masukkan label untuk keterangan.

    b. Untuk Nilai, masukkan nilai metrik tempat keterangan horizontal muncul.

    c. Untuk Isi, tentukan apakah akan menggunakan bayangan pengisian dengan keterangan ini. Sebagai contoh, pilih `Above` atau `Below` untuk area terkait yang akan diisi. Jika Anda menentukan `Between`, bidang `Value` lainnya muncul, dan area grafik di antara kedua nilai diisi.

    d. Untuk Sumbu, tentukan apakah angka di `Value` mengacu pada metrik yang terkait dengan sumbu y kiri atau sumbu y kanan jika grafik mencakup beberapa metrik.

    Anda dapat mengubah warna pengisian keterangan dengan memilih kotak warna di kolom kiri keterangan.

    Ulangi langkah-langkah ini untuk menambahkan beberapa keterangan horizontal ke grafik yang sama.

    Untuk menyembunyikan keterangan, kosongkan kotak centang di kolom kiri untuk keterangan tersebut.

    Untuk menghapus keterangan, pilih x dalam kolom Tindakan.
12. (Opsional) Keterangan vertikal membantu Anda menandai tonggak penting dalam grafik, seperti peristiwa operasi atau awal dan akhir deployment. Untuk menambahkan keterangan vertikal, pilih Opsi grafik dan kemudian Tambahkan keterangan vertikal:

a. Untuk Label, masukkan label untuk keterangan. Untuk menampilkan hanya tanggal dan waktu pada keterangan, biarkan bidang Label kosong.

b. Untuk Tanggal, tentukan tanggal dan waktu keterangan vertikal ditampilkan.

c. Untuk Isi, tentukan apakah akan menggunakan bayangan pengisian sebelum atau setelah keterangan vertikal atau di antara dua keterangan vertikal. Sebagai contoh, pilih `Before` atau `After` untuk area terkait yang akan diisi. Jika Anda menentukan `Between`, bidang `Date` lainnya muncul, dan area grafik di antara kedua nilai diisi.

Ulangi langkah-langkah ini untuk menambahkan beberapa keterangan vertikal ke grafik yang sama.

Untuk menyembunyikan keterangan, kosongkan kotak centang di kolom kiri untuk keterangan tersebut.

Untuk menghapus keterangan, pilih x dalam kolom Tindakan.

13. Pilih Buat widget.
14. Pilih Simpan dasbor.

Untuk menggunakan tampilan Metrik guna membuat grafik metrik yang dicari

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik.
3. Dalam bidang pencarian, masukkan token untuk dicari: misalnya, **`cpuutilization t2.small`**.

   Hasil yang cocok dengan pencarian Anda akan muncul.
4. Untuk membuat grafik semua metrik yang cocok dengan pencarian Anda, pilih Pencarian grafik.

   atau

   Untuk menyempurnakan pencarian Anda, pilih salah satu namespace yang muncul di hasil pencarian Anda.
5. Jika Anda memilih namespace untuk mempersempit hasil, Anda dapat melakukan hal berikut:

   a. Untuk membuat grafik satu metrik atau lebih, pilih kotak centang di sebelah setiap metrik. Untuk memilih semua metrik, pilih kotak centang di baris judul tabel.

b. Untuk menyempurnakan pencarian Anda, arahkan kursor ke nama metrik dan pilih Tambahkan ke pencarian atau Cari ini saja.

c. Untuk melihat bantuan metrik, pilih nama metrik dan pilih Apa ini?.

Metrik yang dipilih muncul di grafik.

6. (Opsional) Pilih salah satu tombol di bilah pencarian untuk menyunting bagian dari istilah pencarian tersebut.
7. (Opsional) Untuk menambahkan grafik ke dasbor, pilih Tindakan dan kemudian Tambahkan ke dasbor.

Mendapatkan statistik untuk metrik

CloudWatch definisi statistik

Statistik adalah agregasi data metrik selama periode waktu tertentu. Saat Anda membuat grafik atau mengambil statistik untuk metrik, Anda menentukan Periode waktu, seperti lima menit, yang akan digunakan untuk menghitung setiap nilai statistik. Sebagai contoh, jika Periode lima menit, Jumlah adalah jumlah semua nilai sampel yang dikumpulkan selama periode lima menit, sedangkan Minimum adalah nilai terendah yang dikumpulkan selama periode lima menit.

CloudWatch mendukung statistik berikut untuk metrik.

- SampleCountadalah jumlah titik data selama periode tersebut.
- Jumlah adalah jumlah nilai dari semua titik data yang dikumpulkan selama periode tersebut.
- Rata-rata adalah nilai dari `Sum/SampleCount` selama periode tertentu.
- Minimum adalah nilai terendah yang diamati selama periode tertentu.
- Maksimum adalah nilai tertinggi yang diamati selama periode tertentu.
- Persentil (p) menunjukkan posisi relatif nilai dalam set data. Sebagai contoh, p95 adalah persentil ke-95 dan berarti bahwa 95 persen data dalam periode lebih rendah dari nilai ini dan 5 persen data lebih tinggi dari nilai ini. Persentil membantu Anda mendapatkan pemahaman yang lebih baik tentang distribusi data metrik Anda.
- Rata-rata terpangkas (TM) adalah rata-rata dari semua nilai yang berada di antara dua batas yang ditentukan. Nilai di luar batas diabaikan ketika mean dihitung. Anda menetapkan batasnya dengan satu atau dua angka antara 0 dan 100, hingga 10 tempat desimal. Angka-angka dapat berupa nilai

absolut atau persentase. Sebagai contoh, tm90 menghitung rata-rata setelah menghapus 10% titik data dengan nilai tertinggi. TM(2%:98%) menghitung rata-rata setelah menghapus titik data terendah 2% dan titik data tertinggi 2%. TM (150:1000) menghitung rata-rata setelah menghapus semua titik data yang lebih rendah dari atau sama dengan 150, atau lebih tinggi dari 1000.

- Rata-rata antar kuartil (IQM) adalah rata-rata terpangkas dari rentang antar kuartil, atau 50% nilai tengah. Ini setara dengan TM(25%:75%).
- Rata-rata terpotong (WM) mirip dengan rata-rata terpangkas. Namun demikian, dengan winsorized mean, nilai yang berada di luar batas tidak diabaikan, tetapi dianggap sama dengan nilai di edge batas yang sesuai. Setelah normalisasi ini, rata-rata dihitung. Anda menetapkan batasnya dengan satu atau dua angka antara 0 dan 100, hingga 10 tempat desimal. Sebagai contoh, wm98 menghitung rata-rata sambil memperlakukan 2% dari nilai tertinggi agar sama dengan nilai pada persentil ke 98. WM(10%:90%) menghitung rata-rata sambil memperlakukan 10% titik data tertinggi sebagai nilai batas 90%, dan memperlakukan 10% titik data terendah sebagai nilai batas 10%.
- Peringkat persentil (PR) adalah persentase nilai yang memenuhi ambang batas tetap. Sebagai contoh, PR(:300) mengembalikan persentase titik data yang memiliki nilai 300 atau kurang. PR(100:2000) mengembalikan persentase titik data yang memiliki nilai antara 100 dan 2000.

  Peringkat persentil eksklusif pada batas bawah dan inklusif di batas atas.
- Hitungan terpangkas (TC) adalah jumlah titik data yang ada dalam rentang yang dipilih untuk statistik rata-rata terpangkas. Sebagai contoh, tc90 mengembalikan jumlah titik data yang tidak termasuk titik data apa pun yang termasuk dalam 10% nilai tertinggi. TC(0.005:0.030) mengembalikan jumlah titik data dengan nilai antara 0,005 (eksklusif) dan 0,030 (inklusif).
- Jumlah terpangkas (TS) adalah jumlah nilai-nilai titik data yang ada dalam rentang yang dipilih untuk statistik rata-rata terpangkas. Ini setara dengan (Rata-rata Terpangkas) * (Hitungan Terpangkas). Sebagai contoh, ts90 mengembalikan jumlah titik data yang tidak termasuk titik data apa pun yang termasuk dalam 10% nilai tertinggi. TS(80%:) mengembalikan jumlah nilai titik data, tidak termasuk titik data apa pun dengan nilai di 80% terendah dari rentang nilai.

> **Note**
>
> Untuk Rata-rata Terpangkas, Hitungan Terpangkas, Jumlah Terpangkas, dan Rata-rata Terpotong, jika Anda mendefinisikan dua batas sebagai nilai tetap alih-alih persentase, penghitungan mencakup nilai yang sama dengan batas yang lebih tinggi, tetapi tidak termasuk nilai yang sama dengan batas bawah.

Sintaks

Untuk Rata-rata Terpangkas, Hitungan Terpangkas, Jumlah Terpangkas, dan Rata-rata Terpotong, aturan sintaks berikut berlaku:

- Menggunakan tanda kurung dengan satu atau dua angka dengan tanda persen mendefinisikan batas yang akan digunakan sebagai nilai dalam set data yang berada di antara dua persentil yang Anda tentukan. Sebagai contoh, TM(10%:90%) hanya menggunakan nilai antara persentil ke 10 dan ke 90. TM(:95%) menggunakan nilai dari ujung terendah set data yang diatur hingga persentil ke 95, mengabaikan 5% titik data dengan nilai tertinggi.
- Menggunakan tanda kurung dengan satu atau dua angka tanpa tanda persen mendefinisikan batas yang akan digunakan sebagai nilai dalam set data yang berada di antara nilai eksplisit yang Anda tentukan. Sebagai contoh, TC(80:500) hanya menggunakan nilai antara 80 (eksklusif) dan 500 (inklusif). TC(:0.5) hanya menggunakan nilai yang sama dengan 0,5 atau lebih rendah.
- Menggunakan satu angka tanpa tanda kurung menghitung menggunakan persentase, mengabaikan titik data yang lebih tinggi dari persentil yang ditentukan. Sebagai contoh, tm99 menghitung mean sambil mengabaikan 1% titik data dengan nilai tertinggi. Ini sama dengan TM(:99%).
- Rata-rata Terpangkas, Hitungan Terpangkas, Jumlah Terpangkas, dan Rata-rata Terpotong semuanya dapat disingkat menggunakan huruf besar saat menentukan rentang, seperti TM(5%:95%), TM(100:200), atau TM(:95%). Anda hanya dapat menyingkat menggunakan huruf kecil ketika menentukan satu angka saja, seperti tm99.

Kasus penggunaan statistik

- Rata-rata terpangkas paling berguna untuk metrik dengan ukuran sampel yang besar, seperti latensi halaman web. Sebagai contoh, tm99 mengabaikan pencilan yang sangat ekstrem yang mungkin disebabkan masalah jaringan atau kesalahan manusia, yang akan memberikan angka yang lebih akurat untuk latensi rata-rata permintaan khusus. Demikian pula, TM(10%:) mengabaikan 10% nilai latensi terendah, seperti yang dihasilkan dari klik cache. Dan TM(10%:99%) tidak termasuk kedua jenis pencilan ini. Kami menyarankan agar Anda menggunakan rata-rata terpangkas untuk memantau latensi.
- Sebaiknya perhatikan hitungan terpangkas setiap kali Anda menggunakan rata-rata terpangkas, untuk memastikan bahwa jumlah nilai yang digunakan dalam penghitungan rata-rata terpangkas cukup untuk secara statistik menjadi signifikan.

- Peringkat persentil memungkinkan Anda memasukkan nilai ke dalam "tempat sampah" rentang, dan Anda dapat menggunakannya untuk membuat histogram secara manual. Untuk melakukan ini, pisahkan nilai Anda menjadi berbagai tempat sampah, seperti PR(:1), PR(1:5), PR(5:10), dan PR(10:). Masukkan masing-masing tempat sampah ini ke dalam visualisasi sebagai diagram batang, dan Anda memiliki histogram.

  Peringkat persentil eksklusif pada batas bawah dan inklusif di batas atas.

Persentil versus rata-rata terpangkas

Persentil seperti p99 dan rata-rata terpangkas seperti tm99 mengukur nilai yang serupa, tetapi tidak identik. Baik p99 dan tm99 mengabaikan 1% titik data dengan nilai tertinggi, yang dianggap pencilan. Setelah itu, p99 adalah nilai maksimum dari 99% sisanya, sedangkan tm99 adalah rata-rata dari 99% sisanya. Jika Anda melihat latensi permintaan web, p99 memberi tahu Anda pengalaman pelanggan terburuk, mengabaikan pencilan, sementara tm99 memberi tahu Anda pengalaman pelanggan rata-rata, mengabaikan pencilan.

Rata-rata terpangkas adalah statistik latensi yang baik untuk dilihat jika Anda ingin mengoptimalkan pengalaman pelanggan Anda.

Persyaratan menggunakan persentil, rata-rata terpangkas, dan beberapa statistik lainnya

CloudWatch membutuhkan poin data mentah untuk menghitung statistik berikut:

- Persentil
- Rata-rata terpangkas
- Interquartile mean
- Winsorized mean
- Trimmed sum
- Trimmed count
- Peringkat persentil

Jika menerbitkan data untuk statistik kustom menggunakan himpunan statistik, alih-alih data mentah, Anda dapat mengambil jenis statistik ini untuk data ini hanya jika salah satu kondisi berikut benar:

- SampleCount Nilai himpunan statistik adalah 1 dan Min, Max, dan Jumlah semuanya sama.

- Min dan Max sama, dan Jumlah sama dengan Min dikalikan dengan. SampleCount

AWS Layanan berikut mencakup metrik yang mendukung jenis statistik ini.

- API Gateway
- Penyeimbang Beban Aplikasi
- Amazon EC2
- Penyeimbang Beban Elastis
- Kinesis
- Amazon RDS

Selain itu, jenis statistik ini tidak tersedia untuk metrik jika nilai metrik merupakan angka negatif.

Contoh berikut menunjukkan cara mendapatkan statistik untuk CloudWatch metrik sumber daya Anda, seperti instans EC2 Anda.

Contoh-contoh

- Dapatkan statistik untuk sumber daya tertentu
- Statistik agregat di seluruh sumber daya
- Gabungkan statistik berdasarkan grup Auto Scaling.
- Statistik agregat menurut Amazon Machine Image (AMI)

Dapatkan statistik untuk sumber daya tertentu

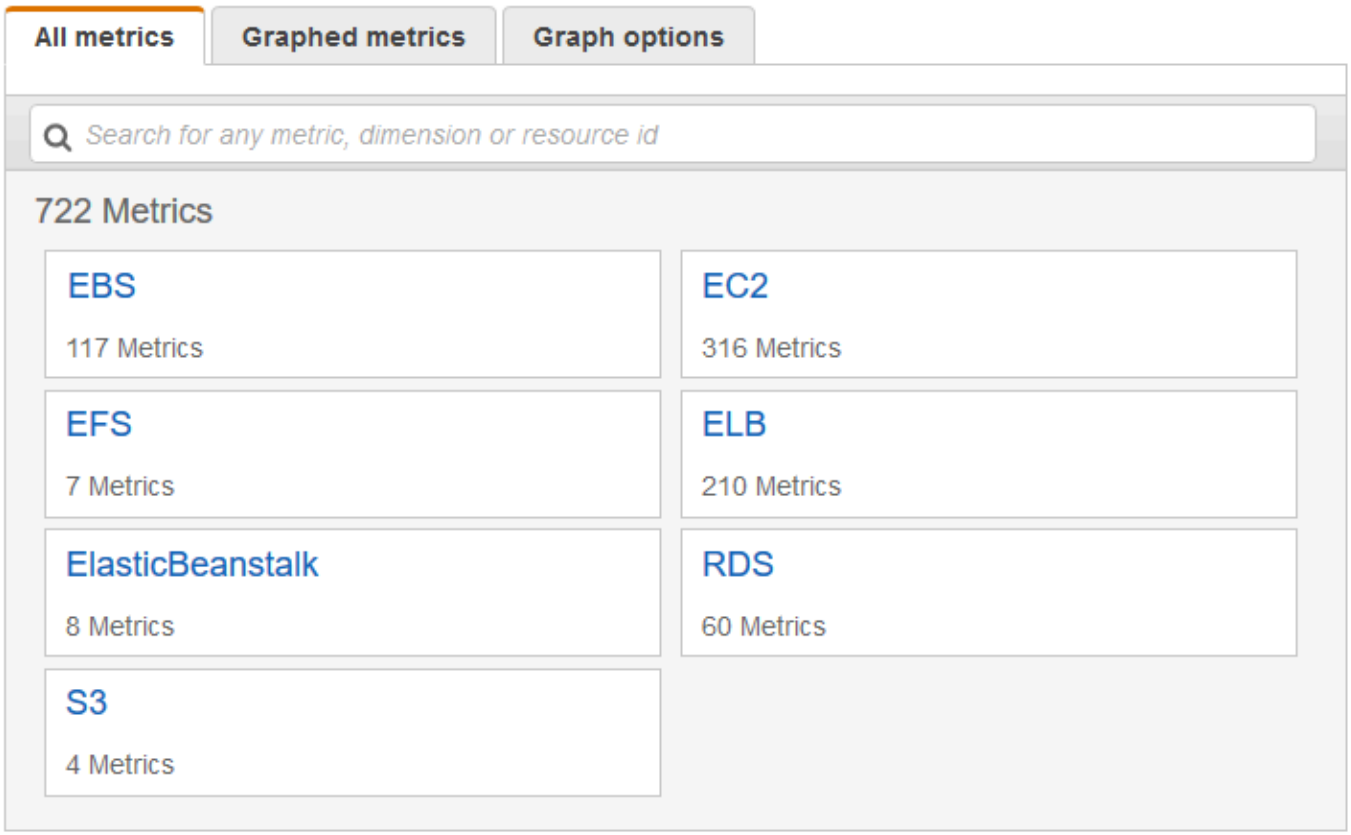
Contoh berikut menunjukkan kepada Anda cara menentukan pemanfaatan CPU maksimal dari instans EC2 tertentu.

Persyaratan

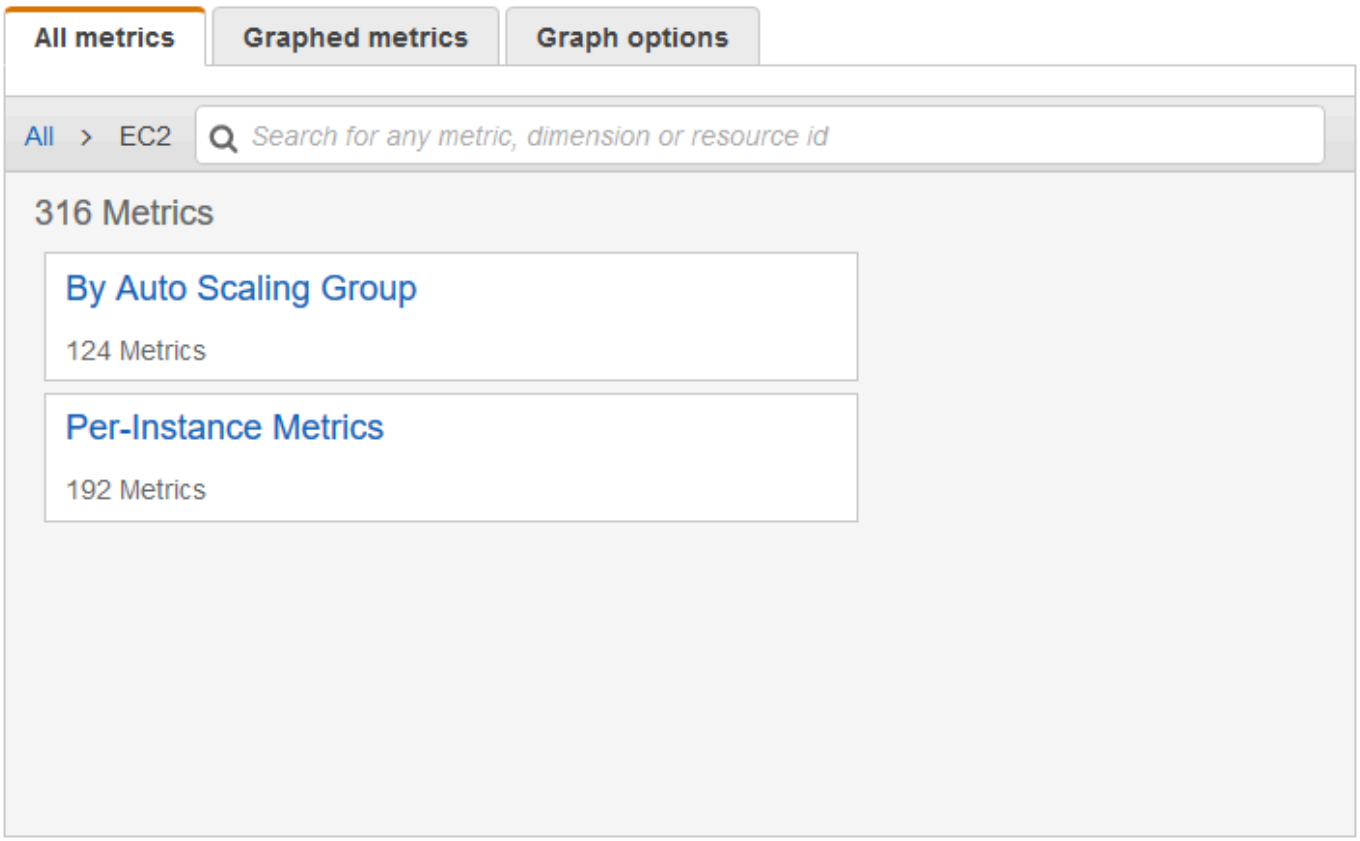
- Anda harus memiliki ID instans. Anda bisa mendapatkan ID instans menggunakan konsol Amazon EC2 atau perintah describe-instances.
- Secara bawaan, pemantauan dasar diaktifkan, tetapi Anda dapat mengaktifkan pemantauan terperinci. Untuk informasi selengkapnya, silakan lihat Aktifkan atau Nonaktifkan Pemantauan Terperinci untuk Instans Anda di Panduan Pengguna Amazon EC2 untuk Instans Linux.

Untuk menampilkan pemanfaatan CPU rata-rata pada instans tertentu menggunakan konsol

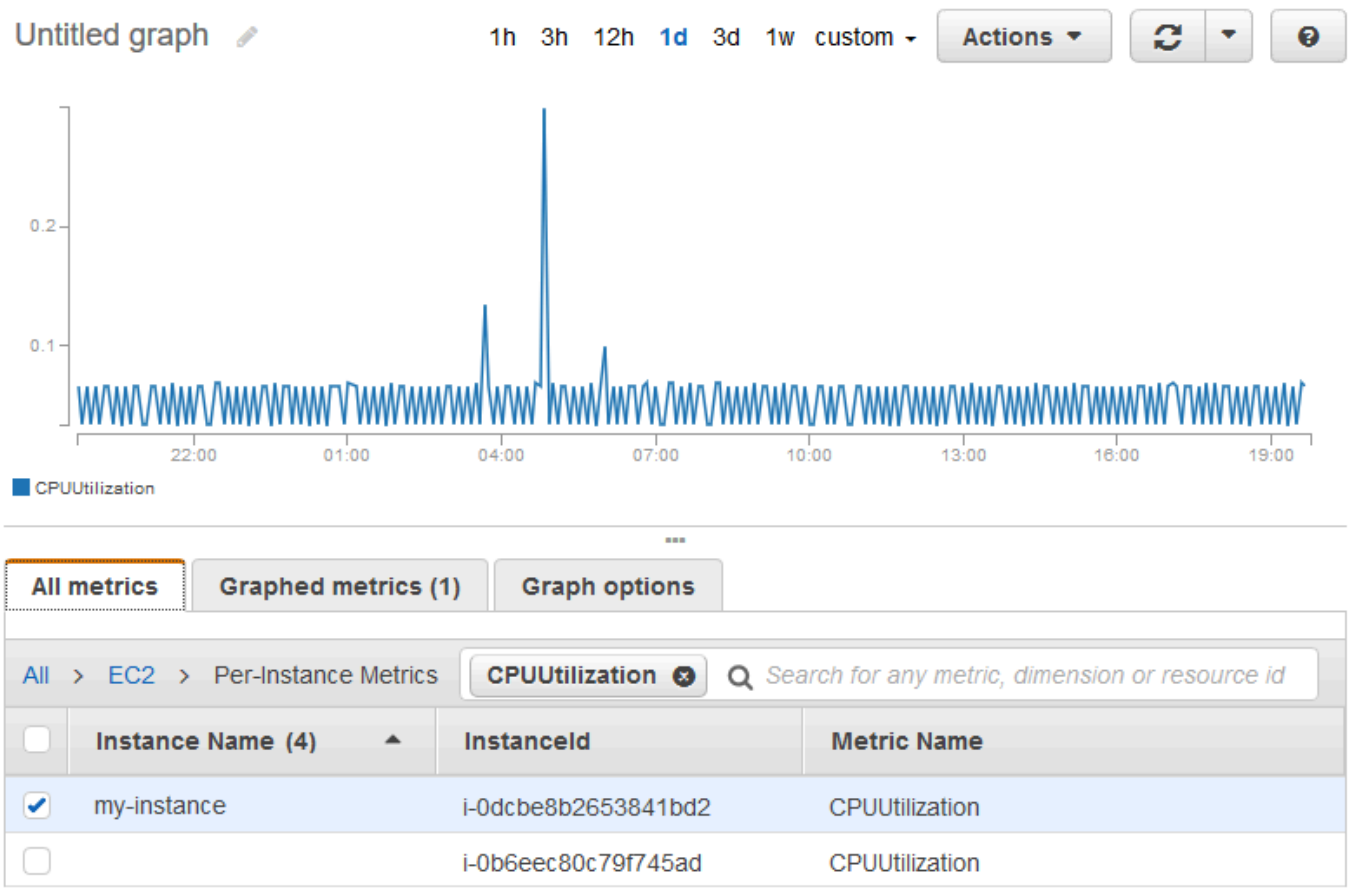
1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik.
3. Pilih namespace metrik EC2.



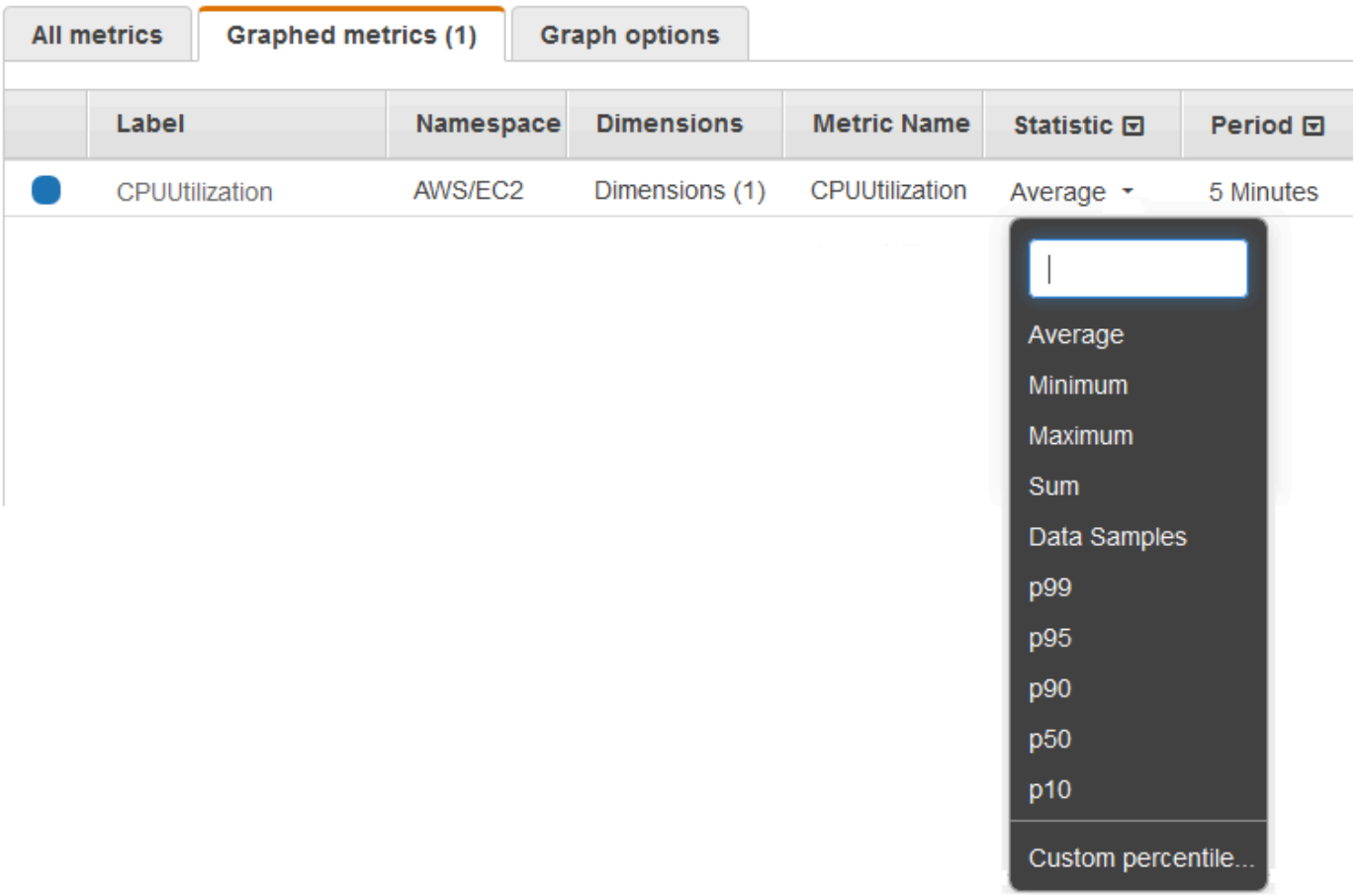
4. Pilih dimensi Metrik Per-Instans.



5. Pada bidang pencarian, masukkan **`CPUUtilization`** dan tekan Enter. Pilih baris untuk instans tertentu, yang menampilkan grafik untuk metrik `CPUUtilization` pada instans tersebut. Untuk mengubah nama grafik, pilih ikon pensil. Untuk mengubah rentang waktu, pilih salah satu nilai yang telah ditentukan sebelumnya atau pilih kustom.



6. Untuk mengubah statistik, pilih tab Metrik bergrafik. Pilih judul kolom atau nilai individu dan kemudian pilih salah satu statistik atau persentil yang telah ditentukan sebelumnya, atau tentukan persentil kustom (misalnya, **`p99.999`**).



7. Untuk mengubah periode, pilih tab Metrik bergrafik. Pilih judul kolom atau nilai individu, dan kemudian pilih nilai yang berbeda.

Untuk mendapatkan pemanfaatan CPU per instans EC2 menggunakan AWS CLI

Gunakan get-metric-statisticsperintah sebagai berikut untuk mendapatkan `CPUUtilization` metrik untuk contoh yang ditentukan.

```
aws cloudwatch get-metric-statistics --namespace AWS/EC2 --metric-name CPUUtilization \
--dimensions Name=InstanceId,Value=i-1234567890abcdef0 --statistics Maximum \
--start-time 2016-10-18T23:18:00 --end-time 2016-10-19T23:18:00 --period 360
```

Statistik yang dikembalikan adalah nilai 6 menit untuk interval waktu 24 jam yang diminta. Setiap nilai mewakili persentase pemanfaatan CPU maksimum untuk instans tertentu pada periode waktu 6 menit tertentu. Titik data tidak dikembalikan dalam urutan kronologis. Berikut ini menunjukkan awal output (output lengkap mencakup titik data untuk setiap 6 menit dari periode 24 jam) contoh.

```
{
```

```
    "Datapoints": [
        {
            "Timestamp": "2016-10-19T00:18:00Z",
            "Maximum": 0.33000000000000002,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2016-10-19T03:18:00Z",
            "Maximum": 99.670000000000002,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2016-10-19T07:18:00Z",
            "Maximum": 0.34000000000000002,
            "Unit": "Percent"
        },
        ...
    ],
    "Label": "CPUUtilization"
}
```

Statistik agregat di seluruh sumber daya

Anda dapat menggabungkan metrik untuk AWS sumber daya di beberapa sumber daya. Metrik benar-benar terpisah di antara Wilayah, tetapi Anda dapat menggunakan matematika metrik untuk menggabungkan metrik serupa di seluruh Wilayah. Untuk informasi selengkapnya, lihat Gunakan matematika metrik.

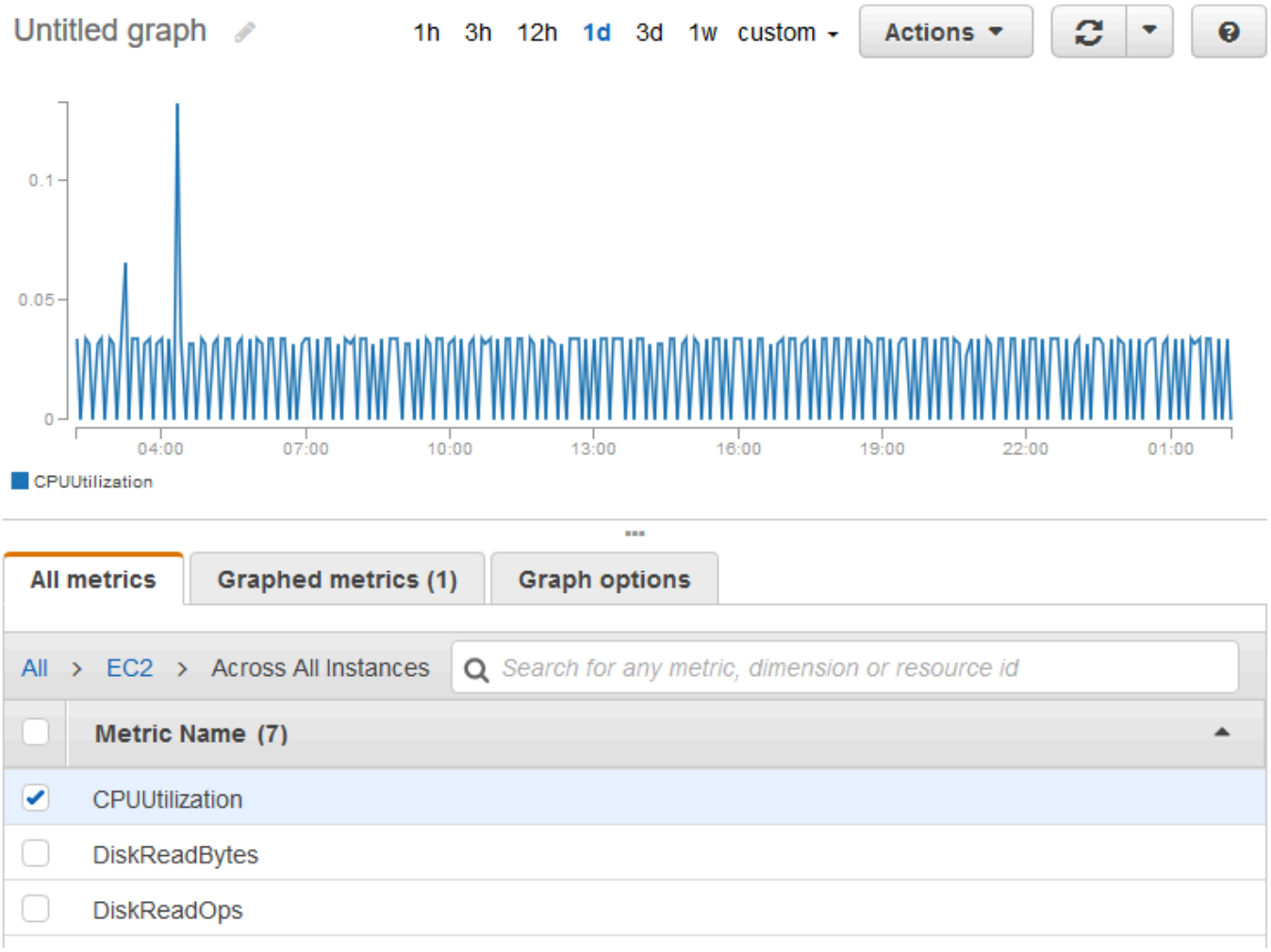
Sebagai contoh, Anda dapat menggabungkan statistik untuk instans EC2 Anda yang memiliki pemantauan terperinci yang diaktifkan. Instans yang menggunakan pemantauan dasar tidak disertakan. Oleh karena itu, Anda harus mengaktifkan pemantauan terperinci (dengan biaya tambahan), yang menyediakan data dalam periode 1 menit. Untuk informasi selengkapnya, silakan lihat Aktifkan atau Nonaktifkan Pemantauan Terperinci untuk Instans Anda di Panduan Pengguna Amazon EC2 untuk Instans Linux.

Contoh ini menunjukkan cara menggunakan pemantauan terperinci untuk mendapatkan penggunaan CPU rata-rata untuk instans EC2 Anda. Karena tidak ada dimensi yang ditentukan, CloudWatch mengembalikan statistik untuk semua dimensi di `AWS/EC2` namespace. Untuk mendapatkan statistik pada metrik lain, silakan lihat AWS layanan yang mempublikasikan CloudWatch metrik.

> **Important**
>
> Teknik untuk mengambil semua dimensi di seluruh AWS namespace ini tidak berfungsi untuk ruang nama khusus yang Anda publikasikan. CloudWatch Dengan namespace kustom, Anda harus menentukan rangkaian dimensi lengkap yang terkait dengan titik data mana pun untuk mengambil statistik yang mencakup titik data tersebut.

Untuk menampilkan pemanfaatan CPU rata-rata pada instans EC2 Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik.
3. Pilih namespace EC2, dan pilih Di Semua Instans.
4. Pilih baris yang berisi `CPUUtilization`, yang menampilkan grafik untuk metrik pada semua instans EC2 Anda. Untuk mengubah nama grafik, pilih ikon pensil. Untuk mengubah rentang waktu, pilih salah satu nilai yang telah ditentukan sebelumnya atau pilih kustom.



5. Untuk mengubah statistik, pilih tab Metrik bergrafik. Pilih judul kolom atau nilai individu dan kemudian pilih salah satu statistik atau persentil yang telah ditentukan sebelumnya, atau tentukan persentil kustom (misalnya, **`p95.45`**).
6. Untuk mengubah periode, pilih tab Metrik bergrafik. Pilih judul kolom atau nilai individu, lalu pilih nilai yang berbeda.

Untuk mendapatkan pemanfaatan CPU rata-rata di seluruh instans EC2 Anda menggunakan AWS CLI

Gunakan get-metric-statisticsperintah sebagai berikut:

```
aws cloudwatch get-metric-statistics --namespace AWS/EC2 --metric-name CPUUtilization
 --statistics "Average" "SampleCount" \
--start-time 2016-10-11T23:18:00 --end-time 2016-10-12T23:18:00 --period 3600
```

Berikut ini output contohnya:

```
{
    "Datapoints": [
        {
            "SampleCount": 238.0,
            "Timestamp": "2016-10-12T07:18:00Z",
            "Average": 0.038235294117647062,
            "Unit": "Percent"
        },
        {
            "SampleCount": 240.0,
            "Timestamp": "2016-10-12T09:18:00Z",
            "Average": 0.16670833333333332,
            "Unit": "Percent"
        },
        {
            "SampleCount": 238.0,
            "Timestamp": "2016-10-11T23:18:00Z",
            "Average": 0.041596638655462197,
            "Unit": "Percent"
        },
        ...
    ],
    "Label": "CPUUtilization"
}
```

Gabungkan statistik berdasarkan grup Auto Scaling.

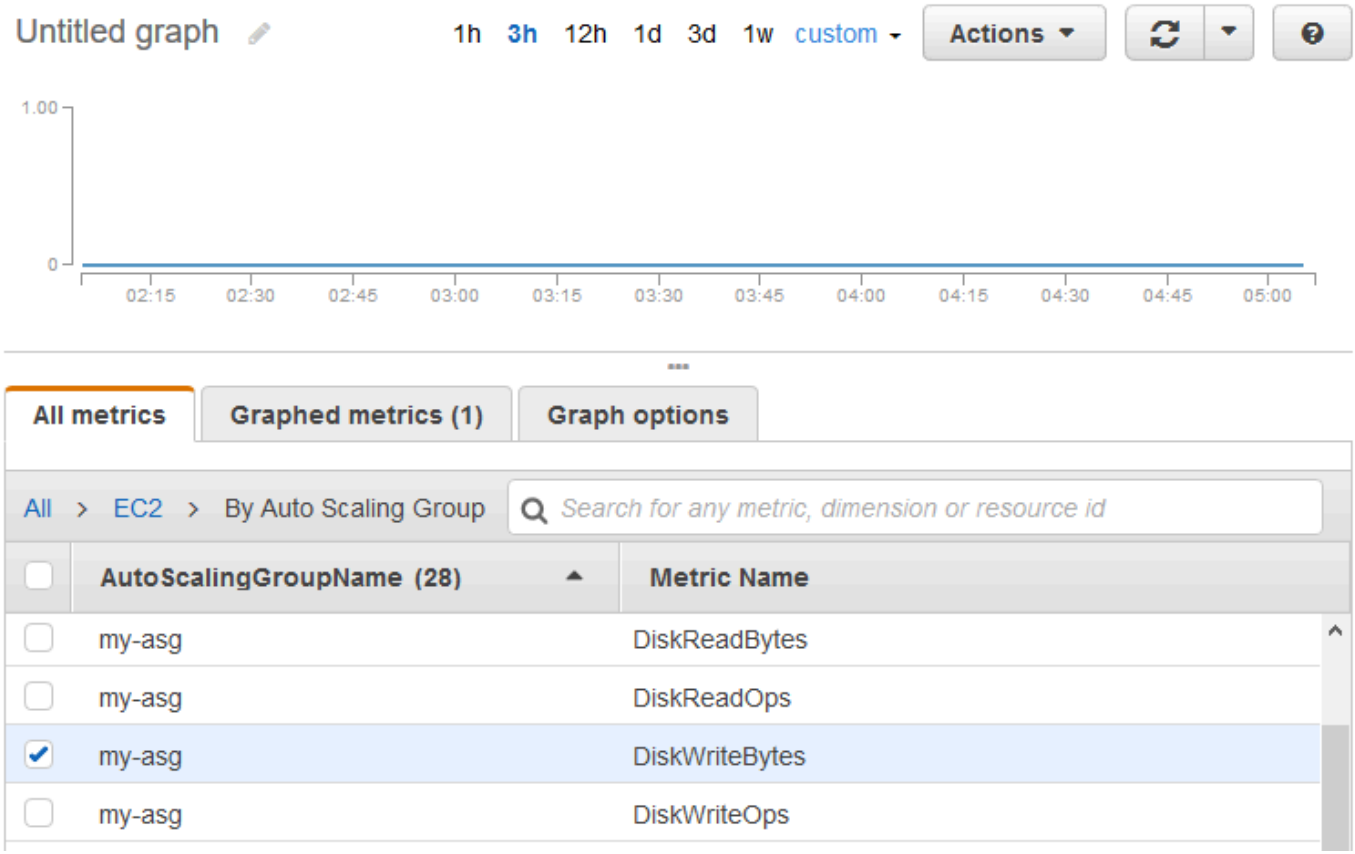
Anda dapat menggabungkan statistik untuk instans EC2 dalam sebuah grup grup Auto Scaling. Metrik benar-benar terpisah antar Wilayah, tetapi Anda dapat menggunakan matematika CloudWatch metrik untuk menggabungkan dan mengubah metrik dari beberapa Wilayah. Anda juga dapat menggunakan dasbor lintas akun untuk melakukan penghitungan metrik pada metrik dari akun yang berbeda.

Contoh ini menunjukkan cara mengambil total byte yang ditulis ke disk untuk satu grup Auto Scaling. Total dihitung selama periode 1 menit untuk interval 24 jam di semua contoh EC2 dalam grup Auto Scaling tertentu.

DiskWriteBytes Untuk menampilkan instance dalam grup Auto Scaling menggunakan konsol

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.

2. Pada panel navigasi, silakan pilih Metrik.
3. Pilih namespace EC2 dan kemudian pilih Berdasarkan grup Auto Scaling.
4. Pilih baris untuk DiskWriteBytesmetrik dan grup Auto Scaling tertentu, yang menampilkan grafik untuk metrik untuk instance dalam grup Auto Scaling. Untuk mengubah nama grafik, pilih ikon pensil. Untuk mengubah rentang waktu, pilih salah satu nilai yang telah ditentukan sebelumnya atau pilih kustom.



5. Untuk mengubah statistik, pilih tab Metrik bergrafik. Pilih judul kolom atau nilai individu dan kemudian pilih salah satu statistik atau persentil yang telah ditentukan sebelumnya, atau tentukan persentil kustom (misalnya, **`p95.45`**).
6. Untuk mengubah periode, pilih tab Metrik bergrafik. Pilih judul kolom atau nilai individu, lalu pilih nilai yang berbeda.

DiskWriteBytes Untuk mendapatkan instance dalam grup Auto Scaling menggunakan AWS CLI

Gunakan perintah get-metric-statistics sebagai berikut.

```
aws cloudwatch get-metric-statistics --namespace AWS/EC2 --metric-name DiskWriteBytes
--dimensions Name=AutoScalingGroupName,Value=my-asg --statistics "Sum" "SampleCount" \
```

```
--start-time 2016-10-16T23:18:00 --end-time 2016-10-18T23:18:00 --period 360
```

Berikut ini adalah output contoh.

```
{
    "Datapoints": [
        {
            "SampleCount": 18.0,
            "Timestamp": "2016-10-19T21:36:00Z",
            "Sum": 0.0,
            "Unit": "Bytes"
        },
        {
            "SampleCount": 5.0,
            "Timestamp": "2016-10-19T21:42:00Z",
            "Sum": 0.0,
            "Unit": "Bytes"
        }
    ],
    "Label": "DiskWriteBytes"
}
```

Statistik agregat menurut Amazon Machine Image (AMI)

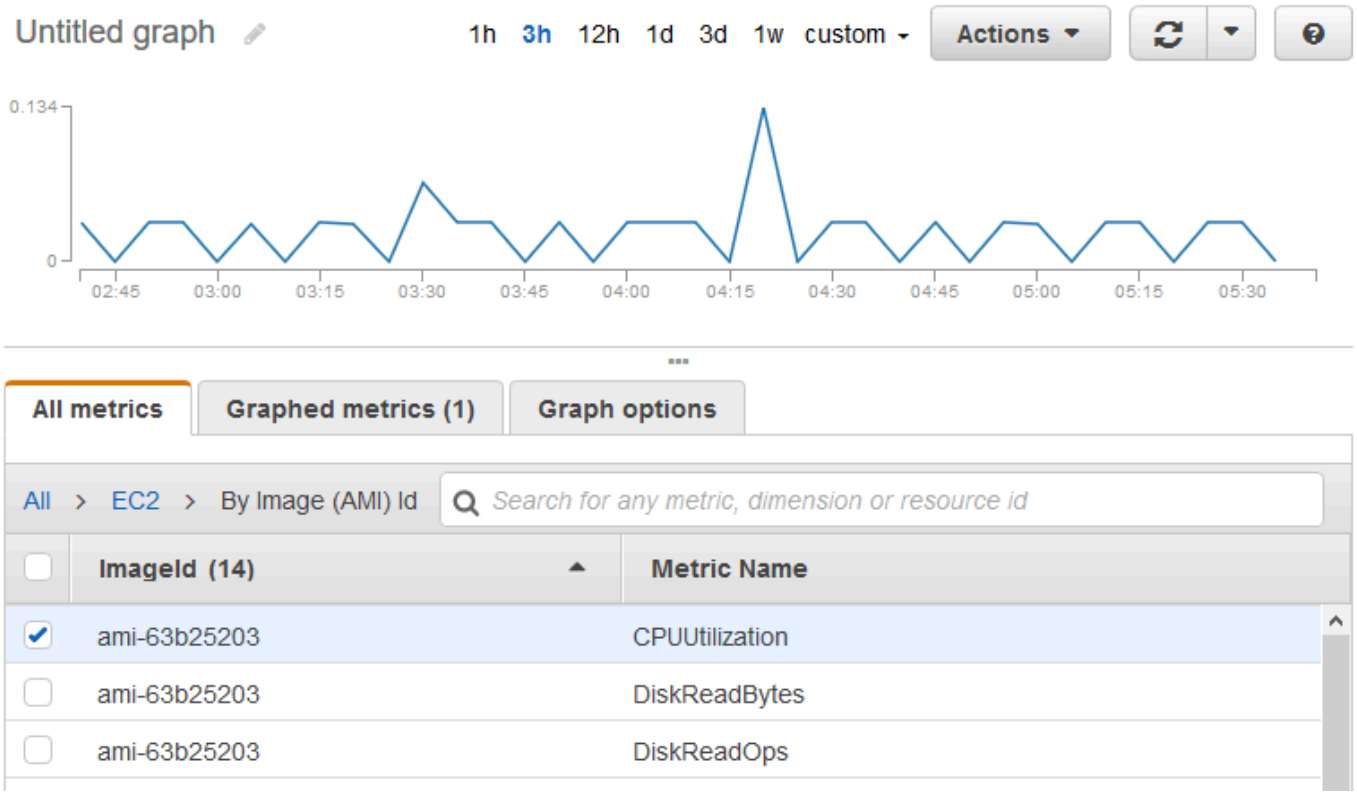
Anda dapat menggabungkan statistik untuk instans EC2 yang telah mengaktifkan pemantauan terperinci. Instans yang menggunakan pemantauan dasar tidak disertakan. Untuk informasi selengkapnya, silakan lihat Aktifkan atau Nonaktifkan Pemantauan Terperinci untuk Instans Anda di Panduan Pengguna Amazon EC2 untuk Instans Linux.

Contoh ini menunjukkan kepada Anda cara menentukan rata-rata pemanfaatan CPU untuk semua instans yang menggunakan AMI tertentu. Rata-rata adalah interval waktu lebih dari 60 detik untuk periode satu hari.

Untuk menampilkan pemanfaatan CPU rata-rata oleh AMI menggunakan konsol

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik.
3. Pilih namespace EC2 dan kemudian pilih Berdasarkan Id Image (AMI).

4. Pilih baris untuk metrik `CPUUtilization` dan AMI tertentu, yang menampilkan grafik metrik untuk AMI tertentu. Untuk mengubah nama grafik, pilih ikon pensil. Untuk mengubah rentang waktu, pilih salah satu nilai yang telah ditentukan sebelumnya atau pilih kustom.



5. Untuk mengubah statistik, pilih tab Metrik bergrafik. Pilih judul kolom atau nilai individu dan kemudian pilih salah satu statistik atau persentil yang telah ditentukan sebelumnya, atau tentukan persentil kustom (misalnya, **`p95.45`**).
6. Untuk mengubah periode, pilih tab Metrik bergrafik. Pilih judul kolom atau nilai individu, lalu pilih nilai yang berbeda.

Untuk mendapatkan pemanfaatan CPU rata-rata oleh AMI menggunakan AWS CLI

Gunakan perintah get-metric-statistics sebagai berikut.

```
aws cloudwatch get-metric-statistics --namespace AWS/EC2 --metric-name CPUUtilization \
--dimensions Name=ImageId,Value=ami-3c47a355 --statistics Average \
--start-time 2016-10-10T00:00:00 --end-time 2016-10-11T00:00:00 --period 3600
```

Operasi mengembalikan statistik yang merupakan nilai satu jam untuk interval satu hari. Setiap nilai mewakili rata-rata persentase pemanfaatan CPU untuk instans EC2 yang menjalankan AMI tertentu. Berikut ini adalah output contoh.

```
{
    "Datapoints": [
        {
            "Timestamp": "2016-10-10T07:00:00Z",
            "Average": 0.041000000000000009,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2016-10-10T14:00:00Z",
            "Average": 0.079579831932773085,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2016-10-10T06:00:00Z",
            "Average": 0.036000000000000011,
            "Unit": "Percent"
        },
        ...
    ],
    "Label": "CPUUtilization"
}
```

Menerbitkan metrik kustom

Anda dapat mempublikasikan metrik Anda sendiri untuk CloudWatch menggunakan AWS CLI atau API. Anda dapat melihat grafik statistik metrik yang dipublikasikan dengan. AWS Management Console

CloudWatch menyimpan data tentang metrik sebagai serangkaian titik data. Setiap titik data memiliki stempel waktu terkait. Bahkan, Anda dapat menerbitkan himpunan titik data gabungan yang disebut himpunan statistik.

Topik

- Berhenti menerbitkan metrik

Metrik resolusi tinggi

Setiap metrik adalah salah satu dari berikut:

- Resolusi standar, dengan data yang memiliki granularitas satu menit
- Resolusi tinggi, dengan data pada granularitas satu detik

Metrik yang dihasilkan oleh AWS layanan adalah resolusi standar secara default. Ketika menerbitkan sebuah metrik kustom, Anda dapat menetapkannya sebagai resolusi standar atau resolusi tinggi. Saat Anda menerbitkan metrik resolusi tinggi, CloudWatch simpan dengan resolusi 1 detik, dan Anda dapat membaca dan mengambilnya dengan jangka waktu 1 detik, 5 detik, 10 detik, 30 detik, atau kelipatan 60 detik.

Metrik resolusi tinggi dapat memberi wawasan yang lebih cepat tentang aktivitas submenit aplikasi Anda. Ingatlah bahwa setiap panggilan `PutMetricData` untuk sebuah metrik kustom yang dikenakan biaya, sehingga memanggil `PutMetricData` lebih sering pada sebuah metrik resolusi tinggi akan dapat mengakibatkan biaya yang lebih tinggi. Untuk informasi selengkapnya tentang CloudWatch harga, lihat CloudWatch Harga Amazon.

Jika Anda mengatur sebuah alarm pada metrik resolusi tinggi, maka Anda dapat menentukan alarm resolusi tinggi dengan periode 10 detik atau 30 detik, atau Anda dapat mengatur sebuah alarm biasa dengan periode lebih dari 60 detik. Terdapat biaya yang lebih tinggi untuk alarm resolusi tinggi dengan periode 10 atau 30 detik.

Gunakan dimensi

Dalam metrik kustom, parameter `--dimensions` merupakan hal umum. Dimensi lebih lanjut menjelaskan apa itu metrik dan data apa yang disimpannya. Anda dapat memiliki hingga 30 dimensi yang ditetapkan pada satu metrik, dan setiap dimensi ditentukan oleh nama dan pasangan nilai.

Cara menentukan dimensi yang berbeda ketika Anda menggunakan perintah yang berbeda. Dengan put-metric-data, Anda menentukan setiap dimensi sebagai *MyName*=*MyValue*, dan dengan get-metric-statisticsatau put-metric-alarmAnda menggunakan format `Name=` *MyName*, `Value=` *MyValue*. Sebagai contoh, perintah berikut menerbitkan metrik `Buffers` dengan dua dimensi yang disebut `InstanceId` dan `InstanceType`.

```
aws cloudwatch put-metric-data --metric-name Buffers --namespace MyNameSpace --unit
 Bytes --value 231434333 --dimensions InstanceId=1-23456789,InstanceType=m1.small
```

Perintah ini mengambil statistik untuk metrik yang sama. Pisahkan bagian Nama dan Nilai dari dimensi tunggal dengan tanda koma, tetapi jika Anda memiliki beberapa dimensi, gunakan ruang antara satu dimensi dan dimensi berikutnya.

```
aws cloudwatch get-metric-statistics --metric-name Buffers --namespace MyNameSpace --
dimensions Name=InstanceId,Value=1-23456789 Name=InstanceType,Value=m1.small --start-
time 2016-10-15T04:00:00Z --end-time 2016-10-19T07:00:00Z --statistics Average --period
 60
```

Jika satu metrik menyertakan beberapa dimensi, Anda harus menentukan nilai untuk setiap dimensi yang ditentukan saat Anda menggunakannya get-metric-statistics. Misalnya, metrik Amazon S3 `BucketSizeBytes` menyertakan dimensi `BucketName` dan`StorageType`, jadi Anda harus menentukan kedua dimensi dengan. get-metric-statistics

```
aws cloudwatch get-metric-statistics --metric-name BucketSizeBytes --start-time
 2017-01-23T14:23:00Z --end-time 2017-01-26T19:30:00Z --period 3600 --namespace
 AWS/S3 --statistics Maximum --dimensions Name=BucketName,Value=MyBucketName
 Name=StorageType,Value=StandardStorage --output table
```

Untuk melihat dimensi apa yang ditentukan untuk metrik, gunakan perintah list-metrics.

Menerbitkan titik data tunggal

Untuk mempublikasikan satu titik data untuk metrik baru atau yang sudah ada, gunakan put-metric-dataperintah dengan satu nilai dan cap waktu. Misalnya, setiap tindakan berikut menerbitkan satu titik data.

```
aws cloudwatch put-metric-data --metric-name PageViewCount --namespace MyService --
value 2 --timestamp 2016-10-20T12:00:00.000Z
aws cloudwatch put-metric-data --metric-name PageViewCount --namespace MyService --
value 4 --timestamp 2016-10-20T12:00:01.000Z
aws cloudwatch put-metric-data --metric-name PageViewCount --namespace MyService --
value 5 --timestamp 2016-10-20T12:00:02.000Z
```

Jika Anda memanggil perintah ini dengan nama metrik baru, CloudWatch buat metrik untuk Anda. Jika tidak, CloudWatch kaitkan data Anda dengan metrik yang ada yang Anda tentukan.

> **Note**
>
> Saat Anda membuat metrik, diperlukan waktu hingga 2 menit sebelum Anda dapat mengambil statistik untuk metrik baru menggunakan get-metric-statisticsperintah. Namun demikian, ini dapat memakan waktu hingga 15 menit sebelum metrik baru tersebut muncul dalam daftar metrik yang diambil menggunakan perintah list-metrics.

Meskipun Anda dapat mempublikasikan titik data dengan stempel waktu sebagai granular seperseribu detik, CloudWatch agregat data ke granularitas minimum 1 detik. CloudWatch mencatat rata-rata (jumlah semua item dibagi dengan jumlah item) dari nilai yang diterima untuk setiap periode, serta jumlah sampel, nilai maksimum, dan nilai minimum untuk periode waktu yang sama. Sebagai contoh, metrik `PageViewCount` dari contoh sebelumnya memuat tiga titik data dengan stempel waktu yang hanya berjarak beberapa detik. Jika periode Anda disetel ke 1 menit, CloudWatch agregat tiga titik data karena semuanya memiliki stempel waktu dalam periode 1 menit.

Anda dapat menggunakan perintah get-metric-statistics untuk mengambil statistik berdasarkan titik data yang dipublikasikan.

```
aws cloudwatch get-metric-statistics --namespace MyService --metric-name PageViewCount
 \
--statistics "Sum" "Maximum" "Minimum" "Average" "SampleCount" \
--start-time 2016-10-20T12:00:00.000Z --end-time 2016-10-20T12:05:00.000Z --period 60
```

Berikut ini adalah output contoh.

```
{
    "Datapoints": [
        {
            "SampleCount": 3.0,
            "Timestamp": "2016-10-20T12:00:00Z",
            "Average": 3.6666666666666665,
            "Maximum": 5.0,
            "Minimum": 2.0,
            "Sum": 11.0,
            "Unit": "None"
        }
    ],
    "Label": "PageViewCount"
}
```

Menerbitkan himpunan statistik

Anda dapat mengumpulkan data Anda sebelum mempublikasikannya CloudWatch. Bila Anda memiliki beberapa titik data per menit, menggabungkan data meminimalkan jumlah panggilan menjadi put-metric-data. Misalnya, alih-alih memanggil put-metric-data beberapa kali untuk tiga titik data yang berada dalam 3 detik satu sama lain, Anda dapat menggabungkan data ke dalam satu himpunan statistik yang Anda publikasikan dengan satu panggilan, menggunakan parameter `--statistic-values`.

```
aws cloudwatch put-metric-data --metric-name PageViewCount --namespace MyService
 --statistic-values Sum=11,Minimum=2,Maximum=5,SampleCount=3 --
timestamp 2016-10-14T12:00:00.000Z
```

CloudWatch membutuhkan titik data mentah untuk menghitung persentil. Jika menerbitkan data menggunakan himpunan statistik, Anda tidak dapat mengambil statistik persentil untuk data ini kecuali jika salah satu ketentuan berikut benar:

- `SampleCount` dari himpunan statistik adalah 1
- `Minimum` dan `Maximum` dari himpunan statistik adalah sama

Menerbitkan nilai Nol

Jika data Anda lebih bersifat jarang dan memiliki periode yang tidak memiliki data terkait, Anda dapat memilih untuk menerbitkan nilai nol (`0`) untuk periode tersebut atau tidak ada nilai sama sekali. Jika menggunakan panggilan berkala ke `PutMetricData` guna memantau kesehatan aplikasi Anda, Anda mungkin ingin menerbitkan nol, alih-alih tanpa nilai. Misalnya, Anda dapat mengatur CloudWatch alarm untuk memberi tahu Anda jika aplikasi Anda gagal mempublikasikan metrik setiap lima menit. Anda ingin aplikasi tersebut menerbitkan nol pada periode tanpa data terkait.

Anda juga dapat menerbitkan nol jika ingin melacak total titik data atau jika menginginkan statistik seperti minimum dan rata-rata untuk menyertakan titik data dengan nilai 0.

Berhenti menerbitkan metrik

Untuk menghentikan penerbitan metrik kustom ke CloudWatch, ubah kode aplikasi atau layanan Anda untuk berhenti menggunakan PutMetricData. CloudWatch tidak menarik metrik dari aplikasi, itu hanya menerima apa yang didorong ke sana, jadi untuk berhenti menerbitkan metrik Anda, Anda harus menghentikannya di sumbernya.

Menggunakan CloudWatch alarm Amazon

Anda dapat membuat alarm metrik dan komposit di Amazon CloudWatch.

- Alarm metrik menonton CloudWatch metrik tunggal atau hasil ekspresi matematika berdasarkan CloudWatch metrik. Alarm tersebut melakukan satu atau beberapa tindakan berdasarkan pada nilai metrik atau ekspresi relatif terhadap ambang batas selama beberapa periode waktu. Tindakan tersebut dapat berupa mengirimkan pemberitahuan ke topik Amazon SNS, melakukan tindakan Amazon EC2 atau tindakan Auto Scaling Amazon EC2, atau membuat insiden atau di Systems Manager. OpsItem
- Sebuah Alarm gabungan mencakup ekspresi aturan yang memperhitungkan status alarm-alarm lainnya yang telah Anda buat. Alarm gabungan tersebut akan beralih ke status ALARM hanya jika semua ketentuan yang ada dalam aturan itu terpenuhi. Alarm yang ditentukan dalam sebuah ekspresi aturan alarm gabungan dapat mencakup alarm-alarm metrik dan alarm-alarm gabungan lainnya.

  Menggunakan alarm gabungan dapat mengurangi bising alarm. Anda dapat membuat beberapa alarm metrik, dan juga membuat sebuah alarm gabungan serta mengatur peringatan hanya untuk alarm gabungan. Misalnya, sebuah alarm gabungan dapat beralih ke status ALARM hanya jika semua alarm metrik yang mendasarinya berada dalam status ALARM.

  Alarm komposit dapat mengirim notifikasi Amazon SNS saat mengubah status, dan dapat membuat Systems OpsItems Manager atau insiden saat masuk ke status ALARM, tetapi tidak dapat melakukan tindakan EC2 atau tindakan Auto Scaling.

> **Note**
>
> Anda dapat membuat alarm sebanyak yang Anda inginkan di AWS akun Anda.

Anda dapat menambahkan alarm ke dasbor, sehingga Anda dapat memantau dan menerima peringatan tentang AWS sumber daya dan aplikasi Anda di beberapa wilayah. Setelah Anda menambahkan sebuah alarm ke dasbor, alarm itu akan berubah menjadi abu-abu saat berada dalam status `INSUFFICIENT_DATA` dan menjadi merah saat berada dalam status `ALARM`. Alarm tersebut akan ditampilkan tanpa warna saat berada dalam status `OK`.

Anda juga dapat memfavoritkan alarm yang baru saja dikunjungi dari opsi Favorit dan terbaru di panel navigasi CloudWatch konsol. Opsi Favorit dan terbaru tersebut memiliki kolom untuk alarm-alarm favorit Anda dan alarm-alarm yang baru saja Anda kunjungi.

Sebuah alarm menginvokasi tindakan hanya ketika status alarm tersebut berubah. Kecuali alarm dengan tindakan penskalaan otomatis (Auto Scaling). Untuk tindakan penskalaan otomatis (Auto Scaling), alarm akan terus menginvokasi tindakan tersebut satu kali per menit sehingga alarm tetap berada dalam status baru.

Sebuah alarm dapat mengawasi metrik yang di akun yang sama. Jika Anda telah mengaktifkan fungsionalitas lintas akun di CloudWatch konsol, Anda juga dapat membuat alarm yang menonton metrik di akun lain. AWS Membuat alarm gabungan lintas akun tidak didukung. Membuat alarm lintas akun yang menggunakan ekspresi matematika didukung, kecuali jika fungsi `ANOMALY_DETECTION_BAND`, `INSIGHT_RULE`, dan `SERVICE_QUOTA` tidak didukung untuk alarm lintas akun.

> **Note**
>
> CloudWatch tidak menguji atau memvalidasi tindakan yang Anda tentukan, juga tidak mendeteksi kesalahan Auto Scaling Amazon EC2 atau Amazon SNS yang dihasilkan dari upaya untuk menjalankan tindakan yang tidak ada. Anda harus memastikan bahwa tindakan alarm Anda ada.

Status-status alarm metrik

Sebuah alarm metrik mungkin saja berada dalam status berikut ini:

- `OK` – Metrik atau ekspresi berada dalam ambang batas yang telah ditetapkan sebelumnya.
- `ALARM` – Metrik atau ekspresi berada di luar ambang batas yang telah ditetapkan sebelumnya.
- `INSUFFICIENT_DATA` – Alarm baru saja dimulai, metrik tidak tersedia, atau tidak ada data yang memadai yang tersedia bagi metrik untuk menentukan status alarm.

Melakukan evaluasi alarm

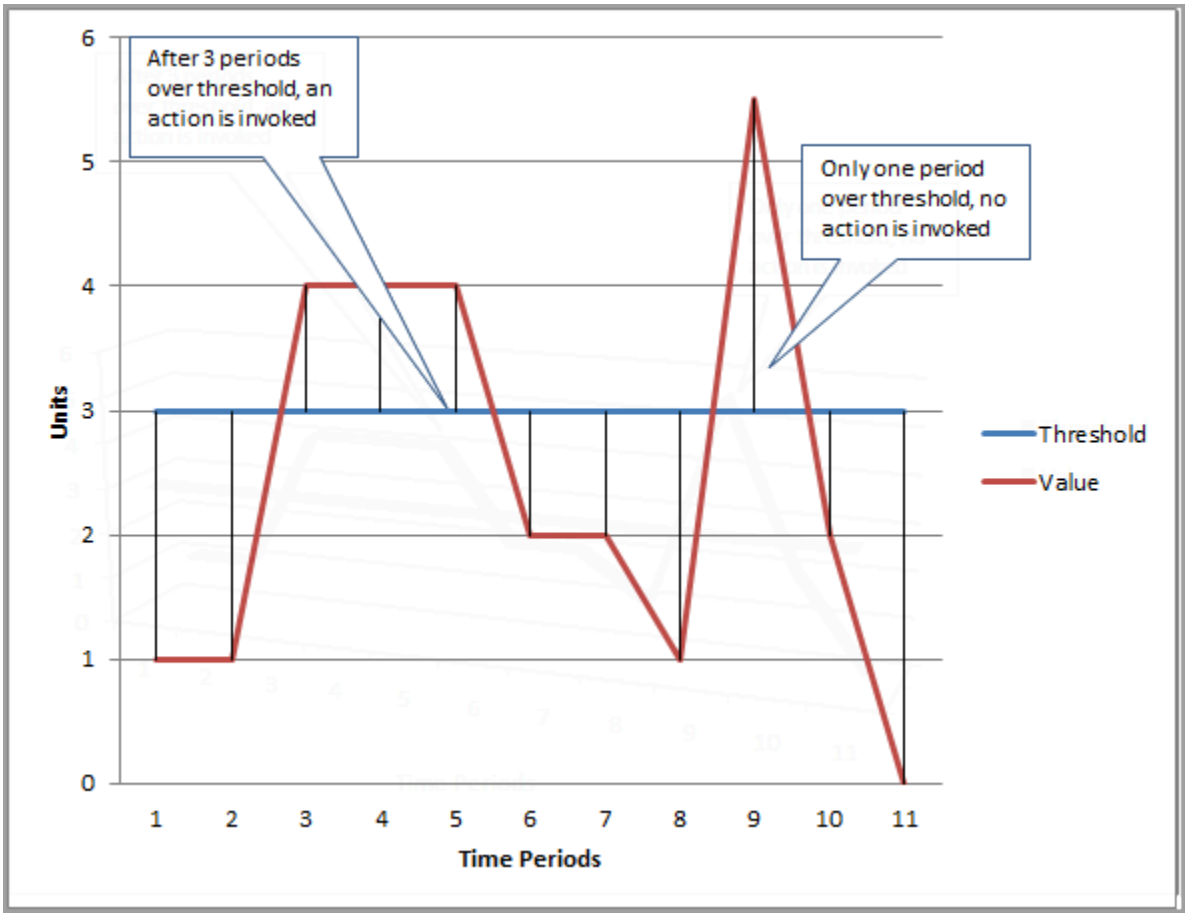
Saat Anda membuat alarm, Anda menentukan tiga pengaturan untuk mengaktifkan CloudWatch untuk mengevaluasi kapan harus mengubah status alarm:

- Periode adalah lamanya waktu yang digunakan untuk mengevaluasi metrik atau ekspresi untuk membuat setiap titik data individu untuk sebuah alarm. Periode dinyatakan dalam detik.
- Periode Evaluasi adalah jumlah periode terbaru, atau titik data, untuk melakukan evaluasi ketika menentukan status alarm.
- Titik data untuk Alarm adalah jumlah titik data dalam Periode Evaluasi yang harus dilanggar untuk menyebabkan alarm beralih ke status `ALARM`. Titik data yang melanggar tidak harus berurutan, tetapi semuanya harus berada dalam jumlah titik data terakhir yang sama dengan Periode Evaluasi.

Untuk setiap periode satu menit atau lebih, sebuah alarm dievaluasi setiap menit dan evaluasinya dilakukan berdasarkan pada jendela waktu yang ditentukan oleh Periode dan Periode Evaluasi. Sebagai contoh, jika Periode adalah 5 menit (300 detik) dan Periode Evaluasi adalah 1, maka pada akhir menit 5 alarm akan melakukan evaluasi berdasarkan data dari menit 1 hingga 5. Kemudian pada akhir menit 6, alarm tersebut akan dievaluasi berdasarkan data dari menit 2 hingga 6.

Jika periode alarm adalah 10 detik atau 30 detik, maka alarm akan dievaluasi setiap 10 detik.

Pada gambar berikut, ambang batas alarm untuk sebuah alarm metrik diatur untuk tiga unit. Baik Periode Evaluasi maupun Titik data untuk Alarm adalah 3. Artinya, ketika semua titik data yang ada dalam tiga periode terakhir yang berurutan berada di atas ambang batas, alarm akan beralih ke status `ALARM`. Dalam gambar tersebut, status ini terjadi pada periode waktu ketiga hingga kelima. Pada periode enam, nilainya turun menjadi di bawah ambang batas, sehingga salah satu periode yang dievaluasi tidak melanggar, dan status alarm kemudian berubah kembali menjadi `OK`. Selama periode kesembilan, ambang batas tersebut dilanggar lagi, tapi hanya selama satu periode. Akibatnya, alarm tetap berada dalam status `OK`.



Ketika Anda mengonfigurasi Periode Evaluasi dan Titik data untuk Alarm dengan nilai yang berbeda, Anda sedang mengatur alarm "M dari N". Titik data untuk Alarm adalah ("M") dan Periode Evaluasi adalah ("N"). Interval evaluasi adalah jumlah periode evaluasi yang dikalikan dengan panjang periode. Sebagai contoh, jika Anda mengonfigurasikan 4 dari 5 titik data dengan periode 1 menit, maka interval evaluasinya adalah 5 menit. Jika Anda mengonfigurasikan 3 dari 3 titik data dengan periode 10 menit, maka interval evaluasinya adalah 30 menit.

> **Note**
>
> Jika titik data hilang segera setelah Anda membuat alarm, dan metrik dilaporkan CloudWatch sebelum Anda membuat alarm, CloudWatch ambil titik data terbaru dari sebelum alarm dibuat saat mengevaluasi alarm.

Tindakan-tindakan alarm

Anda dapat menentukan tindakan-tindakan apa saja yang dilakukan alarm saat mengubah status antara status OK, ALARM, dan INSUFFICIENT_DATA.

Sebagian besar tindakan dapat diatur untuk beralih ke masing-masing tiga status. Kecuali untuk tindakan penskalaan otomatis (Auto Scaling), maka tindakan tersebut hanya terjadi pada peralihan status, dan tidak akan dilakukan lagi jika kondisinya berlanjut selama berjam-jam atau berhari-hari. Anda dapat menggunakan fakta bahwa beberapa tindakan diizinkan untuk sebuah alarm untuk mengirim email ketika ambang batas dilanggar, dan kemudian tindakan lain ketika kondisi pelanggaran berakhir. Hal ini akan membantu Anda memverifikasi bahwa tindakan-tindakan penskalaan atau pemulihan Anda dipicu saat diharapkan dan berfungsi sesuai dengan keinginan Anda.

Berikut ini didukung sebagai tindakan-tindakan alarm.

- Kirimkan notifikasi kepada satu atau beberapa pelanggan dengan menggunakan topik Amazon Simple Notification Service. Pelanggan dapat berupa aplikasi maupun orang perorangan. Untuk informasi selengkapnya mengenai Amazon SNS, silakan lihat Apa yang Dimaksud dengan Amazon SNS?.
- Menginvokasi sebuah fungsi Lambda. Ini adalah cara paling mudah yang bisa dilakukan untuk melakukan otomatisasi tindakan-tindakan kustom pada perubahan status alarm.
- Alarm-alarm berdasarkan metrik EC2 juga dapat melakukan tindakan-tindakan EC2, seperti menghentikan, mengakhiri, me-reboot, atau memulihkan instans EC2. Untuk informasi selengkapnya, lihat Membuat alarm untuk menghentikan, mengakhiri, menyalakan ulang, atau memulihkan instans EC2.
- Alarm-alarm juga dapat melakukan tindakan untuk menskalakan grup Auto Scaling. Untuk informasi selengkapnya, silakan lihat Langkah dan kebijakan penskalaan sederhana untuk Amazon EC2 Auto Scaling.
- Alarm dapat dibuat OpsItems di Systems Manager Ops Center atau membuat insiden di AWS Systems Manager Incident Manager. Tindakan-tindakan ini dilakukan hanya saat alarm beralih statusnya menjadi ALARM. Untuk informasi selengkapnya, lihat Mengkonfigurasi CloudWatch untuk membuat OpsItems dari alarm dan pembuatan Insiden.

Tindakan-tindakan alarm Lambda

CloudWatch alarm menjamin pemanggilan fungsi Lambda asinkron untuk perubahan status tertentu, kecuali dalam kasus berikut:

- Ketika fungsi tidak ada.
- Kapan CloudWatch tidak diizinkan untuk menjalankan fungsi Lambda.

Jika tidak CloudWatch dapat mencapai layanan Lambda atau pesan ditolak karena alasan lain, CloudWatch coba lagi sampai pemanggilan berhasil. Lambda mengantri pesan dan menangani percobaan ulang eksekusi. Untuk informasi selengkapnya tentang model eksekusi ini, termasuk informasi tentang cara Lambda menangani kesalahan, lihat Pemanggilan asinkron di Panduan Pengembang. AWS Lambda

Anda dapat menjalankan fungsi Lambda di akun yang sama, atau di AWS akun lain.

Saat Anda menentukan sebuah alarm untuk menginvokasi sebuah fungsi Lambda sebagai tindakan alarm, Anda dapat memilih untuk menentukan nama fungsi, nama alias fungsi, atau versi tertentu dari sebuah fungsi.

Saat Anda menentukan fungsi Lambda sebagai tindakan alarm, Anda harus membuat kebijakan sumber daya agar fungsi tersebut memungkinkan prinsipal CloudWatch layanan menjalankan fungsi tersebut.

Salah satu cara untuk melakukannya adalah dengan menggunakan AWS CLI, seperti pada contoh berikut:

```
aws lambda add-permission \
--function-name my-function-name \
--statement-id AlarmAction \
--action 'lambda:InvokeFunction' \
--principal lambda.alarms.cloudwatch.amazonaws.com \
--source-account 111122223333 \
--source-arn arn:aws:cloudwatch:us-east-1:111122223333:alarm:alarm-name
```

Atau, Anda dapat membuat sebuah kebijakan yang mirip dengan salah satu contoh berikut dan kemudian menetapkannya ke fungsi tersebut.

Contoh berikut menentukan akun yang menjadi lokasi keberadaan alarm, sehingga hanya alarm-alarm yang ada di akun itu (111122223333) yang dapat menginvokasi fungsi tersebut.

```
{
    "Version": "2012-10-17",
    "Id": "default",
    "Statement": [{
        "Sid": "AlarmAction",
        "Effect": "Allow",
        "Principal": {
            "Service": "lambda.alarms.cloudwatch.amazonaws.com"
        },
        "Action": "lambda:InvokeFunction",
        "Resource": "arn:aws:lambda:us-east-1:444455556666:function:function-name",
        "Condition": {
            "StringEquals": {
                "AWS:SourceAccount": "111122223333"
            }
        }
    }]
}
```

Contoh berikut memiliki cakupan yang lebih sempit, yang memungkinkan hanya alarm yang ditentukan yang ada di akun yang ditentukan saja yang menginvokasi fungsi.

```
{
  "Version": "2012-10-17",
  "Id": "default",
  "Statement": [
    {
      "Sid": "AlarmAction",
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.alarms.cloudwatch.amazonaws.com"
      },
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:us-east-1:444455556666:function:function-name",
      "Condition": {
        "StringEquals": {
          "AWS:SourceAccount": "111122223333",
          "AWS:SourceArn": "arn:aws:cloudwatch:us-east-1:111122223333:alarm:alarm-name"
        }
      }
    }]
}
```

Kami tidak menyarankan Anda membuat sebuah kebijakan yang tidak menentukan akun sumber, karena kebijakan semacam itu akan rentan terhadap masalah deputi yang membingungkan.

Objek acara dikirim dari CloudWatch ke Lambda

Saat Anda mengonfigurasi fungsi Lambda sebagai tindakan alarm, CloudWatch mengirimkan payload JSON ke fungsi Lambda saat memanggil fungsi tersebut. Muatan JSON ini berfungsi sebagai objek peristiwa untuk fungsi tersebut. Anda dapat mengekstrak data dari objek JSON ini dan menggunakannya dalam fungsi Anda. Berikut ini adalah contoh dari sebuah objek peristiwa dari sebuah alarm metrik.

```
{
  'source': 'aws.cloudwatch',
  'alarmArn': 'arn:aws:cloudwatch:us-east-1:444455556666:alarm:lambda-demo-metric-
alarm',
  'accountId': '444455556666',
  'time': '2023-08-04T12:36:15.490+0000',
  'region': 'us-east-1',
  'alarmData': {
    'alarmName': 'lambda-demo-metric-alarm',
    'state': {
      'value': 'ALARM',
      'reason': 'test',
      'timestamp': '2023-08-04T12:36:15.490+0000'
    },
    'previousState': {
      'value': 'INSUFFICIENT_DATA',
      'reason': 'Insufficient Data: 5 datapoints were unknown.',
      'reasonData':
 '{"version":"1.0","queryDate":"2023-08-04T12:31:29.591+0000","statistic":"Average","period":60
[],"threshold":5.0,"evaluatedDatapoints":[{"timestamp":"2023-08-04T12:30:00.000+0000"},
{"timestamp":"2023-08-04T12:29:00.000+0000"},
{"timestamp":"2023-08-04T12:28:00.000+0000"},
{"timestamp":"2023-08-04T12:27:00.000+0000"},
{"timestamp":"2023-08-04T12:26:00.000+0000"}]}',
      'timestamp': '2023-08-04T12:31:29.595+0000'
    },
    'configuration': {
      'description': 'Metric Alarm to test Lambda actions',
      'metrics': [
        {
          'id': '1234e046-06f0-a3da-9534-EXAMPLEe4c',
          'metricStat': {
```

```
            'metric': {
              'namespace': 'AWS/Logs',
              'name': 'CallCount',
              'dimensions': {
                'InstanceId': 'i-12345678'
              }
            },
            'period': 60,
            'stat': 'Average',
            'unit': 'Percent'
          },
          'returnData': True
        }
      ]
    }
  }
}
```

Berikut ini adalah contoh dari sebuah objek peristiwa dari sebuah alarm gabungan.

```
{
  'source': 'aws.cloudwatch',
  'alarmArn': 'arn:aws:cloudwatch:us-east-1:111122223333:alarm:SuppressionDemo.Main',
  'accountId': '111122223333',
  'time': '2023-08-04T12:56:46.138+0000',
  'region': 'us-east-1',
  'alarmData': {
    'alarmName': 'CompositeDemo.Main',
    'state': {
      'value': 'ALARM',
      'reason': 'arn:aws:cloudwatch:us-
east-1:111122223333:alarm:CompositeDemo.FirstChild transitioned to ALARM at Friday 04
 August, 2023 12:54:46 UTC',
      'reasonData': '{"triggeringAlarms":[{"arn":"arn:aws:cloudwatch:us-
east-1:111122223333:alarm:CompositeDemo.FirstChild","state":
{"value":"ALARM","timestamp":"2023-08-04T12:54:46.138+0000"}}]}',
      'timestamp': '2023-08-04T12:56:46.138+0000'
    },
    'previousState': {
      'value': 'ALARM',
      'reason': 'arn:aws:cloudwatch:us-
east-1:111122223333:alarm:CompositeDemo.FirstChild transitioned to ALARM at Friday 04
 August, 2023 12:54:46 UTC',
```

```
      'reasonData': '{"triggeringAlarms":[{"arn":"arn:aws:cloudwatch:us-
east-1:111122223333:alarm:CompositeDemo.FirstChild","state":
{"value":"ALARM","timestamp":"2023-08-04T12:54:46.138+0000"}}]}',
      'timestamp': '2023-08-04T12:54:46.138+0000',
      'actionsSuppressedBy': 'WaitPeriod',
      'actionsSuppressedReason': 'Actions suppressed by WaitPeriod'
    },
    'configuration': {
      'alarmRule': 'ALARM(CompositeDemo.FirstChild) OR
 ALARM(CompositeDemo.SecondChild)',
      'actionsSuppressor': 'CompositeDemo.ActionsSuppressor',
      'actionsSuppressorWaitPeriod': 120,
      'actionsSuppressorExtensionPeriod': 180
    }
  }
}
```

Mengkonfigurasi bagaimana CloudWatch alarm memperlakukan data yang hilang

Terkadang, tidak setiap titik data yang diharapkan untuk metrik dilaporkan CloudWatch. Sebagai contoh, hal ini dapat terjadi ketika koneksi terputus, server rusak, atau ketika metrik melaporkan data hanya secara sementara sesuai desain.

CloudWatch memungkinkan Anda menentukan cara memperlakukan titik data yang hilang saat mengevaluasi alarm. Hal ini akan membantu Anda untuk mengonfigurasi alarm agar statusnya beralih menjadi `ALARM` hanya jika sesuai dengan jenis data yang dipantau. Anda dapat menghindari peringatan palsu ketika data yang hilang tidak menunjukkan adanya masalah.

Mirip dengan bagaimana setiap alarm selalu berada di salah satu dari tiga negara bagian, setiap titik data spesifik dilaporkan CloudWatch termasuk dalam salah satu dari tiga kategori:

- Bukan pelanggaran (dalam ambang batas)
- Pelanggaran (melanggar ambang batas)
- Hilang

Untuk setiap alarm, Anda dapat menentukan CloudWatch untuk memperlakukan titik data yang hilang sebagai salah satu dari berikut ini:

- `notBreaching` – Titik data yang hilang diperlakukan sebagai "baik" dan berada dalam ambang batas,
- `breaching` – Titik data yang hilang diperlakukan sebagai "buruk" dan melanggar ambang batas
- `ignore` – Status alarm saat ini tetap dipertahankan
- `missing` – Jika semua titik data dalam kisaran evaluasi alarm hilang, maka alarm akan beralih menjadi INSUFFIENT_DATA.

Pilihan terbaik bergantung pada jenis metriknya. Untuk sebuah metrik yang terus menerus melaporkan data, seperti `CPUUtilization` dari sebuah instans, Anda mungkin ingin memperlakukan titik data yang hilang sebagai `breaching`, karena mungkin data yang hilang itu menunjukkan bahwa ada yang salah. Namun demikian, untuk sebuah metrik yang menghasilkan titik data hanya ketika kesalahan terjadi, seperti `ThrottledRequests` di Amazon DynamoDB, Anda ingin memperlakukan data yang hilang sebagai `notBreaching`. Perilaku bawaannya adalah `missing`.

Memilih pilihan terbaik untuk alarm-alarm Anda akan mencegah perubahan kondisi alarm yang tidak perlu serta menyesatkan, dan juga akan menunjukkan kesehatan sistem Anda secara lebih akurat.

> **Important**
>
> Alarm-alarm yang mengevaluasi metrik di namespace `AWS/DynamoDB` selalu mengabaikan data yang hilang meskipun Anda memilih opsi berbeda untuk bagaimana alarm tersebut harus memperlakukan data yang hilang. Ketika sebuah metrik `AWS/DynamoDB` memiliki data yang hilang, alarm yang mengevaluasi metrik tersebut akan tetap berada dalam status saat ini.

Cara mengevaluasi status alarm ketika terjadi data hilang

Setiap kali alarm mengevaluasi apakah akan mengubah status, CloudWatch mencoba untuk mengambil jumlah titik data yang lebih tinggi daripada nomor yang ditentukan sebagai Periode Evaluasi. Jumlah pasti titik data yang berusaha diambil oleh CloudWatch bergantung pada lama periode alarm dan apakah didasarkan pada metrik dengan resolusi standar atau resolusi tinggi. Jangka waktu titik data yang berusaha diambil oleh CloudWatch adalah rangkaian evaluasi.

Setelah CloudWatch mengambil titik-titik data ini, hal berikut terjadi:

- Jika tidak ada titik data dalam rentang evaluasi yang hilang, CloudWatch evaluasi alarm berdasarkan titik data terbaru yang dikumpulkan. Jumlah titik data yang dievaluasi setara dengan Periode Evaluasi untuk alarm tersebut. Titik data ekstra dari versi yang lebih jauh dalam rentang evaluasi tidak diperlukan dan akan diabaikan.
- Jika beberapa titik data dalam rentang evaluasi hilang, tetapi jumlah total titik data yang ada yang berhasil diambil dari rentang evaluasi sama dengan atau lebih dari Periode Evaluasi alarm, CloudWatch evaluasi status alarm berdasarkan data nyata terbaru titik yang berhasil diambil, termasuk titik data tambahan yang diperlukan dari jauh ke belakang dalam rentang evaluasi. Dalam kasus ini, nilai yang Anda tetapkan untuk cara memperlakukan data yang hilang tidak diperlukan dan akan diabaikan.
- Jika beberapa titik data dalam rentang evaluasi hilang, dan jumlah titik data aktual yang diambil lebih rendah dari jumlah Periode Evaluasi alarm, CloudWatch isi titik data yang hilang dengan hasil yang Anda tentukan untuk cara memperlakukan data yang hilang, dan kemudian mengevaluasi alarm. Namun, semua titik data nyata dalam rentang evaluasi termasuk dalam evaluasi. CloudWatch menggunakan titik data yang hilang hanya sesedikit mungkin.

> **Note**
>
> Kasus khusus dari perilaku ini adalah bahwa CloudWatch alarm mungkin berulang kali mengevaluasi kembali set titik data terakhir untuk jangka waktu tertentu setelah metrik berhenti mengalir. Evaluasi ulang ini dapat menyebabkan status alarm berubah dan melaksanakan ulang tindakan, jika alarm telah berubah statusnya sesaat sebelum aliran metrik berhenti. Untuk mengurangi perilaku ini, Anda harus menggunakan periode yang lebih singkat.

Tabel berikut menggambarkan beberapa contoh perilaku evaluasi alarm. Pada tabel pertama, Datapoint untuk Alarm dan Periode Evaluasi keduanya 3. CloudWatch mengambil 5 titik data terbaru saat mengevaluasi alarm, jika beberapa dari 3 titik data terbaru hilang. 5 adalah rentang evaluasi untuk alarm.

Kolom 1 menunjukkan 5 titik data terbaru, karena rentang evaluasinya adalah 5. Titik data ini ditampilkan pada titik data terbaru di sebelah kanan. 0 adalah titik data yang tidak terjangkau, X adalah titik data yang dilanggar, dan - adalah titik data yang hilang.

Kolom 2 menunjukkan berapa banyak dari 3 titik data yang diperlukan yang hilang. Meskipun 5 titik data terbaru dievaluasi, hanya 3 (pengaturan untuk Periode Evaluasi) yang diperlukan untuk

mengevaluasi status alarm. Jumlah titik data di Kolom 2 adalah jumlah titik data yang harus "diisi", dengan menggunakan pengaturan tentang cara memperlakukan data yang hilang.

Pada kolom 3-6, header kolom adalah nilai-nilai yang mungkin untuk cara memperlakukan data yang hilang. Baris-baris dalam kolom ini menunjukkan status alarm yang diatur untuk masing-masing cara yang memungkinkan untuk memperlakukan data yang hilang.

Titik data	# titik data yang harus diisi	HILANG	ABAIKAN	MELANGGAR	TIDAK MELANGGAR
0 - X - X	0	OK	OK	OK	OK
0 - - - -	2	OK	OK	OK	OK
- - - - -	3	INSUFFICIENT_DATA	Mempertahankan status saat ini	ALARM	OK
0 X X - X	0	ALARM	ALARM	ALARM	ALARM
- - X - -	2	ALARM	Mempertahankan status saat ini	ALARM	OK

Pada baris kedua tabel sebelumnya, alarm tetap dalam status OK meskipun data yang hilang diperlakukan sebagai pelanggaran, karena satu titik data yang ada tidak melanggar, dan dievaluasi bersama dengan dua titik data yang hilang yang diperlakukan sebagai pelanggaran. Dalam evaluasi alarm yang berikutnya dilakukan, jika data masih hilang maka alarm akan beralih statusnya menjadi ALARM, karena titik data yang tidak melanggar tidak akan lagi berada dalam rentang evaluasi.

Baris ketiga, di mana semua lima titik data terbaru hilang, menggambarkan bagaimana berbagai pengaturan untuk menangani data yang hilang memengaruhi status alarm. Jika kehilangan titik data dianggap telah melanggar, maka alarm akan beralih statusnya menjadi ALARM, sementara jika alarm dianggap tidak melanggar, maka alarm akan beralih statusnya menjadi OK. Jika titik data yang hilang diabaikan, maka alarm akan mempertahankan status saat ini yang dimilikinya sebelum terjadi titik data yang hilang. Dan jika titik data yang hilang hanya dianggap sebagai kehilangan, maka alarm

tidak akan memiliki data nyata terbaru untuk melakukan evaluasi, dan alarm akan beralih statusnya menjadi INSUFFICIENT_DATA.

Di baris keempat, alarm akan beralih statusnya menjadi ALARM dalam semua kasus karena tiga titik data terbaru telah melanggar, serta Periode Evaluasi dan Titik Data untuk Alarm keduanya adalah sama dengan 3. Dalam kasus ini, titik data yang hilang akan diabaikan dan pengaturan tentang cara mengevaluasi data yang hilang tidak lagi diperlukan, karena ada 3 titik data nyata yang harus dievaluasi.

Baris 5 mewakili kasus khusus dalam evaluasi alarm yang disebut sebagai status alarm belum menyala. Untuk informasi selengkapnya, lihat Menghindari peralihan sebelum waktunya pada status alarm.

Pada tabel berikut ini, Periode kembali diatur menjadi 5 menit, dan Titik Data untuk Alarm hanya 2, sedangkan Periode Evaluasi adalah 3. Ini adalah alarm 2 dari 3, M dari N.

Rentang evaluasinya adalah 5. Ini adalah jumlah maksimum titik data terbaru yang diambil dan dapat digunakan jika ada beberapa titik data yang hilang.

Titik data	# titik data yang hilang	HILANG	ABAIKAN	MELANGGAR	TIDAK MELANGGAR
0 - X - X	0	ALARM	ALARM	ALARM	ALARM
0 0 X 0 X	0	ALARM	ALARM	ALARM	ALARM
0 - X - -	1	OK	OK	ALARM	OK
- - - - 0	2	OK	OK	ALARM	OK
- - - X -	2	ALARM	Mempertahankan status saat ini	ALARM	OK

Pada baris 1 dan 2, alarm tersebut selalu berada dalam status ALARM karena 2 dari 3 titik data terbaru dilanggar. Pada baris 2, dua titik data tertua dalam rentang evaluasi tidak diperlukan karena tidak ada dari 3 titik data terbaru yang hilang, sehingga dua titik data yang lama ini diabaikan.

Pada baris 3 dan 4, alarm tersebut beralih statusnya menjadi ALARM hanya jika data yang hilang diperlakukan sebagai pelanggaran, dalam hal ini dua titik data terbaru yang hilang diperlakukan sebagai pelanggaran. Pada baris 4, dua titik data hilang yang diperlakukan sebagai pelanggaran tersebut memberikan dua titik data yang melanggar yang diperlukan untuk memicu status ALARM.

Baris 5 mewakili kasus khusus dalam evaluasi alarm yang disebut sebagai status alarm belum menyala. Untuk informasi selengkapnya, silakan lihat bagian berikut ini.

Menghindari peralihan sebelum waktunya pada status alarm

CloudWatch evaluasi alarm mencakup logika untuk mencoba menghindari alarm palsu, di mana alarm masuk ke keadaan ALARM sebelum waktunya ketika data terputus-putus. Contoh yang ditunjukkan di baris 5 dalam tabel yang ada di bagian sebelumnya menggambarkan logika ini. Dalam baris tersebut, dan dalam contoh berikut, Periode Evaluasi adalah 3 dan rentang evaluasinya adalah 5 titik data. Titik data untuk Alarm adalah 3, kecuali untuk contoh M dari N, di mana Titik data untuk Alarm adalah 2.

Misalkan data terbaru sebuah alarm adalah - - - - X, dengan empat titik data yang hilang dan kemudian pelanggaran titik data sebagai titik data terbaru. Karena titik data berikutnya mungkin tidak melanggar, maka alarm tersebut tidak langsung beralih statusnya menjadi ALARM ketika data baik - - - - X maupun - - - X - dan Titik data untuk Alarm adalah 3. Dengan cara ini, positif palsu akan bisa dihindari ketika titik data berikutnya tidak melanggar dan menyebabkan data menjadi - - - X 0 atau - - X - 0.

Namun demikian, jika beberapa titik data terakhir adalah - - X - -, maka alarm akan beralih statusnya menjadi ALARM bahkan jika titik data yang hilang diperlakukan sebagai hilang. Hal ini karena alarm dirancang untuk selalu berada dalam status ALARM ketika titik data pelanggaran yang paling lama tersedia selama jumlah titik data Periode Evaluasi paling tidak seumur dengan nilai Titik data untuk Alarm, dan semua titik data terbaru lainnya melanggar atau hilang. Dalam kasus ini, alarm tersebut beralih statusnya menjadi ALARM meskipun total titik data yang tersedia lebih rendah dari M (Titik data untuk Alarm).

Logika alarm ini juga berlaku untuk alarm M dari N. Jika titik data pelanggaran paling lama selama rentang evaluasi paling tidak seumur dengan nilai Titik data untuk Alarm, dan semua titik data yang lebih baru akan melanggar atau hilang, maka alarm akan beralih statusnya menjadi ALARM tidak peduli nilai M (Titik data untuk Alarm).

Alarm-alarm resolusi tinggi

Jika Anda mengatur sebuah alarm pada metrik resolusi tinggi, maka Anda dapat menentukan alarm resolusi tinggi dengan periode 10 detik atau 30 detik, atau Anda dapat mengatur sebuah alarm biasa dengan periode lebih dari 60 detik. Ada beban yang lebih tinggi untuk alarm-alarm dengan resolusi tinggi. Untuk informasi selengkapnya tentang metrik resolusi tinggi, silakan lihat Menerbitkan metrik kustom.

Alarm-alarm tentang ekspresi matematika

Anda dapat mengatur alarm pada hasil ekspresi matematika yang didasarkan pada satu atau beberapa CloudWatch metrik. Sebuah ekspresi matematika yang digunakan untuk sebuah alarm dapat menyertakan sebanyak 10 metrik. Masing-masing metrik tersebut harus menggunakan periode yang sama.

Untuk alarm berdasarkan ekspresi matematika, Anda dapat menentukan bagaimana Anda CloudWatch ingin memperlakukan titik data yang hilang. Dalam hal ini, titik data akan dianggap hilang jika ekspresi matematika tersebut tidak mengembalikan nilai untuk titik data itu.

Alarm-alarm yang didasarkan pada ekspresi matematika tidak dapat melakukan tindakan-tindakan Amazon EC2.

Untuk informasi selengkapnya tentang ekspresi matematika metrik dan sintaks, silakan lihat Gunakan matematika metrik.

CloudWatch Alarm berbasis persentil dan sampel data rendah

Ketika Anda mengatur persentil sebagai statistik untuk sebuah alarm, Anda dapat menentukan apa yang harus dilakukan ketika tidak terdapat data yang cukup untuk dilakukannya penilaian statistik yang baik. Anda dapat memilih untuk membuat alarm yang mengevaluasi statistik dan mungkin mengubah status alarm. Atau, Anda dapat membuat alarm yang mengabaikan metrik ketika ukuran sampel kecil, dan menunggu untuk melakukan evaluasi sampai tersedia cukup data yang signifikan secara statistik.

Untuk persentil antara 0,5 (inklusif) dan 1,00 (eksklusif), pengaturan ini digunakan apabila terdapat titik data kurang dari 10/(1-persentil) selama periode evaluasi. Sebagai contoh, pengaturan ini akan digunakan jika terdapat sampel kurang dari 1000 sampel untuk alarm pada persentil p99. Untuk

persentil antara 0 dan 0,5 (eksklusif), pengaturan tersebut digunakan ketika titik data kurang dari 10/ persentil.

Fitur umum CloudWatch alarm

Fitur-fitur berikut berlaku untuk semua CloudWatch alarm:

- Tidak ada batasan jumlah alarm yang dapat Anda buat. Untuk membuat atau memperbarui alarm, Anda menggunakan CloudWatch konsol, tindakan PutMetricAlarmAPI, atau put-metric-alarmperintah di AWS CLI.
- Nama alarm harus menggunakan karakter UTF-8, dan tidak dapat berisi karakter kontrol ASCII
- Anda dapat mencantumkan salah satu atau semua alarm yang saat ini dikonfigurasi, dan mencantumkan alarm apa pun dalam status tertentu dengan menggunakan CloudWatch konsol, tindakan DescribeAlarmsAPI, atau perintah deskripsikan alarm di. AWS CLI
- Anda dapat menonaktifkan dan mengaktifkan alarm dengan menggunakan tindakan DisableAlarmActionsdan EnableAlarmActionsAPI, atau enable-alarm-actionsperintah disable-alarm-actionsdan di AWS CLI.
- Anda dapat menguji alarm dengan menyetelnya ke status apa pun menggunakan tindakan SetAlarmStateAPI atau set-alarm-stateperintah di AWS CLI. Perubahan status sementara ini hanya berlangsung hingga perbandingan alarm berikutnya terjadi.
- Anda dapat membuat sebuah alarm untuk metrik kustom sebelum membuat metrik kustom tersebut. Agar alarm tersebut berfungsi dengan benar, Anda harus menyertakan semua dimensi untuk metrik kustom selain namespace metrik dan nama metrik dalam definisi alarm. Untuk melakukan ini, Anda dapat menggunakan tindakan PutMetricAlarmAPI, atau put-metric-alarmperintah di AWS CLI.
- Anda dapat melihat riwayat alarm menggunakan CloudWatch konsol, tindakan DescribeAlarmHistoryAPI, atau describe-alarm-historyperintah di AWS CLI. CloudWatch mempertahankan riwayat alarm selama dua minggu. Setiap transisi status akan ditandai dengan stempel waktu yang unik. Dalam kasus yang jarang terjadi, riwayat Anda mungkin menampilkan lebih dari satu notifikasi perubahan status. Dengan stempel waktu tersebut, Anda dapat mengonfirmasi perubahan status yang unik.
- Anda dapat memfavoritkan alarm dari opsi Favorit dan terbaru di panel navigasi CloudWatch konsol dengan mengarahkan kursor ke alarm yang ingin Anda favoritkan dan memilih simbol bintang di sebelahnya.

- Jumlah periode evaluasi untuk sebuah alarm dikalikan dengan panjang setiap periode evaluasi tidak boleh melebihi satu hari.

> **Note**
>
> Beberapa AWS sumber daya tidak mengirim data metrik CloudWatch dalam kondisi tertentu. Sebagai contoh, Amazon EBS mungkin tidak akan mengirimkan data metrik untuk volume tersedia yang tidak dilampirkan pada instans Amazon EC2, karena tidak ada aktivitas metrik yang perlu dipantau untuk volume tersebut. Jika Anda memiliki sebuah alarm yang diatur untuk metrik seperti itu, maka Anda mungkin akan melihat statusnya berubah menjadi `INSUFFICIENT_DATA`. Hal ini dapat menunjukkan bahwa sumber daya Anda tidak aktif, dan mungkin tidak selalu berarti bahwa ada masalah yang sedang terjadi. Anda dapat menentukan bagaimana masing-masing alarm memperlakukan data yang hilang. Untuk informasi selengkapnya, lihat Mengkonfigurasi bagaimana CloudWatch alarm memperlakukan data yang hilang.

Rekomendasi alarm praktik terbaik untuk AWS layanan

CloudWatch memberikan rekomendasi alarm out-of-the kotak. Ini adalah CloudWatch alarm yang kami sarankan agar Anda buat untuk metrik yang diterbitkan oleh layanan lain AWS . Rekomendasi ini dapat membantu Anda dalam mengidentifikasi metrik yang harus Anda tetapkan sebagai peringatan alarm untuk mengikuti praktik terbaik dalam melakukan pemantauan. Rekomendasi tersebut juga menyarankan ambang batas alarm yang dapat Anda tetapkan. Mengikuti rekomendasi ini dapat membantu Anda tidak melewatkan pemantauan penting AWS infrastruktur Anda.

Untuk menemukan rekomendasi alarm, Anda menggunakan bagian metrik CloudWatch konsol, dan pilih sakelar filter rekomendasi alarm. Jika Anda menavigasi ke alarm yang disarankan di konsol dan kemudian membuat alarm yang disarankan, CloudWatch dapat pra-mengisi beberapa pengaturan alarm. Untuk beberapa alarm yang direkomendasikan, nilai ambang batas alarm juga sudah diisi sebelumnya. Anda juga dapat menggunakan konsol untuk mengunduh definisi infrastructure-as-code alarm untuk alarm yang direkomendasikan, dan kemudian menggunakan kode ini untuk membuat alarm di AWS CloudFormation, the AWS CLI, atau Terraform.

Anda juga dapat melihat daftar alarm-alarm yang direkomendasikan di Alarm-alarm yang direkomendasikan.

Anda dikenakan biaya untuk alarm yang Anda buat, dengan kecepatan yang sama seperti alarm lain yang Anda buat. CloudWatch Menggunakan rekomendasi tidak akan dikenakan biaya tambahan. Untuk informasi selengkapnya, lihat CloudWatch Harga Amazon.

Temukan dan buat alarm-alarm yang direkomendasikan

Ikuti langkah-langkah berikut untuk menemukan metrik yang CloudWatch menyarankan agar Anda mengatur alarm, dan secara opsional untuk membuat salah satu alarm ini. Prosedur pertama menjelaskan tentang cara menemukan metrik-metrik yang merekomendasikan alarm, dan cara membuat salah satu alarm ini.

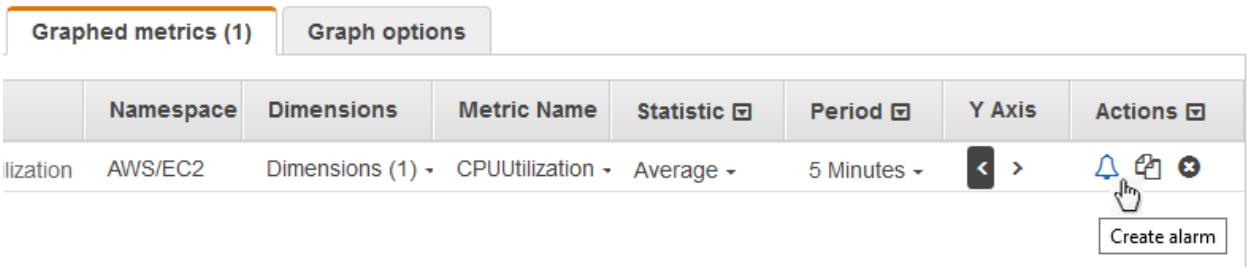
Anda juga bisa mendapatkan unduhan massal definisi infrastructure-as-code alarm untuk semua alarm yang direkomendasikan di AWS namespace, seperti atau. `AWS/Lambda` `AWS/S3` Instruksi-instruksi itu nanti akan dijelaskan dalam topik ini.

Untuk menemukan metrik-metrik dengan alarm yang disarankan, dan membuat satu alarm yang direkomendasikan

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik, Semua metrik.
3. Di atas tabel Metrik, silakan Pilih Rekomendasi alarm.

   Daftar namespace metrik difilter untuk menyertakan hanya metrik-metrik yang memiliki rekomendasi alarm dan yang memberikan layanan di akun yang Anda publikasikan.
4. Pilih namespace untuk sebuah layanan.

   Daftar metrik dalam namespace ini difilter untuk menyertakan hanya metrik-metrik yang memiliki rekomendasi alarm.
5. Untuk melihat maksud alarm dan ambang batas yang direkomendasikan untuk sebuah metrik, silakan pilih Lihat detail.
6. Untuk membuat sebuah alarm untuk salah satu metrik, lakukan salah satu hal berikut ini:
   - Untuk menggunakan konsol untuk membuat alarm tersebut, lakukan hal berikut ini:
     a. Pilih kotak centang untuk metrik tersebut dan pilih tab Metrik bergrafik.
     b. Pilih ikon alarm.



Panduan pembuatan alarm ditampilkan, dengan nama metrik, statistik, dan periode yang sudah diisi berdasarkan rekomendasi alarm. Jika rekomendasi tersebut mencakup nilai ambang batas tertentu, maka nilai itu juga sudah akan diisi sebelumnya.

c. Pilih Berikutnya.

d. Pada Notifikasi, silakan pilih topik SNS yang akan diberitahukan sebagai notifikasi ketika alarm beralih statusnya menjadi `ALARM`, `OK`, atau menjadi `INSUFFICIENT_DATA`.

Agar alarm mengirimkan beberapa notifikasi untuk status alarm yang sama atau untuk status alarm yang berbeda, silakan pilih Tambahkan notifikasi.

Agar alarm tidak mengirim notifikasi, silakan pilih Hapus.

e. Agar alarm dapat melakukan tindakan penskalaan otomatis (Auto Scaling) atau EC2, silakan pilih tombol yang sesuai dan pilih status alarm serta tindakan yang harus dilakukan.

f. Setelah selesai, silakan pilih Berikutnya.

g. Masukkan sebuah nama dan deskripsi untuk alarm tersebut. Nama harus menggunakan karakter ASCII saja. Lalu pilih Berikutnya.

h. Pada Pratinjau dan buat, konfirmasikan bahwa informasi dan ketentuannya sudah sesuai keinginan Anda, kemudian pilih Buat alarm.

- Untuk mengunduh definisi infrastructure-as-code alarm untuk digunakan di salah satu AWS CloudFormation, AWS CLI, atau Terraform, pilih Unduh kode alarm dan pilih format yang Anda inginkan. Kode yang sudah Anda unduh ini akan memiliki pengaturan yang disarankan untuk nama metrik, statistik, dan ambang batas.

Untuk mengunduh definisi infrastructure-as-code alarm untuk semua alarm yang direkomendasikan untuk suatu layanan AWS

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.

2. Pada panel navigasi, silakan pilih Metrik, Semua metrik.
3. Di atas tabel Metrik, silakan Pilih Rekomendasi alarm.

   Daftar namespace metrik difilter untuk menyertakan hanya metrik-metrik yang memiliki rekomendasi alarm dan yang memberikan layanan di akun yang Anda publikasikan.
4. Pilih namespace untuk sebuah layanan.

   Daftar metrik dalam namespace ini difilter untuk menyertakan hanya metrik-metrik yang memiliki rekomendasi alarm.
5. Kode alarm Download menampilkan berapa banyak alarm yang direkomendasikan untuk metrik di namespace ini. Untuk mengunduh definisi infrastructure-as-code alarm untuk semua alarm yang direkomendasikan, pilih Unduh kode alarm dan kemudian pilih format kode yang Anda inginkan.

Alarm-alarm yang direkomendasikan

Bagian-bagian berikut akan mencantumkan metrik-metrik yang kami sarankan untuk diatur dalam alarm Anda sesuai praktik terbaik. Untuk setiap metrik, dimensi, maksud alarm, ambang batas yang direkomendasikan, pembenaran ambang batas, dan panjang periode serta jumlah titik data, juga akan ditampilkan.

Beberapa metrik mungkin muncul dua kali dalam daftar tersebut. Hal ini terjadi ketika alarm yang berbeda direkomendasikan untuk kombinasi dimensi metrik yang berbeda.

Titik data ke alarm adalah jumlah titik data yang harus dilanggar agar alarm beralih statusnya menjadi ALARM. Periode evaluasi adalah jumlah periode yang akan diperhitungkan saat dievaluasi alarm dilakukan. Jika angka-angka ini sama, maka alarm akan beralih statusnya menjadi ALARM hanya ketika jumlah periode berturut-turut memiliki nilai yang melanggar ambang batas yang telah ditetapkan. Jika Titik data ke alarm lebih rendah dari Periode evaluasi, maka itu adalah alarm "M out of N" dan alarm tersebut akan beralih statusnya menjadi ALARM jika setidaknya titik data Titik data ke alarm melanggar dalam setiap rangkaian Periode evaluasi titik data. Untuk informasi selengkapnya, lihat Melakukan evaluasi alarm.

Topik

- Amazon API Gateway
- Amazon EC2 Auto Scaling
- Amazon CloudFront

- Amazon Cognito
- Amazon DynamoDB
- Amazon EBS
- Amazon EC2
- Amazon ElastiCache
- Amazon EC2 (AWS/ElasticGPUs)
- Amazon ECS
- Amazon ECS dengan Wawasan Kontainer
- Amazon EFS
- Amazon EKS dengan Wawasan Kontainer
- Amazon Kinesis Data Streams
- Lambda
- Wawasan Lambda
- Amazon VPC (AWS/NATGateway)
- AWS Tautan Pribadi (AWS/PrivateLinkEndpoints)
- AWS Tautan Pribadi (AWS/PrivateLinkServices)
- Amazon RDS
- Amazon Route 53 Public Data Plane
- Amazon S3
- S3ObjectLambda
- Amazon SNS
- Amazon SQS
- AWS VPN

Amazon API Gateway

4XXError

Dimensi:ApiName, Panggung

Deskripsi alarm: Alarm ini mendeteksi tingkat kesalahan sisi klien yang tinggi. Hal ini dapat menunjukkan adanya masalah dalam parameter otorisasi atau permintaan klien. Hal ini juga

bisa berarti bahwa sebuah sumber daya telah dihapus atau klien meminta sesuatu yang tidak ada. Pertimbangkan untuk mengaktifkan CloudWatch Log dan memeriksa kesalahan apa pun yang mungkin menyebabkan kesalahan 4XX. Selain itu, pertimbangkan untuk mengaktifkan CloudWatch metrik terperinci untuk melihat metrik ini per sumber daya dan metode dan mempersempit sumber kesalahan. Kesalahan juga dapat disebabkan karena dilanggarnya batas throttling yang telah dikonfigurasi sebelumnya. Jika respons dan log melaporkan tingkat kesalahan sebanyak 429, jumlah yang tinggi dan tidak terduga, maka Anda harus mengikuti panduan ini untuk memecahkan masalah tersebut.

Maksud: Alarm ini dapat mendeteksi tingkat kesalahan sisi klien yang tinggi untuk permintaan API Gateway.

Statistik: Rata-rata

Ambang batas yang disarankan: 0,05

Pembenaran ambang batas: Ambang batas yang disarankan yang akan mendeteksi ketika lebih dari 5% dari total permintaan mendapatkan kesalahan 4XX. Namun demikian, Anda dapat menyetel ambang batas tersebut agar sesuai dengan lalu lintas permintaan serta tingkat kesalahan yang dapat diterima. Anda juga dapat menganalisis data historis untuk menentukan tingkat kesalahan yang dapat diterima untuk beban kerja aplikasi dan kemudian menyetel ambang batas yang sesuai. Kesalahan 4XX yang sering terjadi perlu Anda waspadai. Namun demikian, menetapkan nilai yang sangat rendah untuk ambang batas akan dapat menyebabkan alarm menjadi terlalu sensitif.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

5XXError

Dimensi:ApiName, Panggung

Deskripsi alarm: Alarm ini akan membantu dalam mendeteksi tingkat kesalahan sisi server yang tinggi. Hal ini dapat menunjukkan bahwa ada sesuatu yang salah pada backend API, jaringan, atau integrasi antara gateway API dan backend API. Dokumentasi ini dapat membantu Anda dalam memecahkan masalah yang menjadi penyebab terjadinya kesalahan 5xx.

Maksud: Alarm ini dapat mendeteksi tingkat kesalahan sisi server yang tinggi untuk permintaan API Gateway.

Statistik: Rata-rata

Ambang batas yang disarankan: 0,05

Pembenaran ambang batas: Ambang batas yang disarankan yang akan mendeteksi ketika lebih dari 5% dari total permintaan mendapatkan kesalahan 5XX. Namun demikian, Anda dapat menyetel ambang batas tersebut agar sesuai dengan lalu lintas permintaan serta tingkat kesalahan yang dapat diterima. Anda juga dapat menganalisis data historis untuk menentukan tingkat kesalahan yang dapat diterima untuk beban kerja aplikasi dan kemudian menyetel ambang batas yang sesuai dengan itu. Kesalahan 5XX yang sering terjadi perlu Anda waspadai. Namun demikian, menetapkan nilai yang sangat rendah untuk ambang batas akan dapat menyebabkan alarm menjadi terlalu sensitif.

Periode: 60

Titik data untuk alarm: 3

Periode evaluasi: 3

Operator Perbandingan: GREATER_THAN_THRESHOLD

Hitungan

Dimensi:ApiName, Panggung

Deskripsi alarm: Alarm ini akan membantu dalam mendeteksi volume lalu lintas rendah untuk tahap API REST. Hal ini bisa menjadi indikator akan adanya masalah dengan aplikasi yang memanggil API, seperti misalnya penggunaan titik akhir yang salah. Ini juga bisa menjadi indikator dari terjadinya masalah dengan konfigurasi atau izin API sehingga tidak dapat dijangkau oleh klien.

Maksud: Alarm ini dapat mendeteksi volume lalu lintas rendah secara tak terduga untuk tahap API REST. Kami menyarankan Anda untuk membuat alarm ini jika API Anda menerima jumlah permintaan yang dapat diprediksi dan konsisten dalam kondisi normal. Jika Anda mengaktifkan CloudWatch metrik terperinci dan Anda dapat memprediksi volume lalu lintas normal per metode dan sumber daya, kami sarankan Anda membuat alarm alternatif agar pemantauan penurunan volume lalu lintas yang lebih halus untuk setiap sumber daya dan metode. Alarm ini tidak disarankan untuk API yang diperkirakan memiliki lalu lintas yang tidak konstan dan konsisten.

Statistik: SampleCount

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Tetapkan ambang batas berdasarkan analisis data historis untuk menentukan jumlah permintaan dasar yang Anda perkirakan untuk API Anda. Menetapkan ambang batas pada nilai yang sangat tinggi akan dapat menyebabkan alarm menjadi terlalu sensitif pada periode lalu lintas normal dan perkiraan lalu lintas rendah. Sebaliknya, menetapkan ambang batas pada nilai yang sangat rendah akan dapat menyebabkan alarm tidak mendeteksi penurunan volume lalu lintas yang lebih kecil.

Periode: 60

Titik data untuk alarm: 10

Periode evaluasi: 10

Operator Perbandingan: LESS_THAN_THRESHOLD

Hitungan

Dimensi:ApiName, Panggung, Sumber Daya, Metode

Deskripsi alarm: Alarm ini akan membantu dalam mendeteksi volume lalu lintas rendah untuk sumber daya dan metode API REST dalam tahap. Hal ini bisa menunjukkan adanya masalah dengan aplikasi yang memanggil API, seperti misalnya penggunaan titik akhir yang salah. Ini juga bisa menjadi indikator dari terjadinya masalah dengan konfigurasi atau izin API sehingga tidak dapat dijangkau oleh klien.

Maksud: Alarm ini dapat mendeteksi volume lalu lintas rendah secara tak terduga untuk sumber daya dan metode API REST dalam tahap. Kami menyarankan Anda untuk membuat alarm ini jika API Anda menerima jumlah permintaan yang dapat diprediksi dan konsisten dalam kondisi normal. Alarm ini tidak disarankan untuk API yang diperkirakan memiliki lalu lintas yang tidak konstan dan konsisten.

Statistik: SampleCount

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Tetapkan ambang batas berdasarkan analisis data historis untuk menentukan jumlah permintaan dasar yang Anda perkirakan untuk API Anda. Menetapkan

ambang batas pada nilai yang sangat tinggi akan dapat menyebabkan alarm menjadi terlalu sensitif pada periode lalu lintas normal dan perkiraan lalu lintas rendah. Sebaliknya, menetapkan ambang batas pada nilai yang sangat rendah akan dapat menyebabkan alarm tidak mendeteksi penurunan volume lalu lintas yang lebih kecil.

Periode: 60

Titik data untuk alarm: 10

Periode evaluasi: 10

Operator Perbandingan: LESS_THAN_THRESHOLD

Hitungan

Dimensi:ApiId, Panggung

Deskripsi alarm: Alarm ini akan membantu dalam mendeteksi volume lalu lintas rendah untuk tahap API HTTP. Hal ini bisa menunjukkan adanya masalah dengan aplikasi yang memanggil API, seperti misalnya penggunaan titik akhir yang salah. Ini juga bisa menjadi indikator dari terjadinya masalah dengan konfigurasi atau izin API sehingga tidak dapat dijangkau oleh klien.

Maksud: Alarm ini dapat mendeteksi volume lalu lintas rendah secara tak terduga untuk tahap API HTTP. Kami menyarankan Anda untuk membuat alarm ini jika API Anda menerima jumlah permintaan yang dapat diprediksi dan konsisten dalam kondisi normal. Jika Anda mengaktifkan CloudWatch metrik terperinci dan Anda dapat memprediksi volume lalu lintas normal per rute, kami sarankan Anda membuat alarm alternatif untuk ini agar pemantauan penurunan volume lalu lintas yang lebih halus untuk setiap rute. Alarm ini tidak disarankan untuk API yang diperkirakan memiliki lalu lintas yang tidak konstan dan konsisten.

Statistik: SampleCount

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Tetapkan nilai ambang batas berdasarkan analisis data historis untuk menentukan jumlah permintaan dasar yang Anda perkirakan untuk API Anda. Menetapkan ambang batas pada nilai yang sangat tinggi akan dapat menyebabkan alarm menjadi terlalu sensitif pada periode lalu lintas normal dan perkiraan lalu lintas rendah. Sebaliknya, menetapkan ambang batas pada nilai yang sangat rendah akan dapat menyebabkan alarm tidak mendeteksi penurunan volume lalu lintas yang lebih kecil.

Periode: 60

Titik data untuk alarm: 10

Periode evaluasi: 10

Operator Perbandingan: LESS_THAN_THRESHOLD

Hitungan

Dimensi:ApiId, Panggung, Sumber Daya, Metode

Deskripsi alarm: Alarm ini akan membantu Anda dalam mendeteksi volume lalu lintas rendah untuk rute API HTTP dalam tahap. Hal ini bisa menunjukkan adanya masalah dengan aplikasi yang memanggil API, seperti misalnya penggunaan titik akhir yang salah. Hal ini juga bisa menunjukkan adanya masalah dengan konfigurasi atau izin API sehingga tidak dapat dijangkau oleh klien.

Maksud: Alarm ini dapat mendeteksi volume lalu lintas rendah secara tak terduga untuk rute API HTTP dalam tahap. Kami menyarankan Anda untuk membuat alarm ini jika API Anda menerima jumlah permintaan yang dapat diprediksi dan konsisten dalam kondisi normal. Alarm ini tidak disarankan untuk API yang diperkirakan memiliki lalu lintas yang tidak konstan dan konsisten.

Statistik: SampleCount

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Tetapkan nilai ambang batas berdasarkan analisis data historis untuk menentukan jumlah permintaan dasar yang Anda perkirakan untuk API Anda. Menetapkan ambang batas pada nilai yang sangat tinggi akan dapat menyebabkan alarm menjadi terlalu sensitif pada periode lalu lintas normal dan perkiraan lalu lintas rendah. Sebaliknya, menetapkan ambang batas pada nilai yang sangat rendah akan dapat menyebabkan alarm tidak mendeteksi penurunan volume lalu lintas yang lebih kecil.

Periode: 60

Titik data untuk alarm: 10

Periode evaluasi: 10

Operator Perbandingan: LESS_THAN_THRESHOLD

IntegrationLatency

Dimensi:ApiId, Panggung

Deskripsi alarm: Alarm ini akan membantu Anda dalam mendeteksi apakah ada latensi integrasi tinggi untuk permintaan API dalam suatu tahap. Anda dapat mengkorelasikan nilai metrik `IntegrationLatency` dengan metrik latensi yang sesuai dari backend Anda seperti metrik `Duration` untuk integrasi Lambda. Hal ini membantu Anda dalam menentukan apakah API backend membutuhkan lebih banyak waktu untuk memproses permintaan dari klien karena masalah performa, atau apakah ada overhead lain dari inisialisasi atau cold start. Selain itu, pertimbangkan untuk mengaktifkan CloudWatch Log untuk API Anda dan memeriksa log untuk setiap kesalahan yang mungkin menyebabkan masalah latensi tinggi. Selain itu, pertimbangkan untuk mengaktifkan CloudWatch metrik terperinci untuk mendapatkan tampilan metrik ini per rute, untuk membantu Anda mempersempit sumber latensi integrasi.

Maksud: Alarm ini dapat mendeteksi kapan permintaan API Gateway dalam satu tahap memiliki latensi integrasi yang tinggi. Kami merekomendasikan alarm ini untuk WebSocket API, dan kami menganggapnya opsional untuk API HTTP karena mereka sudah memiliki rekomendasi alarm terpisah untuk metrik Latensi. Jika CloudWatch metrik mendetail telah diaktifkan dan Anda memiliki persyaratan kinerja latensi integrasi yang berbeda per rute, kami sarankan Anda membuat alarm alternatif agar memiliki pemantauan latensi integrasi yang lebih halus untuk setiap rute.

Statistik: p90

Ambang batas yang disarankan: 2000,0

Pembenaran ambang batas: Nilai ambang yang disarankan tidak akan berfungsi untuk semua beban kerja API. Namun demikian, Anda dapat menggunakannya sebagai titik awal untuk ambang batas tersebut. Anda kemudian dapat memilih nilai ambang batas yang berbeda berdasarkan beban kerja dan persyaratan latensi, performa, dan SLA yang dapat diterima untuk API. Jika API dapat diterima untuk memiliki latensi yang lebih tinggi secara umum, maka Anda harus menetapkan nilai ambang batas yang lebih tinggi untuk membuat alarm tersebut menjadi kurang sensitif. Namun demikian, jika API tersebut Anda harapkan dapat memberikan respons mendekati waktu nyata, maka Anda harus menetapkan nilai ambang batas yang lebih rendah. Anda juga dapat menganalisis data historis untuk menentukan latensi dasar yang diharapkan untuk beban kerja aplikasi, dan kemudian menggunakannya untuk menyetel nilai ambang batas yang sesuai.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

IntegrationLatency

Dimensi:ApiId, Panggung, Rute

Deskripsi alarm: Alarm ini membantu mendeteksi apakah ada latensi integrasi tinggi untuk permintaan WebSocket API untuk rute dalam satu tahap. Anda dapat mengkorelasikan nilai metrik `IntegrationLatency` dengan metrik latensi yang sesuai dari backend Anda seperti metrik `Duration` untuk integrasi Lambda. Hal ini akan membantu Anda dalam menentukan apakah API backend membutuhkan lebih banyak waktu untuk memproses permintaan dari klien karena adanya masalah performa atau apakah ada beberapa overhead lain dari inisialisasi atau cold start. Selain itu, pertimbangkan untuk mengaktifkan CloudWatch Log untuk API Anda dan memeriksa log untuk setiap kesalahan yang mungkin menyebabkan masalah latensi tinggi.

Maksud: Alarm ini dapat mendeteksi kapan permintaan API Gateway untuk sebuah rute dalam satu tahap memiliki latensi integrasi yang tinggi.

Statistik: p90

Ambang batas yang disarankan: 2000,0

Pembenaran ambang batas: Nilai ambang yang disarankan tidak akan berfungsi untuk semua beban kerja API. Namun demikian, Anda dapat menggunakannya sebagai titik awal untuk ambang batas tersebut. Anda kemudian dapat memilih nilai ambang batas yang berbeda berdasarkan beban kerja dan persyaratan latensi, performa, dan SLA yang dapat diterima untuk API. Jika API dapat diterima untuk memiliki latensi yang lebih tinggi secara umum, maka Anda dapat menetapkan nilai ambang batas yang lebih tinggi untuk membuat alarm tersebut menjadi tidak begitu sensitif. Namun demikian, jika API tersebut Anda harapkan dapat memberikan respons mendekati waktu nyata, maka Anda harus menetapkan nilai ambang batas yang lebih rendah. Anda juga dapat menganalisis data historis untuk menentukan latensi dasar yang diharapkan untuk beban kerja aplikasi, dan kemudian menggunakannya untuk menyetel nilai ambang batas yang sesuai.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

Latensi

Dimensi:ApiName, Panggung

Deskripsi alarm: Alarm ini akan mendeteksi latensi tinggi dalam satu tahap. Temukan nilai metrik `IntegrationLatency` untuk memeriksa latensi API backend. Jika kedua metrik sebagian besar selaras, maka API backend akan menjadi sumber latensi yang lebih tinggi dan Anda harus menyelidiki masalah yang terjadi di sana. Pertimbangkan juga mengaktifkan CloudWatch Log dan memeriksa kesalahan yang mungkin menyebabkan latensi tinggi. Selain itu, pertimbangkan untuk mengaktifkan CloudWatch metrik terperinci untuk melihat metrik ini per sumber daya dan metode dan mempersempit sumber latensi. Jika perlu, silakan lihat pemecahan masalah dengan Lambda atau panduan pemecahan masalah untuk titik akhir API yang dioptimalkan-edge.

Maksud: Alarm ini dapat mendeteksi kapan permintaan API Gateway dalam satu tahap memiliki latensi yang tinggi. Jika CloudWatch metrik mendetail telah diaktifkan dan Anda memiliki persyaratan kinerja latensi yang berbeda untuk setiap metode dan sumber daya, sebaiknya Anda membuat alarm alternatif agar pemantauan latensi yang lebih halus untuk setiap sumber daya dan metode.

Statistik: p90

Ambang batas yang disarankan: 2500,0

Pembenaran ambang batas: Nilai ambang batas yang disarankan tidak akan berfungsi untuk semua beban kerja API. Namun demikian, Anda dapat menggunakannya sebagai titik awal untuk ambang batas tersebut. Anda kemudian dapat memilih nilai ambang batas yang berbeda berdasarkan beban kerja dan persyaratan latensi, performa, dan SLA yang dapat diterima untuk API. Jika API dapat diterima untuk memiliki latensi yang lebih tinggi secara umum, maka Anda dapat menetapkan nilai ambang batas yang lebih tinggi untuk membuat alarm tersebut menjadi tidak begitu sensitif. Namun demikian, jika API tersebut Anda harapkan dapat memberikan respons mendekati waktu nyata, maka Anda harus menetapkan nilai ambang batas yang lebih rendah. Anda juga dapat menganalisis data historis untuk menentukan apa latensi dasar yang diharapkan untuk beban kerja aplikasi dan kemudian menggunakannya untuk menyetel nilai ambang batas yang sesuai.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

Latensi

Dimensi:ApiName, Panggung, Sumber Daya, Metode

Deskripsi alarm: Alarm ini dapat mendeteksi latensi tinggi untuk sebuah sumber daya dan metode dalam satu tahap. Temukan nilai metrik `IntegrationLatency` untuk memeriksa latensi API backend. Jika kedua metrik sebagian besar selaras, maka API backend akan menjadi sumber latensi yang lebih tinggi dan Anda harus menyelidiki masalah performa yang terjadi di sana. Pertimbangkan juga mengaktifkan CloudWatch Log dan memeriksa kesalahan apa pun yang mungkin menyebabkan latensi tinggi. Jika perlu, Anda juga dapat melihat pemecahan masalah dengan Lambda atau panduan pemecahan masalah untuk titik akhir API yang dioptimalkan-edge.

Maksud: Alarm ini dapat mendeteksi kapan permintaan API Gateway untuk sebuah sumber daya dan metode dalam satu tahap memiliki latensi yang tinggi.

Statistik: p90

Ambang batas yang disarankan: 2500,0

Pembenaran ambang batas: Nilai ambang yang disarankan tidak akan berfungsi untuk semua beban kerja API. Namun demikian, Anda dapat menggunakannya sebagai titik awal untuk ambang batas tersebut. Anda kemudian dapat memilih nilai ambang batas yang berbeda berdasarkan beban kerja dan persyaratan latensi, performa, dan SLA yang dapat diterima untuk API. Jika API dapat diterima untuk memiliki latensi yang lebih tinggi secara umum, maka Anda dapat menetapkan nilai ambang batas yang lebih tinggi untuk membuat alarm tersebut menjadi tidak begitu sensitif. Namun demikian, jika API tersebut Anda harapkan dapat memberikan respons mendekati waktu nyata, maka Anda harus menetapkan nilai ambang batas yang lebih rendah. Anda juga dapat menganalisis data historis untuk menentukan latensi dasar yang diharapkan untuk beban kerja aplikasi dan kemudian menggunakannya untuk menyetel nilai ambang batas yang sesuai.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

Latensi

Dimensi:ApiId, Panggung

Deskripsi alarm: Alarm ini akan mendeteksi latensi tinggi dalam satu tahap. Temukan nilai metrik `IntegrationLatency` untuk memeriksa latensi API backend. Jika kedua metrik sebagian besar selaras, maka API backend akan menjadi sumber latensi yang lebih tinggi dan Anda harus menyelidiki masalah performa yang terjadi di sana. Pertimbangkan juga mengaktifkan CloudWatch Log dan memeriksa kesalahan apa pun yang mungkin menyebabkan latensi tinggi. Selain itu, pertimbangkan untuk mengaktifkan CloudWatch metrik terperinci untuk melihat metrik ini per rute dan mempersempit sumber latensi. Anda juga dapat melihat panduan pemecahan masalah dengan integrasi Lambda, jika perlu.

Maksud: Alarm ini dapat mendeteksi kapan permintaan API Gateway dalam satu tahap memiliki latensi yang tinggi. Jika CloudWatch metrik mendetail telah diaktifkan dan Anda memiliki persyaratan kinerja latensi yang berbeda per rute, kami sarankan Anda membuat alarm alternatif agar pemantauan latensi yang lebih halus untuk setiap rute.

Statistik: p90

Ambang batas yang disarankan: 2500,0

Pembenaran ambang batas: Nilai ambang yang disarankan tidak akan berfungsi untuk semua beban kerja API. Namun demikian, ia dapat Anda gunakan sebagai titik awal untuk ambang batas tersebut. Anda kemudian dapat memilih nilai ambang batas yang berbeda berdasarkan beban kerja dan persyaratan latensi, performa dan SLA yang dapat diterima untuk API. Jika API tersebut dapat diterima untuk memiliki latensi yang lebih tinggi secara umum, maka Anda dapat menetapkan nilai ambang batas yang lebih tinggi agar API menjadi tidak begitu sensitif. Namun demikian, jika API diharapkan untuk memberikan respons mendekati waktu nyata, maka Anda harus menetapkan nilai ambang batas yang lebih rendah. Anda juga dapat menganalisis data historis untuk menentukan latensi dasar yang diharapkan untuk beban kerja aplikasi dan kemudian menggunakannya untuk menyetel nilai ambang batas yang sesuai.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

Latensi

Dimensi:ApiId, Panggung, Sumber Daya, Metode

Deskripsi alarm: Alarm ini dapat mendeteksi latensi tinggi untuk sebuah rute dalam satu tahap. Temukan nilai metrik `IntegrationLatency` untuk memeriksa latensi API backend. Jika kedua metrik sebagian besar selaras, maka API backend akan menjadi sumber latensi yang lebih tinggi dan Anda harus menyelidiki masalah performa yang terjadi. Pertimbangkan juga mengaktifkan CloudWatch log dan memeriksa kesalahan apa pun yang mungkin menyebabkan latensi tinggi. Anda juga dapat melihat panduan pemecahan masalah dengan integrasi Lambda, jika perlu.

Maksud: Alarm ini dapat digunakan untuk mendeteksi kapan permintaan API Gateway untuk sebuah rute dalam satu tahap memiliki latensi yang tinggi.

Statistik: p90

Ambang batas yang disarankan: 2500,0

Pembenaran ambang batas: Nilai ambang yang disarankan tidak akan berfungsi untuk semua beban kerja API. Namun demikian, ia dapat Anda gunakan sebagai titik awal untuk ambang batas tersebut. Anda kemudian dapat memilih nilai ambang batas yang berbeda berdasarkan beban kerja dan persyaratan latensi, performa, dan SLA yang dapat diterima untuk API. Jika API dapat diterima untuk memiliki latensi yang lebih tinggi secara umum, maka Anda dapat menetapkan nilai ambang batas yang lebih tinggi untuk membuat alarm tersebut menjadi tidak begitu sensitif. Namun demikian, jika API tersebut Anda harapkan dapat memberikan respons mendekati waktu nyata, maka Anda harus menetapkan nilai ambang batas yang lebih rendah. Anda juga dapat menganalisis data historis untuk menentukan latensi dasar yang diharapkan untuk beban kerja aplikasi dan kemudian menggunakannya untuk menyetel nilai ambang batas yang sesuai.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

4xx

Dimensi:ApiId, Panggung

Deskripsi alarm: Alarm ini mendeteksi tingkat kesalahan sisi klien yang tinggi. Hal ini dapat menunjukkan adanya masalah dalam parameter otorisasi atau permintaan klien. Hal ini juga bisa berarti bahwa sebuah rute telah dihapus atau klien meminta sesuatu yang tidak ada dalam API. Pertimbangkan untuk mengaktifkan CloudWatch Log dan memeriksa kesalahan apa pun yang mungkin menyebabkan kesalahan 4xx. Selain itu, pertimbangkan untuk mengaktifkan CloudWatch metrik terperinci untuk melihat metrik ini per rute, untuk membantu Anda mempersempit sumber kesalahan. Kesalahan juga bisa disebabkan karena dilanggarnya batas throttling yang telah dikonfigurasi sebelumnya. Jika respons dan log melaporkan tingkat kesalahan sebanyak 429, jumlah yang tinggi dan tidak terduga, maka Anda harus mengikuti panduan ini untuk memecahkan masalah tersebut.

Maksud: Alarm ini dapat mendeteksi tingkat kesalahan sisi klien yang tinggi untuk permintaan API Gateway.

Statistik: Rata-rata

Ambang batas yang disarankan: 0,05

Pembenaran ambang batas: Ambang batas yang disarankan yang akan mendeteksi ketika lebih dari 5% dari total permintaan mendapatkan kesalahan 4xx. Namun demikian, Anda dapat menyetel ambang batas tersebut agar sesuai dengan lalu lintas permintaan serta tingkat kesalahan yang dapat diterima. Anda juga dapat menganalisis data historis untuk menentukan tingkat kesalahan yang dapat diterima untuk beban kerja aplikasi dan kemudian menyetel ambang batas yang sesuai. Kesalahan 4xx yang sering terjadi perlu Anda waspadai. Namun demikian, menetapkan nilai yang sangat rendah untuk ambang batas akan dapat menyebabkan alarm menjadi terlalu sensitif.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

5xx

Dimensi:ApiId, Panggung

Deskripsi alarm: Alarm ini akan membantu dalam mendeteksi tingkat kesalahan sisi server yang tinggi. Hal ini dapat menunjukkan bahwa ada sesuatu yang salah pada backend API, jaringan,

atau integrasi antara gateway API dan backend API. Dokumentasi ini dapat membantu Anda memecahkan masalah penyebab terjadinya kesalahan 5xx.

Maksud: Alarm ini dapat mendeteksi tingkat kesalahan sisi server yang tinggi untuk permintaan API Gateway.

Statistik: Rata-rata

Ambang batas yang disarankan: 0,05

Pembenaran ambang batas: Ambang batas yang disarankan yang akan mendeteksi ketika lebih dari 5% dari total permintaan mendapatkan kesalahan 5xx. Namun demikian, Anda dapat menyetel ambang batas tersebut agar sesuai dengan lalu lintas permintaan serta tingkat kesalahan yang dapat diterima. Anda juga dapat menganalisis data historis untuk menentukan tingkat kesalahan yang dapat diterima untuk beban kerja aplikasi, dan kemudian Anda dapat menyetel ambang batas sesuai dengan itu. Kesalahan 5xx yang sering terjadi perlu Anda waspadai. Namun demikian, menetapkan nilai yang sangat rendah untuk ambang batas akan dapat menyebabkan alarm menjadi terlalu sensitif.

Periode: 60

Titik data untuk alarm: 3

Periode evaluasi: 3

Operator Perbandingan: GREATER_THAN_THRESHOLD

MessageCount

Dimensi:ApiId, Panggung

Deskripsi alarm: Alarm ini membantu mendeteksi volume lalu lintas rendah untuk tahap WebSocket API. Hal ini dapat menunjukkan adanya masalah saat klien memanggil API, seperti penggunaan titik akhir yang salah, atau terjadinya masalah dengan backend yang mengirim pesan ke klien. Hal ini juga bisa menunjukkan adanya masalah dengan konfigurasi atau izin API, sehingga API tidak dapat dijangkau oleh klien.

Maksud: Alarm ini dapat mendeteksi volume lalu lintas rendah secara tak terduga untuk tahap API. WebSocket Kami menyarankan Anda untuk membuat alarm ini jika API Anda menerima dan mengirimkan jumlah pesan yang dapat diprediksi dan tetap konsisten dalam kondisi normal. Jika Anda mengaktifkan CloudWatch metrik terperinci dan Anda dapat memprediksi volume lalu

lintas normal per rute, lebih baik membuat alarm alternatif untuk yang satu ini, agar pemantauan penurunan volume lalu lintas yang lebih halus untuk setiap rute. Kami tidak merekomendasikan alarm ini untuk API yang diperkirakan tidak memiliki lalu lintas yang konstan dan konsisten.

Statistik: SampleCount

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Tetapkan nilai ambang batas berdasarkan analisis data historis untuk menentukan jumlah pesan dasar yang Anda perkirakan untuk API Anda. Menetapkan ambang batas dengan nilai yang sangat tinggi akan dapat menyebabkan alarm menjadi terlalu sensitif pada periode lalu lintas normal dan perkiraan lalu lintas rendah. Sebaliknya, menetapkan ambang batas dengan nilai yang sangat rendah akan dapat menyebabkan alarm tidak mendeteksi penurunan volume lalu lintas yang lebih kecil.

Periode: 60

Titik data untuk alarm: 10

Periode evaluasi: 10

Operator Perbandingan: LESS_THAN_THRESHOLD

MessageCount

Dimensi:ApiId, Panggung, Rute

Deskripsi alarm: Alarm ini membantu mendeteksi volume lalu lintas rendah untuk rute WebSocket API di panggung. Hal ini dapat menunjukkan adanya masalah pada klien yang memanggil API, seperti penggunaan titik akhir yang salah, atau terjadinya masalah dengan backend yang mengirim pesan ke klien. Hal ini juga bisa menunjukkan adanya masalah dengan konfigurasi atau izin API, sehingga API tidak dapat dijangkau oleh klien.

Maksud: Alarm ini dapat mendeteksi volume lalu lintas rendah secara tak terduga untuk rute WebSocket API di panggung. Kami menyarankan Anda untuk membuat alarm ini jika API Anda menerima dan mengirimkan jumlah pesan yang dapat diprediksi dan tetap konsisten dalam kondisi normal. Kami tidak merekomendasikan alarm ini untuk API yang diperkirakan tidak memiliki lalu lintas yang konstan dan konsisten.

Statistik: SampleCount

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Tetapkan ambang batas berdasarkan analisis data historis untuk menentukan jumlah pesan dasar yang Anda perkirakan untuk API Anda. Menetapkan ambang batas dengan nilai yang sangat tinggi akan dapat menyebabkan alarm menjadi terlalu sensitif pada periode lalu lintas normal dan perkiraan lalu lintas rendah. Sebaliknya, menetapkan ambang batas dengan nilai yang sangat rendah akan dapat menyebabkan alarm tidak mendeteksi penurunan volume lalu lintas yang lebih kecil.

Periode: 60

Titik data untuk alarm: 10

Periode evaluasi: 10

Operator Perbandingan: LESS_THAN_THRESHOLD

ClientError

Dimensi:ApiId, Panggung

Deskripsi alarm: Alarm ini akan mendeteksi tingkat kesalahan klien yang tinggi. Hal ini dapat menunjukkan adanya masalah dalam parameter otorisasi atau pesan. Hal ini juga bisa berarti bahwa sebuah rute telah dihapus atau klien meminta sesuatu yang tidak ada dalam API. Pertimbangkan untuk mengaktifkan CloudWatch Log dan memeriksa kesalahan apa pun yang mungkin menyebabkan kesalahan 4xx. Selain itu, pertimbangkan untuk mengaktifkan CloudWatch metrik terperinci untuk melihat metrik ini per rute, untuk membantu Anda mempersempit sumber kesalahan. Kesalahan juga dapat disebabkan karena dilanggarnya batas throttling yang telah dikonfigurasi sebelumnya. Jika respons dan log melaporkan tingkat kesalahan sebanyak 429, jumlah yang tinggi dan tidak terduga, maka Anda harus mengikuti panduan ini untuk memecahkan masalah tersebut.

Maksud: Alarm ini dapat mendeteksi tingkat kesalahan klien yang tinggi untuk pesan WebSocket API Gateway.

Statistik: Rata-rata

Ambang batas yang disarankan: 0,05

Pembenaran ambang batas: Ambang batas yang disarankan yang akan mendeteksi ketika lebih dari 5% dari total permintaan mendapatkan kesalahan 4xx. Anda dapat menyetel ambang batas tersebut agar sesuai dengan lalu lintas permintaan serta sesuai dengan tingkat kesalahan yang dapat diterima. Anda juga dapat menganalisis data historis untuk menentukan tingkat kesalahan yang dapat diterima untuk beban kerja aplikasi, dan kemudian Anda dapat menyetel ambang

batas yang sesuai. Kesalahan 4xx yang sering terjadi perlu Anda waspadai. Namun demikian, menetapkan nilai yang sangat rendah untuk ambang batas akan dapat menyebabkan alarm menjadi terlalu sensitif.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

ExecutionError

Dimensi:ApiId, Panggung

Deskripsi alarm: Alarm ini akan membantu Anda dalam mendeteksi tingkat kesalahan eksekusi yang tinggi. Hal ini dapat disebabkan oleh adanya kesalahan 5xx dari integrasi Anda, permasalahan izin, atau faktor lain yang mencegah berhasilnya invokasi atas integrasi, seperti integrasi yang dibatasi atau dihapus. Pertimbangkan untuk mengaktifkan CloudWatch Log untuk API Anda dan memeriksa log untuk jenis dan penyebab kesalahan. Selain itu, pertimbangkan untuk mengaktifkan CloudWatch metrik terperinci untuk mendapatkan tampilan metrik ini per rute, untuk membantu Anda mempersempit sumber kesalahan. Dokumentasi ini juga dapat membantu Anda dalam memecahkan masalah penyebab terjadinya kesalahan koneksi.

Maksud: Alarm ini dapat mendeteksi tingkat kesalahan eksekusi yang tinggi untuk pesan WebSocket API Gateway.

Statistik: Rata-rata

Ambang batas yang disarankan: 0,05

Pembenaran ambang batas: Ambang batas yang disarankan yang akan mendeteksi ketika lebih dari 5% dari total permintaan mendapatkan kesalahan eksekusi. Anda dapat menyetel ambang batas tersebut agar sesuai dengan lalu lintas permintaan, dan sesuai dengan tingkat kesalahan yang dapat diterima. Anda dapat menganalisis data historis untuk menentukan tingkat kesalahan yang dapat diterima untuk beban kerja aplikasi, dan kemudian Anda dapat menyetel ambang batas yang sesuai. Kesalahan eksekusi yang sering terjadi perlu Anda waspadai. Namun demikian, menetapkan nilai yang sangat rendah untuk ambang batas akan dapat menyebabkan alarm menjadi terlalu sensitif.

Periode: 60

Titik data untuk alarm: 3

Periode evaluasi: 3

Operator Perbandingan: GREATER_THAN_THRESHOLD

Amazon EC2 Auto Scaling

GroupInServiceCapacity

Dimensi: AutoScalingGroupName

Deskripsi alarm: Alarm ini akan membantu Anda dalam mendeteksi kapan kapasitas dalam grup berada di bawah kapasitas yang diinginkan yang diperlukan untuk beban kerja Anda. Untuk memecahkan masalah tersebut, Anda perlu memeriksa aktivitas penskalaan Anda, apakah ada kegagalan peluncuran yang terjadi, dan konfirmasikan apakah konfigurasi kapasitas yang Anda inginkan sudah dibuat dengan benar.

Maksud: Alarm ini dapat mendeteksi ketersediaan rendah di grup penskalaan otomatis (grup Auto Scaling) yang terjadi karena kegagalan peluncuran atau peluncuran yang ditangguhkan.

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Nilai ambang batas harus dijadikan sebagai kapasitas minimum yang diperlukan untuk menjalankan beban kerja Anda. Dalam kebanyakan kasus, Anda dapat mengatur ini agar sesuai dengan GroupDesiredCapacity metrik.

Periode: 60

Titik data untuk alarm: 10

Periode evaluasi: 10

Operator Perbandingan: LESS_THAN_THRESHOLD

Amazon CloudFront

5 xxErrorRate

Dimensi:DistributionId, Wilayah = Global

Deskripsi alarm: Alarm ini memantau persentase respons kesalahan 5xx dari server asal Anda, untuk membantu Anda mendeteksi apakah CloudFront layanan mengalami masalah. Lihat Memecahkan masalah respons kesalahan dari server asal Anda untuk mendapatkan informasi yang akan membantu Anda memahami masalah yang terjadi pada server Anda. Selain itu, aktifkan metrik tambahan jika Anda ingin mendapatkan metrik kesalahan terperinci.

Maksud: Alarm ini digunakan untuk mendeteksi masalah dengan melayani permintaan dari server asal, atau masalah dengan komunikasi antara CloudFront dan server asal Anda.

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Nilai ambang batas yang direkomendasikan untuk alarm ini sangat tergantung pada toleransi untuk respons 5xx. Anda dapat melakukan analisis data historis dan tren, dan kemudian Anda dapat menetapkan ambang batas yang sesuai. Karena kesalahan 5xx dapat disebabkan oleh masalah sementara, maka kami sarankan Anda untuk mengatur ambang batas ke nilai yang lebih besar dari 0 sehingga alarm tidak menjadi terlalu sensitif.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

OriginLatency

Dimensi:DistributionId, Wilayah = Global

Deskripsi alarm: Alarm ini dapat membantu Anda memantau apakah server asal terlalu lama untuk memberikan respons. Jika server membutuhkan waktu terlalu lama untuk memberi respons, hal itu mungkin akan menyebabkan habis waktu. Silakan lihat menemukan dan memperbaiki respons yang tertunda dari aplikasi di server asal Anda jika Anda terus-menerus mengalami nilai `OriginLatency` yang tinggi.

Maksud: Alarm ini dapat digunakan untuk mendeteksi masalah yang terjadi pada server asal yang terlalu lama dalam memberi respons.

Statistik: p90

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda harus menghitung nilainya sekitar 80% dari batas waktu respons asal, dan menggunakan hasilnya sebagai nilai ambang batas. Jika metrik ini terus-menerus mendekati nilai batas waktu respons asal, maka Anda mungkin mulai mengalami kesalahan 504.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

FunctionValidationErrors

Dimensi:DistributionId, FunctionName, Wilayah = Global

Deskripsi alarm: Alarm ini membantu Anda memantau kesalahan validasi dari CloudFront fungsi sehingga Anda dapat mengambil langkah-langkah untuk mengatasinya. Analisis log CloudWatch fungsi dan lihat kode fungsi untuk menemukan dan menyelesaikan akar penyebab masalah. Lihat Pembatasan fungsi tepi untuk memahami kesalahan konfigurasi umum untuk CloudFront Fungsi.

Maksud: Alarm ini digunakan untuk mendeteksi kesalahan validasi dari CloudFront fungsi.

Statistik: Jumlah

Ambang batas yang disarankan: 0,0

Pembenaran ambang batas: Nilai yang lebih besar dari 0 menunjukkan terjadinya kesalahan validasi. Kami merekomendasikan pengaturan ambang batas ke 0 karena kesalahan validasi menyiratkan masalah ketika CloudFront fungsi diserahkan kembali. CloudFront Misalnya, CloudFront membutuhkan header Host HTTP untuk memproses permintaan. Tidak ada yang menghentikan pengguna untuk menghapus header Host dalam kode CloudFront fungsinya. Tetapi ketika CloudFront mendapat respons kembali dan header Host hilang, CloudFront melempar kesalahan validasi.

Periode: 60

Titik data untuk alarm: 2

Periode evaluasi: 2

Operator Perbandingan: GREATER_THAN_THRESHOLD

FunctionExecutionErrors

Dimensi:DistributionId, FunctionName, Wilayah = Global

Deskripsi alarm: Alarm ini membantu Anda memantau kesalahan eksekusi dari CloudFront fungsi sehingga Anda dapat mengambil langkah-langkah untuk mengatasinya. Analisis log CloudWatch fungsi dan lihat kode fungsi untuk menemukan dan menyelesaikan akar penyebab masalah.

Maksud: Alarm ini digunakan untuk mendeteksi kesalahan eksekusi dari CloudFront fungsi.

Statistik: Jumlah

Ambang batas yang disarankan: 0,0

Pembenaran ambang batas: Kami merekomendasikan Anda untuk mengatur ambang batasnya menjadi 0 karena kesalahan eksekusi menunjukkan adanya masalah dengan kode yang terjadi saat runtime.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

FunctionThrottles

Dimensi:DistributionId, FunctionName, Wilayah = Global

Deskripsi alarm: Alarm ini membantu Anda memantau jika CloudFront fungsi Anda dibatasi. Jika fungsi Anda dibatasi, maka itu berarti fungsi tersebut terlalu lama untuk melakukan eksekusi. Untuk menghindari pembatasan fungsi, Anda perlu mempertimbangkan untuk mengoptimalkan kode fungsi.

Maksud: Alarm ini dapat mendeteksi ketika CloudFront fungsi Anda dibatasi sehingga Anda dapat bereaksi dan menyelesaikan masalah untuk pengalaman pelanggan yang lancar.

Statistik: Jumlah

Ambang batas yang disarankan: 0,0

Pembenaran ambang batas: Kami merekomendasikan Anda untuk mengatur ambang batasnya menjadi 0, hal ini untuk memungkinkan resolusi lebih cepat atas pembatasan fungsi.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

Amazon Cognito

SignUpThrottles

Dimensi:UserPool, UserPoolClient

Deskripsi alarm: Alarm ini dapat memantau jumlah permintaan yang dibatasi. Jika pengguna terus-menerus mengalami pembatasan, maka Anda harus meningkatkan batasnya dengan meminta peningkatan kuota layanan. Silakan lihat Kuota di Amazon Cognito untuk mempelajari cara meminta kenaikan kuota. Untuk melakukan tindakan-tindakan secara proaktif, Anda perlu mempertimbangkan untuk melacak kuota penggunaan.

Maksud: Alarm ini akan membantu Anda dalam memantau terjadinya permintaan pendaftaran yang dibatasi. Hal ini dapat membantu Anda mengetahui kapan harus melakukan tindakan-tindakan untuk mengurangi penurunan dalam pengalaman pendaftaran. Throttling permintaan yang terjadi terus-menerus adalah pengalaman pendaftaran pengguna yang negatif.

Statistik: Jumlah

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Kolam pengguna yang disediakan dengan baik seharusnya tidak mengalami throttling apa pun yang mencakup beberapa titik data. Jadi, ambang batas yang biasa digunakan untuk beban kerja yang diharapkan adalah nol. Untuk beban kerja yang tidak teratur dengan peningkatan beban tiba-tiba yang sering terjadi, Anda dapat melakukan analisis data historis untuk menentukan throttling yang dapat diterima untuk beban kerja aplikasi, dan kemudian Anda dapat menyetel ambang batas yang sesuai. Permintaan yang dibatasi harus dicoba ulang untuk meminimalkan dampaknya terhadap aplikasi.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

SignInThrottles

Dimensi:UserPool, UserPoolClient

Deskripsi alarm: Alarm ini dapat memantau jumlah permintaan autentikasi pengguna yang dibatasi. Jika pengguna terus-menerus mengalami pembatasan, maka Anda mungkin perlu meningkatkan batasnya dengan meminta peningkatan kuota layanan. Silakan lihat Kuota di Amazon Cognito untuk mempelajari cara meminta kenaikan kuota. Untuk melakukan tindakan-tindakan secara proaktif, Anda perlu mempertimbangkan untuk melacak kuota penggunaan.

Maksud: Alarm ini akan membantu Anda dalam memantau terjadinya permintaan masuk yang dibatasi. Hal ini dapat membantu Anda mengetahui kapan harus melakukan tindakan-tindakan untuk mengurangi penurunan dalam pengalaman masuk. Throttling permintaan yang terus-menerus terjadi akan menjadi pengalaman autentikasi pengguna yang buruk.

Statistik: Jumlah

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Kolam pengguna yang disediakan dengan baik seharusnya tidak mengalami throttling apa pun yang mencakup beberapa titik data. Jadi, ambang batas yang biasa digunakan untuk beban kerja yang diharapkan adalah nol. Untuk beban kerja yang tidak teratur dengan peningkatan beban tiba-tiba yang sering terjadi, Anda dapat melakukan analisis data historis untuk menentukan throttling yang dapat diterima untuk beban kerja aplikasi, dan kemudian Anda dapat menyetel ambang batas yang sesuai. Permintaan yang dibatasi harus dicoba ulang untuk meminimalkan dampaknya terhadap aplikasi.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

TokenRefreshThrottles

Dimensi:UserPool, UserPoolClient

Deskripsi alarm: Anda dapat mengatur nilai ambang batasnya agar sesuai dengan lalu lintas permintaan serta agar sesuai dengan throttling yang dapat diterima untuk permintaan penyegaran token. Throttling digunakan untuk melindungi sistem Anda dari terlalu banyaknya permintaan yang dikirimkan. Namun demikian, Anda juga harus memantau apakah Anda berada di bawah lalu lintas normal yang ditentukan. Anda dapat melakukan analisis data historis untuk menemukan throttling yang dapat diterima untuk beban kerja aplikasi, dan kemudian Anda dapat menyetel ambang batas alarm Anda menjadi lebih tinggi dari tingkat throttling yang dapat diterima. Permintaan-permintaan yang dibatasi harus dicoba ulang oleh aplikasi/layanan karena pembatasan bersifat sementara. Oleh karena itu, menetapkan nilai ambang batas yang sangat rendah dapat menyebabkan alarm menjadi sensitif.

Maksud: Alarm ini akan membantu Anda dalam memantau terjadinya permintaan penyegaran token yang dibatasi. Hal ini dapat membantu Anda dalam mengetahui kapan harus melakukan tindakan untuk mengurangi potensi masalah, dan untuk memastikan pengalaman pengguna yang lancar serta kesehatan dan keandalan sistem autentikasi Anda. Throttling permintaan yang terus-menerus terjadi akan menjadi pengalaman autentikasi pengguna yang buruk.

Statistik: Jumlah

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Nilai ambang batasnya juga dapat Anda atur agar sesuai dengan lalu lintas permintaan serta throttling yang dapat diterima untuk permintaan penyegaran token. Throttling ada untuk melindungi sistem Anda dari terlalu banyaknya permintaan yang dikirim, namun demikian, Anda juga harus memantau apakah Anda berada di bawah lalu lintas normal yang ditentukan dan memeriksa apakah hal itu yang menyebabkan dampak tersebut. Data historis juga dapat Anda analisis untuk melihat tingkat throttling yang dapat diterima untuk beban kerja aplikasi dan Anda kemudian dapat mengatur ambang batasnya lebih tinggi dari tingkat throttling yang dapat diterima yang biasa Anda gunakan. Permintaan-permintaan yang dibatasi harus dicoba ulang oleh aplikasi/layanan karena pembatasan bersifat sementara. Oleh karena itu, menetapkan nilai ambang batas yang sangat rendah dapat menyebabkan alarm menjadi sensitif.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

FederationThrottles

Dimensi:UserPool, UserPoolClient, IdentityProvider

Deskripsi alarm: Alarm ini dapat memantau jumlah permintaan federasi identitas yang dibatasi. Jika Anda terus-menerus melihat terjadinya throttling, hal itu mungkin menunjukkan bahwa Anda perlu meningkatkan batas dengan meminta peningkatan kuota layanan. Silakan lihat Kuota di Amazon Cognito untuk mempelajari cara meminta kenaikan kuota.

Maksud: Alarm ini akan membantu Anda dalam memantau terjadinya permintaan federasi identitas yang dibatasi. Hal ini dapat membantu Anda melakukan respons proaktif terhadap kemacetan performa atau kesalahan konfigurasi, dan memastikan pengalaman autentikasi yang lancar bagi pengguna Anda. Throttling permintaan yang terus-menerus terjadi akan menjadi pengalaman autentikasi pengguna yang buruk.

Statistik: Jumlah

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda dapat mengatur ambang batasnya agar sesuai dengan lalu lintas permintaan dan untuk menyesuaikannya dengan throttling yang dapat diterima untuk permintaan federasi identitas. Throttling digunakan untuk melindungi sistem Anda dari terlalu banyaknya permintaan yang dikirim. Namun demikian, Anda juga harus memantau apakah Anda berada di bawah lalu lintas normal yang ditentukan. Anda dapat melakukan analisis data historis untuk menemukan tingkat throttling yang dapat diterima untuk beban kerja aplikasi, dan Anda kemudian dapat menetapkan ambang batasnya dengan nilai yang lebih tinggi dari tingkat throttling yang dapat diterima. Permintaan-permintaan yang dibatasi harus dicoba ulang oleh aplikasi/layanan karena pembatasan bersifat sementara. Oleh karena itu, menetapkan nilai ambang batas yang sangat rendah dapat menyebabkan alarm menjadi sensitif.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

Amazon DynamoDB

AccountProvisionedReadCapacityUtilization

Dimensi: Tidak ada

Deskripsi alarm: Alarm ini akan mendeteksi apakah kapasitas baca akun telah mencapai batas yang ditentukan. Jika hal ini terjadi, maka Anda dapat menaikkan kuota akun untuk pemanfaatan kapasitas baca. Anda dapat melihat kuota untuk unit kapasitas baca Anda saat ini dan meminta peningkatan dengan menggunakan Kuota Layanan.

Maksud: Alarm ini dapat mendeteksi apakah pemanfaatan kapasitas baca akun sudah mendekati pemanfaatan kapasitas baca yang disediakan. Jika pemanfaatan tersebut telah mencapai batas maksimumnya, maka DynamoDB akan mulai membatasi permintaan baca.

Statistik: Maksimum

Ambang batas yang disarankan: 80,0

Pembenaran ambang batas: Tetapkan ambang batas menjadi 80%, sehingga tindakan (seperti menaikkan batas akun) dapat dilakukan sebelum mencapai kapasitas penuh untuk menghindari terjadinya throttling.

Periode: 300

Titik data untuk alarm: 2

Periode evaluasi: 2

Operator Perbandingan: GREATER_THAN_THRESHOLD

AccountProvisionedWriteCapacityUtilization

Dimensi: Tidak ada

Deskripsi alarm: Alarm ini akan mendeteksi apakah kapasitas tulis akun telah mencapai batas yang ditentukan. Jika hal ini terjadi, maka Anda dapat menaikkan kuota akun untuk pemanfaatan kapasitas tulis. Anda dapat melihat kuota untuk unit kapasitas tulis Anda saat ini dan meminta peningkatan dengan menggunakan Kuota Layanan.

Maksud: Alarm ini dapat mendeteksi apakah pemanfaatan kapasitas tulis akun sudah mendekati pemanfaatan kapasitas tulis yang disediakan. Jika pemanfaatan tersebut telah mencapai batas maksimumnya, maka DynamoDB akan mulai membatasi permintaan tulis.

Statistik: Maksimum

Ambang batas yang disarankan: 80,0

Pembenaran ambang batas: Tetapkan ambang batas menjadi 80%, sehingga tindakan (seperti menaikkan batas akun) dapat dilakukan sebelum mencapai kapasitas penuh yang ditentukan untuk menghindari terjadinya throttling.

Periode: 300

Titik data untuk alarm: 2

Periode evaluasi: 2

Operator Perbandingan: GREATER_THAN_THRESHOLD

AgeOfOldestUnreplicatedRecord

Dimensi:TableName, DelegatedOperation

Deskripsi alarm: Alarm ini dapat mendeteksi penundaan pada replikasi ke aliran data Kinesis. Dalam operasi normal, `AgeOfOldestUnreplicatedRecord` seharusnya hanya memerlukan waktu milidetik. Jumlah ini akan semakin besar sesuai dengan upaya replikasi yang gagal yang disebabkan oleh pilihan konfigurasi yang dikendalikan pelanggan. Contoh konfigurasi yang dikendalikan oleh pelanggan yang menyebabkan upaya replikasi yang gagal adalah kapasitas aliran data Kinesis yang penyediaannya kurang yang menyebabkan terjadinya throttling yang berlebihan. atau pembaruan manual dapat dilakukan pada kebijakan akses aliran data Kinesis yang mencegah DynamoDB menambahkan data ke aliran data. Untuk menjaga metrik ini tetap berada pada tingkat serendah mungkin, Anda perlu memastikan bahwa penyediaan kapasitas aliran data Kinesis yang tepat telah dilakukan dan memastikan bahwa izin DynamoDB tidak berubah.

Maksud: Alarm ini dapat memantau upaya replikasi yang gagal dan penundaan replikasi yang diakibatkan terhadap aliran data Kinesis.

Statistik: Maksimum

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda harus menetapkan ambang batas sesuai dengan penundaan replikasi yang diinginkan yang diukur dalam milidetik. Nilai ini tergantung pada persyaratan beban kerja Anda dan performa yang Anda harapkan.

Periode: 300

Titik data untuk alarm: 3

Periode evaluasi: 3

Operator Perbandingan: GREATER_THAN_THRESHOLD

FailedToReplicateRecordCount

Dimensi:TableName, DelegatedOperation

Deskripsi alarm: Alarm ini dapat mendeteksi jumlah catatan yang gagal direplikasi oleh DynamoDB ke aliran data Kinesis Anda. Item tertentu yang ukurannya lebih besar dari 34 KB mungkin akan diperluas ukurannya untuk mengubah catatan data yang lebih besar dari batas ukuran item 1 MB dari Aliran Data Kinesis. Perluasan ukuran ini terjadi ketika item yang lebih besar dari 34 KB ini menyertakan sejumlah besar nilai atribut Boolean atau kosong. Nilai atribut Boolean dan kosong disimpan sebagai 1 byte di DynamoDB, namun diperluas hingga 5 byte saat diserialkan menggunakan JSON standar untuk replikasi Kinesis Data Streams. DynamoDB tidak dapat mereplikasi catatan perubahan tersebut ke aliran data Kinesis Anda. DynamoDB akan melewatkan catatan data perubahan ini, dan secara otomatis akan tetap melakukan replikasi atas catatan-catatan berikutnya.

Maksud: Alarm ini dapat memantau jumlah catatan yang gagal direplikasi oleh DynamoDB ke aliran data Kinesis Anda karena batas ukuran item Aliran Data Kinesis.

Statistik: Jumlah

Ambang batas yang disarankan: 0,0

Pembenaran ambang batas: Anda harus menetapkan ambang batasnya menjadi 0 untuk mendeteksi catatan apa pun yang gagal direplikasi oleh DynamoDB.

Periode: 60

Titik data untuk alarm: 1

Periode evaluasi: 1

Operator Perbandingan: GREATER_THAN_THRESHOLD

ReadThrottleEvents

Dimensi: TableName

Deskripsi alarm: Alarm ini dapat mendeteksi apakah ada banyak permintaan baca yang dibatasi untuk tabel DynamoDB. Untuk memecahkan masalah ini, silakan lihat Memecahkan masalah throttling di Amazon DynamoDB.

Maksud: Alarm ini dapat mendeteksi throttling yang berkelanjutan untuk permintaan baca ke tabel DynamoDB. Throttling permintaan baca yang berkelanjutan dapat berdampak negatif pada operasi pembacaan beban kerja Anda dan akan mengurangi efisiensi sistem secara keseluruhan.

Statistik: Jumlah

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda harus menetapkan ambang batas sesuai dengan lalu lintas baca yang diharapkan untuk tabel DynamoDB, dan memperhitungkan tingkat throttling yang dapat diterima. Penting untuk memantau apakah Anda sedang mengalami kurang penyediaan dan bahwa hal itu tidak menyebabkan throttling yang terjadi terus-menerus Anda juga dapat melakukan analisis data historis untuk menemukan tingkat throttling yang dapat diterima untuk beban kerja aplikasi, dan kemudian Anda dapat menyetel ambang batasnya ke tingkat yang lebih tinggi dari level throttling yang biasa Anda gunakan. Permintaan-permintaan yang dibatasi harus dicoba ulang oleh aplikasi atau layanan karena pembatasan bersifat sementara. Oleh karena itu, ambang batas yang ditetapkan sangat rendah akan dapat menyebabkan alarm menjadi terlalu sensitif, dan menyebabkan perubahan status yang tidak diinginkan.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

ReadThrottleEvents

Dimensi:TableName, GlobalSecondaryIndexName

Deskripsi alarm: Alarm ini dapat mendeteksi apakah ada banyak jumlah permintaan baca yang dibatasi untuk Indeks Sekunder Global dari tabel DynamoDB. Untuk memecahkan masalah ini, silakan lihat Memecahkan masalah throttling di Amazon DynamoDB.

Maksud: Alarm ini dapat mendeteksi adanya throttling yang terjadi secara berkelanjutan untuk permintaan baca untuk Indeks Sekunder Global dari Tabel DynamoDB. Throttling permintaan

baca yang berkelanjutan dapat berdampak negatif pada operasi pembacaan beban kerja Anda dan akan mengurangi efisiensi sistem secara keseluruhan.

Statistik: Jumlah

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda harus menetapkan ambang batas sesuai dengan lalu lintas baca yang diharapkan untuk tabel DynamoDB, dan memperhitungkan tingkat throttling yang dapat diterima. Penting untuk memantau apakah Anda sedang mengalami kurang penyediaan dan bahwa hal itu tidak menyebabkan throttling yang terjadi terus-menerus. Anda juga dapat melakukan analisis data historis untuk menemukan tingkat throttling yang dapat diterima untuk beban kerja aplikasi, dan kemudian Anda dapat menyetel ambang batasnya ke tingkat yang lebih tinggi dari level throttling yang dapat diterima yang biasa Anda gunakan. Permintaan-permintaan yang dibatasi harus dicoba ulang oleh aplikasi atau layanan karena pembatasan bersifat sementara. Oleh karena itu, ambang batas yang ditetapkan sangat rendah akan dapat menyebabkan alarm menjadi terlalu sensitif, dan menyebabkan perubahan status yang tidak diinginkan.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

ReplicationLatency

Dimensi:TableName, ReceivingRegion

Deskripsi alarm: Alarm ini dapat mendeteksi apakah replika dalam sebuah Wilayah untuk tabel global tertinggal di belakang Wilayah sumber. Latensi dapat meningkat jika suatu AWS Wilayah terdegradasi dan Anda memiliki tabel replika di Wilayah tersebut. Dalam hal ini, Anda dapat mengalihkan sementara aktivitas baca dan tulis aplikasi Anda ke AWS Wilayah yang berbeda. Jika Anda menggunakan tabel global 2017.11.29 (Legacy) dari tabel global, maka Anda harus memverifikasi apakah unit kapasitas tulis (WCU) tersebut identik untuk masing-masing tabel replika. Anda juga dapat memastikan untuk mengikuti rekomendasi-rekomendasi yang diberikan dalam Praktik terbaik dan persyaratan untuk mengelola kapasitas.

Maksud: Alarm ini dapat mendeteksi apakah tabel replika di sebuah Wilayah tertinggal untuk mereplikasi perubahan yang datang dari Wilayah lain. Hal ini dapat menyebabkan replika Anda

menyimpang dari replika-replika lainnya. Sangat berguna untuk mengetahui latensi replikasi setiap AWS Wilayah dan memperingatkan jika latensi replikasi itu terus meningkat. Replikasi tabel hanya berlaku untuk tabel global.

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Nilai ambang batas yang disarankan untuk alarm ini sangat tergantung pada kasus penggunaan Anda. Latensi replikasi dengan durasi lebih dari 3 menit biasanya akan menjadi penyebab dimulainya penyelidikan. Lakukan peninjauan kekritisan dan persyaratan penundaan replikasi dan lakukan analisis tren historis, dan kemudian pilih ambang batas yang sesuai.

Periode: 60

Titik data untuk alarm: 15

Periode evaluasi: 15

Operator Perbandingan: GREATER_THAN_THRESHOLD

SuccessfulRequestLatency

Dimensi:TableName, Operasi

Deskripsi alarm: Alarm ini dapat mendeteksi latensi tinggi untuk operasi tabel DynamoDB (ditunjukkan oleh nilai dimensi dari `Operation` pada alarm tersebut). Silakan lihat dokumen pemecahan masalah ini untuk mengatasi terjadinya masalah latensi di Amazon DynamoDB.

Maksud: Alarm ini akan dapat mendeteksi latensi tinggi untuk operasi tabel DynamoDB. Latensi yang lebih tinggi untuk operasi akan dapat berdampak negatif pada efisiensi sistem secara keseluruhan.

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: DynamoDB menyediakan latensi milidetik satu digit rata-rata untuk operasi tunggal seperti,, dan sebagainya. GetItem PutItem Namun demikian, Anda dapat mengatur ambang batasnya berdasarkan toleransi yang dapat diterima untuk latensi untuk jenis operasi dan tabel yang digunakan dalam beban kerja. Anda dapat melakukan analisis data historis dari metrik ini untuk menemukan latensi yang biasanya terjadi untuk operasi tabel, dan

kemudian Anda dapat mengatur ambang batasnya ke angka yang mewakili penundaan kritis untuk operasi tersebut.

Periode: 60

Titik data untuk alarm: 10

Periode evaluasi: 10

Operator Perbandingan: GREATER_THAN_THRESHOLD

SystemErrors

Dimensi: TableName

Deskripsi alarm: Alarm ini dapat mendeteksi sejumlah besar kesalahan sistem yang terjadi terus-menerus untuk permintaan tabel DynamoDB. Jika Anda masih tetap mendapatkan kesalahan 5xx, silakan buka AWS Service Health Dashboard untuk memeriksa apakah terjadi masalah operasional dengan layanan. Anda dapat menggunakan alarm ini untuk mendapatkan pemberitahuan jika ada masalah yang terjadi terhadap layanan internal yang berkepanjangan atas DynamoDB dan hal ini akan membantu Anda mengorelasikan diri Anda dengan masalah yang dihadapi oleh aplikasi klien Anda. Silakan lihat Penanganan kesalahan untuk DynamoDB, untuk mendapatkan informasi lebih lanjut mengenai hal ini.

Maksud: Alarm ini dapat mendeteksi terjadinya kesalahan sistem yang berkelanjutan untuk permintaan tabel DynamoDB. Kesalahan sistem ini menunjukkan adanya kesalahan dalam layanan internal dari DynamoDB dan akan membantu Anda dalam mengorelasikan diri Anda dengan masalah yang dialami oleh klien Anda.

Statistik: Jumlah

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda harus menetapkan ambang batas sesuai dengan lalu lintas yang Anda harapkan, yang memperhitungkan tingkat kesalahan sistem yang dapat diterima. Anda juga dapat melakukan analisis data historis untuk menemukan jumlah kesalahan yang dapat diterima untuk beban kerja aplikasi, dan kemudian Anda dapat menyetel ambang batas yang sesuai. Kesalahan-kesalahan sistem harus dicoba ulang oleh aplikasi/layanan karena kesalahan tersebut bersifat sementara. Oleh karena itu, ambang batas yang ditetapkan sangat rendah akan dapat membuat alarm menjadi terlalu sensitif, dan menyebabkan perubahan status yang tidak diinginkan.

Periode: 60

Titik data untuk alarm: 15

Periode evaluasi: 15

Operator Perbandingan: GREATER_THAN_THRESHOLD

ThrottledPutRecordCount

Dimensi:TableName, DelegatedOperation

Deskripsi alarm: Alarm ini dapat mendeteksi catatan-catatan yang terhambat oleh aliran data Kinesis Anda selama dilakukan replikasi pengambilan data perubahan ke Kinesis. Throttling ini disebabkan oleh kapasitas aliran data Kinesis yang tidak mencukupi. Jika Anda mengalami throttling yang berlebihan dan secara rutin, maka Anda mungkin perlu meningkatkan jumlah serpihan aliran Kinesis secara proporsional dengan throughput tulis yang diamati pada tabel Anda. Untuk mempelajari selengkapnya tentang menentukan ukuran aliran data Kinesis, silakan lihat Menentukan Ukuran Awal Aliran Data Kinesis.

Maksud: Alarm ini dapat memantau jumlah data yang dibatasi oleh aliran data Kinesis karena kapasitas aliran data Kinesis tidak mencukupi.

Statistik: Maksimum

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda mungkin mengalami beberapa throttling selama puncak penggunaan yang luar biasa, tetapi catatan-catatan yang dibatasi harus tetap berada pada tingkat serendah mungkin untuk menghindari latensi replikasi yang lebih tinggi (DynamoDB akan mencoba lagi mengirimkan catatan-catatan yang dibatasi ke aliran data Kinesis). Anda harus mengatur ambang batasnya ke angka yang dapat membantu Anda dalam mengatasi throttling berlebihan yang terjadi secara rutin. Anda juga dapat melakukan analisis data historis mengenai metrik ini untuk menemukan tingkat throttling yang dapat diterima untuk beban kerja aplikasi. Anda kemudian harus mengatur ambang batasnya ke nilai yang dapat ditoleransi oleh aplikasi Anda berdasarkan kasus penggunaan Anda.

Periode: 60

Titik data untuk alarm: 10

Periode evaluasi: 10

Operator Perbandingan: GREATER_THAN_THRESHOLD

UserErrors

Dimensi: Tidak ada

Deskripsi alarm: Alarm ini dapat mendeteksi sejumlah besar kesalahan pengguna yang terjadi terus-menerus untuk permintaan tabel DynamoDB. Anda dapat memeriksa log aplikasi klien selama kerangka waktu terjadinya masalah untuk memeriksa penyebab yang membuat permintaan menjadi tidak valid. Anda dapat memeriksa kode status HTTP 400 untuk melihat jenis kesalahan apa yang Anda alami dan kemudian melakukan tindakan yang semestinya. Anda mungkin juga harus memperbaiki logika aplikasi untuk membuat permintaan yang valid.

Maksud: Alarm ini dapat mendeteksi terjadinya kesalahan pengguna yang berkelanjutan untuk permintaan tabel DynamoDB. Terjadinya kesalahan pengguna untuk operasi yang diminta menandakan bahwa klien membuat permintaan yang tidak valid dan gagal.

Statistik: Jumlah

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda harus menetapkan ambang batas ke nol untuk mendeteksi kesalahan yang terjadi pada sisi klien. Atau Anda dapat mengaturnya ke nilai yang lebih tinggi jika Anda tidak ingin membuat alarm yang aktif karena jumlah kesalahan yang sangat rendah. Anda harus membuat keputusan berdasarkan kasus penggunaan dan lalu lintas Anda untuk permintaan Anda.

Periode: 60

Titik data untuk alarm: 10

Periode evaluasi: 10

Operator Perbandingan: GREATER_THAN_THRESHOLD

WriteThrottleEvents

Dimensi: TableName

Deskripsi alarm: Alarm ini dapat mendeteksi apakah ada banyak permintaan tulis yang dibatasi untuk tabel DynamoDB. Silakan lihat Memecahkan masalah throttling di Amazon DynamoDB, untuk memecahkan masalah ini.

Maksud: Alarm ini dapat mendeteksi throttling yang berkelanjutan untuk permintaan tulis ke tabel DynamoDB. Throttling permintaan tulis yang berkelanjutan dapat berdampak negatif pada operasi tulis beban kerja Anda dan akan mengurangi efisiensi sistem secara keseluruhan.

Statistik: Jumlah

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda harus menetapkan ambang batas sesuai dengan lalu lintas tulis yang diharapkan untuk tabel DynamoDB, dan memperhitungkan tingkat throttling yang dapat diterima. Penting untuk memantau apakah Anda sedang mengalami kurang penyediaan dan bahwa hal itu tidak menyebabkan throttling yang terjadi terus-menerus. Anda juga dapat melakukan analisis data historis untuk menemukan tingkat throttling yang dapat diterima untuk beban kerja aplikasi, dan kemudian Anda dapat menyetel ambang batasnya dengan tingkat yang lebih tinggi dari level throttling yang dapat diterima yang biasa Anda gunakan. Permintaan-permintaan yang dibatasi harus dicoba ulang oleh aplikasi/layanan karena pembatasan bersifat sementara. Oleh karena itu, ambang batas yang ditetapkan sangat rendah akan dapat membuat alarm menjadi terlalu sensitif, dan menyebabkan perubahan status yang tidak diinginkan.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

WriteThrottleEvents

Dimensi:TableName, GlobalSecondaryIndexName

Deskripsi alarm: Alarm ini dapat mendeteksi apakah ada banyak jumlah permintaan tulis yang dibatasi untuk Indeks Sekunder Global dari tabel DynamoDB. Silakan lihat Memecahkan masalah throttling di Amazon DynamoDB, untuk memecahkan masalah ini.

Maksud: Alarm ini dapat mendeteksi adanya throttling yang terjadi secara berkelanjutan untuk permintaan tulis untuk Indeks Sekunder Global dari Tabel DynamoDB. Throttling permintaan tulis yang berkelanjutan dapat berdampak negatif pada operasi tulis beban kerja Anda dan akan mengurangi efisiensi sistem secara keseluruhan.

Statistik: Jumlah

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda harus menetapkan ambang batas sesuai dengan lalu lintas Tulis yang diharapkan untuk tabel DynamoDB, dan memperhitungkan tingkat throttling yang dapat diterima. Penting untuk memantau apakah Anda sedang mengalami kurang penyediaan dan bahwa hal itu tidak menyebabkan throttling yang terjadi terus-menerus. Anda juga dapat melakukan analisis data historis untuk menemukan tingkat throttling yang dapat diterima untuk beban kerja aplikasi, dan kemudian Anda dapat menyetel ambang batasnya dengan tingkat yang lebih tinggi dari level throttling yang dapat diterima yang biasa digunakan. Permintaan-permintaan yang dibatasi harus dicoba ulang oleh aplikasi/layanan karena pembatasan bersifat sementara. Oleh karena itu, nilai yang ditetapkan sangat rendah akan dapat membuat alarm menjadi terlalu sensitif, dan menyebabkan perubahan status yang tidak diinginkan.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

Amazon EBS

VolumeStalledIOCheck

Dimensi:VolumeId, InstanceId

Deskripsi alarm: Alarm ini membantu Anda memantau kinerja IO volume Amazon EBS Anda. Pemeriksaan ini mendeteksi masalah mendasar dengan infrastruktur Amazon EBS, seperti masalah perangkat keras atau perangkat lunak pada subsistem penyimpanan yang mendasari volume Amazon EBS, masalah perangkat keras pada host fisik yang memengaruhi jangkauan volume Amazon EBS dari instans Amazon EC2 Anda, dan dapat mendeteksi masalah konektivitas antara instans dan volume Amazon EBS. Jika Pemeriksaan IO Terhenti gagal, Anda dapat menunggu AWS untuk menyelesaikan masalah, atau Anda dapat mengambil tindakan seperti mengganti volume yang terpengaruh atau menghentikan dan memulai ulang instance tempat volume terpasang. Dalam kebanyakan kasus, ketika metrik ini gagal, Amazon EBS akan secara otomatis mendiagnosis dan memulihkan volume Anda dalam beberapa menit.

Maksud: Alarm ini dapat mendeteksi status volume Amazon EBS Anda untuk menentukan kapan volume ini terganggu dan tidak dapat menyelesaikan operasi I/O.

Statistik: Maksimum

Ambang batas yang disarankan: 1,0

Pembenaran ambang batas: Ketika terjadi kegagalan pemeriksaan status, nilai metrik ini adalah 1. Ambang batasnya diatur sedemekian rupa sehingga setiap kali terjadi kegagalan pemeriksaan status, alarm akan berada dalam status ALARM.

Periode: 60

Titik data untuk alarm: 10

Periode evaluasi: 10

Operator Perbandingan: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

Amazon EC2

CPUUtilization

Dimensi: InstanceId

Deskripsi alarm: Alarm ini dapat membantu Anda memantau pemanfaatan CPU dari instans EC2. Tergantung pada aplikasinya, tingkat pemanfaatan yang tinggi mungkin akan normal secara terus-menerus. Tetapi jika performanya menurun, dan aplikasi tidak dibatasi oleh disk I/O, memori, atau sumber daya jaringan, maka CPU yang telah mencapai batas atasnya mungkin akan menunjukkan hambatan sumber daya atau masalah performa aplikasi. Pemanfaatan CPU yang tinggi mungkin menunjukkan bahwa diperlukan adanya peningkatan ke instans yang lebih intensif CPU. Jika Anda mengaktifkan pemantauan terperinci, maka Anda dapat mengubah periodenya menjadi 60 detik, bukan 300 detik. Untuk informasi selengkapnya, silakan lihat Mengaktifkan atau menonaktifkan pemantauan terperinci untuk instans Anda.

Maksud: Alarm ini dapat digunakan untuk mendeteksi pemanfaatan CPU yang tinggi.

Statistik: Rata-rata

Ambang batas yang disarankan: 80,0

Pembenaran ambang batas: Anda biasanya dapat mengatur ambang batas untuk pemanfaatan CPU hingga 70-80%. Namun demikian, Anda dapat menyesuaikan nilai ini berdasarkan tingkat performa dan karakteristik beban kerja yang dapat diterima. Untuk beberapa sistem, pemanfaatan

CPU yang tinggi yang terjadi secara terus-menerus mungkin adalah hal normal dan tidak menunjukkan adanya masalah, sementara untuk yang lain, hal itu mungkin akan menimbulkan kekhawatiran tersendiri. Anda perlu melakukan analisis data pemanfaatan CPU historis untuk mengidentifikasi penggunaannya, menemukan seberapa besar pemanfaatan CPU yang dapat diterima untuk sistem Anda, dan kemudian menetapkan ambang batas sesuai dengan itu.

Periode: 300

Titik data untuk alarm: 3

Periode evaluasi: 3

Operator Perbandingan: GREATER_THAN_THRESHOLD

StatusCheckFailed

Dimensi: InstanceId

Deskripsi alarm: Alarm ini akan membantu Anda memantau pemeriksaan status sistem dan pemeriksaan status instans. Jika salah satu jenis pemeriksaan status mengalami kegagalan, maka alarm ini seharusnya beralih menjadi ALARM.

Maksud: Alarm ini dapat digunakan untuk mendeteksi masalah-masalah mendasar yang terjadi terhadap instans, termasuk masalah kegagalan pemeriksaan status sistem dan kegagalan pemeriksaan status instans.

Statistik: Maksimum

Ambang batas yang disarankan: 1,0

Pembenaran ambang batas: Ketika terjadi kegagalan pemeriksaan status, nilai metrik ini adalah 1. Ambang batasnya diatur sedemekian rupa sehingga setiap kali terjadi kegagalan pemeriksaan status, alarm akan berada dalam status ALARM.

Periode: 300

Titik data untuk alarm: 2

Periode evaluasi: 2

Operator Perbandingan: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

StatusCheckFailed_melampirkan Debs

Dimensi: InstanceId

Deskripsi alarm: Alarm ini membantu Anda memantau apakah volume Amazon EBS yang dilampirkan ke instans dapat dijangkau dan dapat menyelesaikan operasi I/O. Pemeriksaan status ini mendeteksi masalah mendasar pada komputasi atau infrastruktur Amazon EBS seperti berikut ini:

- Masalah perangkat keras atau perangkat lunak pada subsistem penyimpanan yang mendasari volume Amazon EBS
- Masalah perangkat keras pada host fisik yang memengaruhi jangkauan volume Amazon EBS
- Masalah konektivitas antara instans dan volume Amazon EBS

Jika pemeriksaan status EBS terlampir gagal, Anda dapat menunggu Amazon menyelesaikan masalah, atau Anda dapat mengambil tindakan seperti mengganti volume yang terpengaruh atau menghentikan dan memulai ulang instance.

Maksud: Alarm ini digunakan untuk mendeteksi volume Amazon EBS yang tidak dapat dijangkau yang dilampirkan ke sebuah instans. Ini dapat menyebabkan kegagalan dalam operasi I/O.

Statistik: Maksimum

Ambang batas yang disarankan: 1,0

Pembenaran ambang batas: Ketika terjadi kegagalan pemeriksaan status, nilai metrik ini adalah 1. Ambang batasnya diatur sedemekian rupa sehingga setiap kali terjadi kegagalan pemeriksaan status, alarm akan berada dalam status ALARM.

Periode: 60

Titik data untuk alarm: 10

Periode evaluasi: 10

Operator Perbandingan: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

Amazon ElastiCache

CPUUtilization

Dimensi:CacheClusterId, CacheNodeId

Deskripsi alarm: Alarm ini membantu memantau pemanfaatan CPU untuk seluruh ElastiCache instance, termasuk proses mesin database dan proses lain yang berjalan pada instance. AWS

Elasticache mendukung dua jenis mesin: Memcached dan Redis. Ketika Anda mencapai pemanfaatan CPU yang tinggi pada sebuah simpul Memcached, Anda harus mempertimbangkan untuk meningkatkan tipe instans Anda atau menambahkan simpul cache yang baru. Untuk Redis, jika beban kerja utama Anda adalah dari permintaan baca, maka Anda harus mempertimbangkan untuk menambahkan lebih banyak replika baca pada klaster cache Anda. Jika beban kerja utama Anda berasal dari permintaan tulis, maka Anda harus mempertimbangkan untuk menambahkan lebih banyak serpihan untuk mendistribusikan beban kerja di lebih banyak simpul primer jika Anda berjalan dalam mode klaster, atau menaikkan skala tipe instans Anda jika Anda menjalankan Redis dalam mode non-klaster.

Maksud: Alarm ini digunakan untuk mendeteksi pemanfaatan CPU host yang ElastiCache tinggi. Hal ini berguna untuk mendapatkan pandangan yang luas dari penggunaan CPU di seluruh instans, termasuk proses-proses non-mesin.

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda harus menetapkan ambang batasnya dengan persentase yang mencerminkan tingkat pemanfaatan CPU kritis untuk aplikasi Anda. Untuk Memcached, mesinnya dapat menggunakan hingga inti num_threads. Untuk Redis, mesinnya sebagian besar berupa mesin single-threaded, tetapi mungkin akan menggunakan inti tambahan, jika ada, untuk mempercepat I/O. Dalam kebanyakan kasus, Anda dapat mengatur ambang batasnya hingga sekitar 90% dari CPU yang tersedia. Karena Redis bersifat single-threaded, nilai ambang batas yang sebenarnya harus dihitung sebagai bagian kecil dari seluruh kapasitas simpul ini.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

CurrConnections

Dimensi:CacheClusterId, CacheNodeId

Deskripsi alarm: Alarm ini dapat mendeteksi jumlah koneksi yang tinggi, yang mungkin mengindikasikan beban berat atau masalah-masalah performa. Peningkatan yang terjadi secara terus-menerus terhadap `CurrConnections` akan dapat menyebabkan terjadinya letih yang

berlebihan pada 65.000 koneksi yang tersedia. Hal ini mungkin menunjukkan bahwa koneksi-koneksi tersebut tidak ditutup dengan benar di sisi aplikasi dan dibiarkan dibuat di sisi server. Anda harus mempertimbangkan untuk menggunakan pengumpulan koneksi atau batas waktu koneksi idle untuk membatasi jumlah koneksi yang dibuat ke klaster, atau untuk Redis, Anda harus mempertimbangkan untuk menyetel tcp-keepalive di klaster Anda untuk mendeteksi dan menghentikan potensi rekan mati.

Maksud: Alarm membantu Anda mengidentifikasi jumlah koneksi tinggi yang dapat memengaruhi kinerja dan stabilitas klaster Anda ElastiCache .

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Nilai ambang batas yang disarankan untuk alarm ini sangat bergantung pada rentang koneksi yang dapat diterima untuk klaster Anda. Tinjau kapasitas dan beban kerja yang diharapkan dari ElastiCache klaster Anda dan analisis jumlah koneksi historis selama penggunaan reguler untuk menetapkan garis dasar, lalu pilih ambang batas yang sesuai. Ingatlah bahwa masing-masing simpul dapat mendukung hingga 65.000 koneksi secara bersamaan.

Periode: 60

Titik data untuk alarm: 10

Periode evaluasi: 10

Operator Perbandingan: GREATER_THAN_THRESHOLD

DatabaseMemoryUsagePercentage

Dimensi: CacheClusterId

Deskripsi alarm: Alarm ini akan membantu Anda dalam memantau pemanfaatan memori pada klaster Anda. Ketika `DatabaseMemoryUsagePercentage` Anda mencapai 100%, kebijakan memori maksimal Redis akan diaktifkan dan pengosongan mungkin akan dilakukan berdasarkan kebijakan yang dipilih. Jika tidak ada objek dalam cache yang cocok dengan kebijakan pengosongan, maka operasi penulisan akan digagalkan. Beberapa beban kerja mengharapkan atau mengandalkan pengosongan, tetapi jika tidak, Anda perlu meningkatkan kapasitas memori klaster Anda. Anda dapat meningkatkan skala klaster Anda dengan menambahkan lebih banyak simpul primer, atau menaikkan skalanya dengan menggunakan tipe simpul yang lebih besar. Lihat Scaling ElastiCache for Redis cluster untuk detailnya.

Maksud: Alarm ini dapat digunakan untuk mendeteksi pemanfaatan memori klaster yang tinggi sehingga Anda dapat mencegah terjadinya kegagalan saat menulis ke klaster Anda. Akan sangat berguna jika Anda mengetahui kapan Anda perlu menaikkan skala klaster Anda jika aplikasi Anda tidak diharapkan untuk mengalami pengosongan.

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Bergantung pada persyaratan memori aplikasi Anda dan kapasitas memori ElastiCache klaster Anda, Anda harus menetapkan ambang batas ke persentase yang mencerminkan tingkat kritis penggunaan memori klaster. Anda dapat menggunakan data penggunaan memori historis sebagai acuan untuk ambang batas penggunaan memori yang dapat diterima.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

EngineCPUUtilization

Dimensi: CacheClusterId

Deskripsi alarm: Alarm ini membantu memantau pemanfaatan CPU dari thread mesin Redis dalam instance. ElastiCache Alasan umum yang menyebabkan adanya CPU mesin tinggi adalah perintah yang berjalan lama yang mengkonsumsi CPU tinggi, jumlah permintaan yang tinggi, peningkatan permintaan koneksi klien baru dalam periode waktu singkat, dan pengosongan yang tinggi ketika cache tidak memiliki cukup memori untuk menyimpan data baru. Anda harus mempertimbangkan Penskalaan ElastiCache untuk klaster Redis dengan menambahkan lebih banyak node atau meningkatkan jenis instance Anda.

Maksud: Alarm ini dapat digunakan untuk mendeteksi pemanfaatan CPU yang tinggi dari thread mesin Redis. Hal ini akan berguna jika Anda ingin memantau penggunaan CPU dari mesin basis data itu sendiri.

Statistik: Rata-rata

Ambang batas yang disarankan: 90,0

Pembenaran ambang batas: Anda harus menetapkan ambang batasnya dengan persentase yang mencerminkan tingkat pemanfaatan CPU mesin kritis untuk aplikasi Anda. Anda dapat melakukan penolokukuran atas klaster Anda dengan menggunakan aplikasi dan beban kerja yang diharapkan untuk mengkorelasikan EngineCPUUtilization dan performa sebagai referensi, dan kemudian Anda harus menetapkan ambang batas yang sesuai. Dalam kebanyakan kasus, Anda dapat mengatur ambang batas tersebut hingga sekitar 90% dari CPU yang tersedia.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

ReplicationLag

Dimensi: CacheClusterId

Deskripsi alarm: Alarm ini membantu memantau kesehatan replikasi ElastiCache cluster Anda. Kelambatan replikasi yang tinggi menunjukkan bahwa simpul primer atau replikanya tidak dapat mengikuti laju replikasi. Jika aktivitas tulis Anda terlalu tinggi, maka Anda harus mempertimbangkan untuk menaikkan skala klaster Anda dengan menambahkan lebih banyak simpul primer, atau menaikkan skalanya dengan menggunakan tipe simpul yang lebih besar. Lihat Scaling ElastiCache for Redis cluster untuk detailnya. Jika replika baca Anda kelebihan beban karena jumlah permintaan baca yang terlalu besar, maka Anda harus mempertimbangkan untuk menambahkan lebih banyak replika baca.

Maksud: Alarm ini dapat digunakan untuk mendeteksi penundaan antara pembaruan data pada simpul primer dan sinkronisasinya dengan simpul replika. Hal ini akan membantu Anda memastikan konsistensi data dari simpul klaster replika baca.

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda harus menetapkan ambang batas sesuai dengan persyaratan aplikasi Anda dan dampak kelambatan replikasi yang mungkin terjadi. Anda juga harus mempertimbangkan tingkat tulis yang diharapkan dan kondisi jaringan aplikasi Anda untuk kelambatan replikasi yang dapat diterima.

Periode: 60

Titik data untuk alarm: 15

Periode evaluasi: 15

Operator Perbandingan: GREATER_THAN_THRESHOLD

Amazon EC2 (**`AWS/ElasticGPUs`**)

GPU ConnectivityCheckFailed

Dimensi:InstanceId, EGPUID

Deskripsi alarm: Alarm ini dapat membantu Anda mendeteksi kegagalan koneksi antara instans dan akselerator Elastic Graphics. Elastic Graphics menggunakan jaringan instans untuk mengirimkan perintah-perintah OpenGL ke kartu grafis yang terpasang secara jarak jauh. Selain itu, sebuah desktop yang menjalankan aplikasi OpenGL dengan akselerator Elastic Graphics biasanya dapat diakses dengan menggunakan teknologi akses jarak jauh. Penting bagi Anda untuk bisa membedakan antara masalah performa yang terkait dengan rendering OpenGL atau teknologi akses jarak jauh desktop. Untuk mempelajari lebih lanjut tentang masalah ini, silakan lihat Menyelidiki masalah performa aplikasi.

Maksud: Alarm ini dapat digunakan untuk mendeteksi masalah-masalah terkait dengan konektivitas dari instans ke akselerator Elastic Graphics.

Statistik: Maksimum

Ambang batas yang disarankan: 0,0

Pembenaran ambang batas: Nilai ambang batas 1 menunjukkan bahwa terjadi kegagalan konektivitas.

Periode: 300

Titik data untuk alarm: 3

Periode evaluasi: 3

Operator Perbandingan: GREATER_THAN_THRESHOLD

GPU HealthCheckFailed

Dimensi:InstanceId, EGPUID

Deskripsi alarm: Alarm ini dapat membantu Anda mengetahui saat status akselerator grafis Elastis dalam kondisi tidak sehat. Jika akselerator berada dalam kondisi tidak sehat, silakan lihat langkah-langkah pemecahan masalah di Menyelesaikan masalah status Tidak Sehat.

Maksud: Alarm ini digunakan untuk mendeteksi apakah akselerator Elastic Graphics sedang tidak sehat atau tidak.

Statistik: Maksimum

Ambang batas yang disarankan: 0,0

Pembenaran ambang batas: Nilai ambang batas 1 menunjukkan terjadinya kegagalan pemeriksaan status.

Periode: 300

Titik data untuk alarm: 3

Periode evaluasi: 3

Operator Perbandingan: GREATER_THAN_THRESHOLD

Amazon ECS

CPUReservation

Dimensi: ClusterName

Deskripsi alarm: Alarm ini dapat membantu Anda mendeteksi reservasi CPU yang tinggi pada klaster ECS. Reservasi CPU yang tinggi mungkin menunjukkan bahwa klaster telah mengalami kehabisan CPU terdaftar untuk tugas tersebut. Untuk memecahkan masalah itu, Anda dapat menambahkan kapasitas lebih banyak, Anda juga dapat menskalakan klaster, atau Anda dapat mengatur penskalaan otomatis.

Maksud: Alarm ini digunakan untuk mendeteksi apakah jumlah total unit CPU yang dicadangkan oleh tugas pada klaster mencapai total unit CPU yang terdaftar untuk klaster tersebut. Hal ini akan membantu Anda mengetahui kapan Anda seharusnya menaikkan skala klaster. Jika total unit CPU untuk klaster telah tercapai, maka hal itu dapat mengakibatkan tugas kehabisan CPU. Jika Anda mengaktifkan penskalaan terkelola penyedia kapasitas EC2, atau Anda telah mengaitkan Fargate untuk penyedia kapasitas, maka alarm ini tidak disarankan untuk Anda.

Statistik: Rata-rata

Ambang batas yang disarankan: 90,0

Pembenaran ambang batas: Anda harus menetapkan ambang batas untuk reservasi CPU hingga 90%. Atau, Anda bisa memilih nilai yang lebih rendah berdasarkan karakteristik klaster.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

CPUUtilization

Dimensi:ClusterName, ServiceName

Deskripsi alarm: Alarm ini akan membantu Anda mendeteksi pemanfaatan CPU yang tinggi pada layanan-layanan ECS. Jika tidak ada deployment ECS yang sedang berlangsung, maka pemanfaatan CPU yang maksimal mungkin menunjukkan adanya kemacetan sumber daya atau masalah performa pada aplikasi. Untuk mengatasi masalah ini, Anda dapat menaikkan batas CPU.

Maksud: Alarm ini dapat Anda gunakan untuk mendeteksi pemanfaatan CPU yang tinggi untuk layanan-layanan ECS. Pemanfaatan CPU tinggi yang terjadi secara terus-menerus dapat menunjukkan adanya kemacetan sumber daya atau masalah performa pada aplikasi.

Statistik: Rata-rata

Ambang batas yang disarankan: 90,0

Pembenaran ambang batas: Metrik layanan untuk pemanfaatan CPU boleh melebihi 100% pemanfaatan. Namun demikian, kami menyarankan Anda untuk memantau metrik untuk pemanfaatan CPU yang tinggi agar hal itu tidak memengaruhi layanan-layanan yang lain. Aturlah ambang batasnya menjadi sekitar 90-95%. Kami menyarankan Anda untuk memperbarui penentuan tugas untuk mencerminkan penggunaan aktual sehingga Anda bisa mencegah masalah-masalah yang mungkin terjadi di masa depan dengan layanan-layanan lainnya.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

MemoryReservation

Dimensi: ClusterName

Deskripsi alarm: Alarm ini dapat membantu Anda mendeteksi reservasi memori tinggi pada klaster ECS. Reservasi memori yang tinggi mungkin menunjukkan adanya kemacetan sumber daya untuk klaster tersebut. Untuk memecahkan masalah ini, Anda harus melakukan analisis tugas layanan untuk performa sehingga Anda bisa memeriksa apakah pemanfaatan memori tugas dapat dioptimalkan. Selain itu, Anda juga dapat mendaftarkan lebih banyak memori atau mengatur penskalaan otomatis.

Intent: Alarm ini digunakan untuk mendeteksi apakah total unit memori yang dicadangkan oleh tugas di klaster telah mencapai total unit memori yang terdaftar untuk klaster tersebut. Hal ini akan dapat membantu Anda mengetahui kapan Anda seharusnya menaikkan skala klaster. Ketika unit memori total untuk klaster tersebut telah tercapai, hal itu dapat menyebabkan klaster tidak dapat meluncurkan tugas baru. Jika Anda mengaktifkan penskalaan terkelola penyedia kapasitas EC2, atau Anda telah mengaitkan Fargate untuk penyedia kapasitas, maka alarm ini tidak disarankan bagi Anda.

Statistik: Rata-rata

Ambang batas yang disarankan: 90,0

Pembenaran ambang batas: Anda harus menetapkan ambang batas untuk reservasi memori menjadi 90%. Anda dapat menyesuaikan ambang batas ini dengan nilai yang lebih rendah berdasarkan karakteristik klaster.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

HTTPCode_Target_5XX_Count

Dimensi:ClusterName, ServiceName

Deskripsi alarm: Alarm ini dapat membantu Anda mendeteksi jumlah kesalahan sisi server yang tinggi untuk layanan ECS. Hal ini dapat menunjukkan bahwa telah terjadi kesalahan yang menyebabkan server tidak dapat melayani permintaan. Untuk memecahkan masalah seperti itu, Anda bisa memeriksa log aplikasi Anda.

Maksud: Alarm ini dapat digunakan untuk mendeteksi jumlah kesalahan sisi server yang tinggi untuk layanan ECS.

Statistik: Jumlah

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda harus menghitung nilai sekitar 5% dari lalu lintas rata-rata Anda dan kemudian menggunakannya sebagai titik awal untuk ambang batas. Anda dapat menemukan lalu lintas rata-rata dengan menggunakan metrik `RequestCount`. Anda juga dapat menganalisis data historis untuk menentukan tingkat kesalahan yang dapat diterima untuk beban kerja aplikasi, dan kemudian Anda dapat menyetel ambang batas yang sesuai. Kesalahan 5XX yang sering terjadi perlu Anda waspadai. Namun demikian, menetapkan nilai yang sangat rendah untuk ambang batas akan dapat menyebabkan alarm menjadi terlalu sensitif.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

TargetResponseTime

Dimensi:ClusterName, ServiceName

Deskripsi alarm: Alarm ini akan membantu Anda mendeteksi waktu respons target yang tinggi untuk permintaan-permintaan layanan ECS. Hal ini dapat menunjukkan bahwa telah terjadi masalah yang menyebabkan layanan tidak dapat melayani permintaan secara tepat waktu. Untuk memecahkan masalah tersebut, Anda perlu memeriksa metrik CPUUtilization untuk melihat apakah layanan kehabisan CPU, atau memeriksa pemanfaatan CPU dari layanan hilir lain yang bergantung pada layanan Anda.

Maksud: Alarm ini digunakan untuk mendeteksi waktu respons target yang tinggi untuk permintaan-permintaan layanan ECS.

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Nilai ambang batas yang disarankan untuk alarm ini sangat tergantung pada kasus penggunaan Anda. Anda perlu meninjau kekritisan dan persyaratan waktu respons target layanan dan kemudian melakukan analisis perilaku historis metrik ini untuk menentukan tingkat ambang batas yang masuk akal.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

Amazon ECS dengan Wawasan Kontainer

EphemeralStorageUtilized

Dimensi:ClusterName, ServiceName

Deskripsi alarm: Alarm ini akan membantu Anda dalam mendeteksi penyimpanan sementara tinggi yang digunakan dari klaster Fargate. Jika penyimpanan sementara secara konsisten tinggi, Anda dapat memeriksa penggunaan penyimpanan sementara dan meningkatkan penyimpanan sementara tersebut.

Maksud: Alarm ini digunakan untuk mendeteksi penggunaan penyimpanan sementara yang tinggi untuk klaster Fargate. Penyimpanan sementara yang tinggi dan konsisten yang digunakan dapat menunjukkan bahwa disk sudah penuh dan dapat menyebabkan terjadinya kegagalan terhadap kontainer.

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Tetapkan ambang batas menjadi sekitar 90% dari ukuran penyimpanan sementara. Anda dapat menyesuaikan nilai ini berdasarkan pemanfaatan penyimpanan sementara yang dapat diterima dari klaster Fargate. Untuk beberapa sistem, adanya penyimpanan sementara tinggi yang digunakan secara konsisten mungkin bisa jadi hal

normal, sedangkan untuk sistem yang lain, dapat menyebabkan terjadinya kegagalan terhadap kontainer.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

RunningTaskCount

Dimensi:ClusterName, ServiceName

Deskripsi alarm: Alarm ini akan membantu Anda dalam mendeteksi jumlah tugas yang berjalan rendah dari layanan ECS. Jika jumlah tugas yang berjalan terlalu rendah, hal ini dapat menunjukkan bahwa aplikasi tidak dapat menangani beban layanan dan hal tersebut dapat menyebabkan terjadinya masalah performa. Jika tidak ada tugas yang berjalan, layanan Amazon ECS mungkin tidak tersedia atau mungkin ada masalah yang terjadi dalam deployment.

Maksud: Alarm ini digunakan untuk mendeteksi apakah jumlah tugas yang berjalan terlalu rendah atau tidak. Jumlah tugas berjalan yang rendah yang terjadi secara konsisten dapat menunjukkan adanya masalah deployment layanan ECS atau masalah performa.

Statistik: Rata-rata

Ambang batas yang disarankan: 0,0

Pembenaran ambang batas: Anda dapat menyesuaikan ambang batas berdasarkan jumlah tugas minimum yang berjalan dari layanan ECS. Jika jumlah tugas yang berjalan adalah 0, artinya layanan Amazon ECS tidak akan tersedia.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: LESS_THAN_OR_EQUAL_TO_THRESHOLD

instance_filesystem_utilization

Dimensi:InstanceId, ContainerInstanceId, ClusterName

Deskripsi alarm: Alarm ini akan membantu Anda dalam mendeteksi pemanfaatan sistem file klaster ECS yang berada dalam level yang tinggi. Jika pemanfaatan sistem file secara konsisten berada dalam level yang tinggi, periksa penggunaan disk.

Maksud: Alarm ini digunakan untuk mendeteksi pemanfaatan sistem file yang tinggi untuk klaster Amazon ECS. Pemanfaatan sistem file yang berada dalam level yang tinggi secara konsisten dapat menunjukkan terjadinya kemacetan sumber daya atau masalah performa pada aplikasi, dan mungkin akan mencegah berjalannya tugas-tugas baru.

Statistik: Rata-rata

Ambang batas yang disarankan: 90,0

Pembenaran ambang batas: Anda dapat mengatur ambang batas untuk pemanfaatan sistem file hingga 90-95%. Anda dapat menyesuaikan nilai ini berdasarkan tingkat kapasitas sistem file klaster Amazon ECS yang dapat diterima. Untuk beberapa sistem, terjadinya pemanfaatan sistem file yang berada dalam level yang tinggi secara konsisten mungkin menjadi sesuatu yang normal dan tidak menunjukkan adanya masalah, sedangkan untuk sistem yang lain, hal itu mungkin menjadi penyebab munculnya kekhawatiran dan dapat menyebabkan terjadinya masalah performa dan mencegah berjalannya tugas-tugas baru.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

Amazon EFS

PercentIOLimit

Dimensi: FileSystemId

Deskripsi alarm: Alarm ini akan membantu Anda untuk memastikan bahwa beban kerja tetap berada dalam ambang batas I/O yang tersedia untuk sistem file. Jika metrik terus-menerus mencapai batas I/O, Anda perlu mempertimbangkan untuk memindahkan aplikasi ke sistem file yang menggunakan mode performa Max I/O. Untuk memecahkan masalah ini, Anda perlu memeriksa klien yang terhubung ke sistem file dan aplikasi klien yang membatasi sistem file.

Maksud: Alarm ini digunakan untuk mendeteksi seberapa dekat sistem file untuk mencapai batas I/O pada mode performa Tujuan Umum. Persentase I/O yang tinggi yang terjadi secara terus-menerus dapat menjadi indikator yang menunjukkan bahwa sistem file tidak dapat menskalakan sehubungan dengan permintaan I/O yang cukup dan bahwa sistem file dapat menjadi hambatan sumber daya untuk aplikasi yang menggunakan sistem file tersebut.

Statistik: Rata-rata

Ambang batas yang disarankan: 100,0

Pembenaran ambang batas: Ketika sistem file mencapai batas I/O, sistem file tersebut mungkin akan merespons permintaan baca dan tulis dengan lebih lambat. Oleh karena itu, Anda disarankan untuk memantau metrik ini sehingga Anda dapat mencegah dampak aplikasi yang menggunakan sistem file tersebut. Ambang batasnya dapat diatur sekitar 100%. Namun demikian, nilai ini dapat disesuaikan dengan nilai yang lebih rendah berdasarkan karakteristik sistem file.

Periode: 60

Titik data untuk alarm: 15

Periode evaluasi: 15

Operator Perbandingan: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

BurstCreditBalance

Dimensi: FileSystemId

Deskripsi alarm: Alarm ini akan membantu Anda memastikan bahwa ada lonjakan saldo kredit yang tersedia untuk penggunaan sistem file. Ketika tidak ada lonjakan kredit yang tersedia, maka akses aplikasi ke sistem file akan terbatas karena throughput yang rendah. Jika metrik terus-menerus turun menjadi 0, maka Anda harus mempertimbangkan untuk mengubah mode throughput ke mode throughput Elastic atau Provisioned.

Maksud: Alarm ini digunakan untuk mendeteksi lonjakan saldo kredit yang rendah pada sistem file. Lonjakan saldo kredit yang rendah yang terjadi secara terus-menerus dapat menjadi indikator yang menunjukkan terjadinya perlambatan throughput dan peningkatan latensi I/O.

Statistik: Rata-rata

Ambang batas yang disarankan: 0,0

Pembenaran ambang batas: Ketika sistem file kehabisan kredit burst dan bahkan jika tingkat throughput dasar lebih rendah, EFS terus memberikan throughput terukur 1 ke semua sistem file. MiBps Namun demikian, Anda disarankan untuk memantau lonjakan saldo kredit rendah pada metrik untuk mencegah sistem file bertindak menjadi hambatan sumber daya untuk aplikasi. Ambang batasnya dapat Anda atur di sekitar 0 byte.

Periode: 60

Titik data untuk alarm: 15

Periode evaluasi: 15

Operator Perbandingan: LESS_THAN_OR_EQUAL_TO_THRESHOLD

Amazon EKS dengan Wawasan Kontainer

node_cpu_utilization

Dimensi: ClusterName

Deskripsi alarm: Alarm ini akan membantu Anda mendeteksi pemanfaatan CPU yang tinggi pada simpul pekerja klaster EKS. Jika pemanfaatannya secara terus-menerus berada dalam level yang tinggi, hal ini mungkin menunjukkan bahwa Anda perlu mengganti simpul pekerja Anda dengan instans yang memiliki CPU lebih besar atau Anda perlu menskalakan sistem secara horizontal.

Maksud: Alarm ini akan membantu Anda memantau pemanfaatan CPU dari simpul pekerja di klaster EKS sehingga penurunan performa sistem tidak terjadi.

Statistik: Maksimum

Ambang batas yang disarankan: 80,0

Pembenaran ambang batas: Anda disarankan untuk menetapkan ambang batas kurang dari atau sama dengan 80% sehingga Anda dapat memberikan waktu yang cukup untuk melakukan men-debug masalah sebelum sistem mulai merasakan dampak yang diakibatkannya.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

node_filesystem_utilization

Dimensi: ClusterName

Deskripsi alarm: Alarm ini akan membantu mendeteksi pemanfaatan sistem file yang berada dalam level tinggi pada simpul pekerja klaster EKS. Jika pemanfaatan yang tinggi itu terjadi secara terus-menerus, Anda mungkin perlu memperbarui simpul pekerja Anda agar Anda memiliki volume disk yang lebih besar, atau Anda mungkin perlu menskalakan secara horizontal.

Maksud: Alarm ini akan membantu Anda memantau pemanfaatan sistem file simpul pekerja yang ada di klaster EKS. Jika pemanfaatan mencapai 100%, hal itu akan dapat menyebabkan terjadinya kegagalan aplikasi, kemacetan I/O disk, pengosongan pod, atau simpul menjadi tidak sepenuhnya responsif.

Statistik: Maksimum

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Jika ada tekanan disk yang memadai (artinya disk semakin penuh), maka simpul akan ditandai sebagai simpul yang tidak sehat, dan pod akan dikeluarkan dari simpul. Pod yang ada pada sebuah simpul dengan tekanan disk akan dikosongkan ketika sistem file yang tersedia berada pada level yang lebih rendah dari ambang batas pengosongan yang telah ditetapkan pada kubelet. Anda harus mengatur ambang batas alarm sedemikian rupa sehingga Anda akan memiliki cukup waktu untuk bereaksi sebelum simpul tersebut dikeluarkan dari klaster.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

node_memory_utilization

Dimensi: ClusterName

Deskripsi alarm: Alarm ini akan membantu Anda dalam mendeteksi pemanfaatan memori yang tinggi pada simpul pekerja di klaster EKS. Jika pemanfaatannya berada dalam level yang tinggi

secara terus-menerus, hal ini mungkin menunjukkan bahwa Anda perlu menskalakan jumlah replika pod, atau Anda harus mengoptimalkan aplikasi Anda.

Maksud: Alarm ini akan membantu Anda memantau pemanfaatan memori dari simpul pekerja di klaster EKS sehingga penurunan performa sistem tidak terjadi.

Statistik: Maksimum

Ambang batas yang disarankan: 80,0

Pembenaran ambang batas: Anda disarankan untuk menetapkan ambang batas kurang dari atau sama dengan 80% sehingga Anda dapat memiliki waktu yang cukup untuk melakukan men-debug masalah sebelum sistem mulai merasakan dampak yang diakibatkannya.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

pod_cpu_utilization_over_pod_limit

Dimensi:ClusterName, Namespace, Layanan

Deskripsi alarm: Alarm ini akan membantu Anda dalam mendeteksi pemanfaatan CPU yang berada dalam level tinggi yang terjadi di pod klaster EKS. Jika pemanfaatannya secara terus-menerus berada dalam level yang tinggi, hal ini mungkin menunjukkan bahwa Anda harus menaikkan batas CPU untuk pod yang terpengaruh.

Maksud: Alarm ini akan membantu Anda memantau pemanfaatan CPU dari Pod yang dimiliki oleh Layanan Kubernetes yang ada di klaster EKS, sehingga Anda dapat dengan cepat mengidentifikasi apakah pod layanan mengkonsumsi CPU yang lebih tinggi dari yang diharapkan.

Statistik: Maksimum

Ambang batas yang disarankan: 80,0

Pembenaran ambang batas: Anda disarankan untuk menetapkan ambang batas kurang dari atau sama dengan 80% sehingga Anda dapat memiliki waktu yang cukup untuk melakukan men-debug masalah sebelum sistem mulai merasakan dampak yang diakibatkannya.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

pod_memory_utilization_over_pod_limit

Dimensi:ClusterName, Namespace, Layanan

Deskripsi alarm: Alarm ini akan membantu Anda dalam mendeteksi pemanfaatan memori yang berada dalam level tinggi yang terjadi di pod klaster EKS. Jika pemanfaatannya secara terus-menerus berada dalam level yang tinggi, hal ini mungkin menunjukkan bahwa Anda harus menaikkan batas memori untuk pod yang terpengaruh.

Maksud: Alarm ini akan membantu Anda memantau pemanfaatan memori dari pod yang ada di klaster EKS sehingga penurunan performa sistem tidak terjadi.

Statistik: Maksimum

Ambang batas yang disarankan: 80,0

Pembenaran ambang batas: Anda disarankan untuk menetapkan ambang batas kurang dari atau sama dengan 80% sehingga Anda dapat memiliki waktu yang cukup untuk melakukan men-debug masalah sebelum sistem mulai merasakan dampak yang diakibatkannya.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

Amazon Kinesis Data Streams

GetRecords.IteratorAgeMilliseconds

Dimensi: StreamName

Deskripsi alarm: Alarm ini dapat mendeteksi apakah usia maksimum iterator terlalu tinggi, atau tidak. Untuk aplikasi pemrosesan data waktu nyata, Anda perlu mengonfigurasikan retensi

data sesuai dengan toleransi penundaan. Ini biasanya memakan waktu beberapa menit. Untuk aplikasi-aplikasi yang memproses data historis, Anda perlu menggunakan metrik ini untuk memantau kecepatan penangkapan. Solusi cepat untuk menghentikan peristiwa kehilangan data adalah dengan meningkatkan periode retensi saat Anda melakukan pemecahan masalah. Anda juga dapat menaikkan jumlah pekerja yang memproses catatan dalam aplikasi konsumen Anda. Penyebab yang paling umum yang mengakibatkan peningkatan usia iterator bertahap adalah sumber daya fisik yang tidak memadai atau logika pemrosesan catatan yang belum diskalakan dengan peningkatan throughput aliran. Silakan lihat tautan ini untuk informasi selengkapnya.

Maksud: Alarm ini digunakan untuk mendeteksi apakah data dalam aliran Anda akan menjadi kedaluwarsa karena sudah disimpan terlalu lama atau karena pemrosesan catatan yang dilakukan terlalu lambat. Hal ini akan membantu Anda dalam mencegah terjadinya kehilangan data setelah mencapai 100% waktu retensi aliran.

Statistik: Maksimum

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Nilai ambang batas yang kami sarankan untuk alarm ini sangat tergantung pada periode retensi aliran dan toleransi penundaan pemrosesan untuk catatan. Anda harus meninjau persyaratan Anda dan kemudian melakukan analisis tren historis, lalu menetapkan ambang batasnya dengan jumlah milidetik yang mewakili penundaan pemrosesan kritis. Jika usia iterator melewati 50% periode retensi (secara default, 24 jam, dapat dikonfigurasi hingga 365 hari), maka akan ada risiko terjadinya kehilangan data karena kedaluwarsanya catatan. Anda dapat memantau metrik untuk memastikan bahwa tidak ada serpihan Anda yang pernah mendekati batas ini.

Periode: 60

Titik data untuk alarm: 15

Periode evaluasi: 15

Operator Perbandingan: GREATER_THAN_THRESHOLD

GetRecords.Sukses

Dimensi: StreamName

Deskripsi alarm: Metrik ini akan meningkat setiap kali konsumen Anda berhasil membaca data dari streaming Anda. `GetRecords` tidak akan mengembalikan data

apa pun saat menerapkan pengecualian. Pengecualian yang paling umum adalah `ProvisionedThroughputExceededException` karena tingkat permintaan untuk alirannya yang terlalu tinggi, atau karena throughput yang tersedia sudah disajikan untuk detik tertentu. Kurangi frekuensi atau ukuran dari permintaan-permintaan Anda. Untuk informasi selengkapnya, silakan lihat Batas dalam Panduan Developer Amazon Kinesis Data Streams, dan Mencoba Ulang Kesalahan dan Mundur Eksponensial di AWS.

Maksud: Alarm ini dapat mendeteksi apakah pengambilan catatan dari aliran oleh konsumen gagal, atau tidak. Dengan menyetel alarm pada metrik ini, Anda akan dapat secara proaktif mendeteksi setiap masalah yang terjadi dengan konsumsi data, misalnya peningkatan tingkat kesalahan atau penurunan pengambilan yang berhasil. Hal ini akan memungkinkan Anda untuk melakukan tindakan secara tepat waktu untuk menyelesaikan masalah yang mungkin terjadi di masa depan dan mempertahankan jalur pemrosesan data yang lancar.

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Bergantung pada pentingnya pengambilan catatan dari aliran, Anda harus menetapkan ambang batas ini berdasarkan toleransi aplikasi Anda terhadap catatan-catatan yang gagal. Ambang batas tersebut harus berupa persentase yang sesuai dari operasi yang berhasil. Anda dapat menggunakan data GetRecords metrik historis sebagai referensi untuk tingkat kegagalan yang dapat diterima. Anda juga harus mempertimbangkan percobaan ulang saat menyetel ambang batas ini karena catatan yang gagal dapat Anda coba lagi. Hal ini akan membantu mencegah lonjakan sementara sehingga tidak memicu peringatan yang tidak perlu.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: LESS_THAN_THRESHOLD

PutRecord.Sukses

Dimensi: StreamName

Deskripsi alarm: Alarm ini akan mendeteksi apakah jumlah operasi `PutRecord` yang gagal sudah melanggar ambang batas, atau tidak. Anda perlu menyelidiki log produsen data untuk menemukan akar penyebab yang mengakibatkan terjadinya kegagalan. Alasan yang paling umum

untuk kegagalan tersebut karena throughput yang disediakan tidak memadai pada serpihan yang menyebabkan `ProvisionedThroughputExceededException`. Hal itu terjadi karena tingkat permintaan untuk aliran terlalu tinggi, atau throughput yang dicoba untuk dimasukkan ke dalam serpihan terlalu tinggi. Kurangi frekuensi atau ukuran dari permintaan-permintaan Anda. Untuk informasi selengkapnya, lihat Batas Aliran dan Percobaan Ulang Kesalahan dan Backoff Eksponensial di. AWS

Maksud: Alarm ini dapat mendeteksi apakah penyerapan catatan ke dalam aliran mengalami kegagalan, atau tidak. Hal ini akan membantu Anda mengidentifikasi masalah yang terjadi dalam penulisan data ke aliran. Dengan menyetel alarm pada metrik ini, Anda akan dapat secara proaktif mendeteksi masalah yang dialami produsen saat menerbitkan data ke aliran, seperti terjadinya peningkatan tingkat kesalahan atau menurunnya catatan sukses yang diterbitkan. Hal ini akan memungkinkan Anda untuk melakukan tindakan-tindakan secara tepat waktu untuk mengatasi masalah yang mungkin terjadi dan untuk mempertahankan proses penyerapan data yang andal.

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Bergantung pada pentingnya penyerapan data dan pemrosesan data untuk layanan Anda, Anda harus menetapkan ambang batasnya berdasarkan toleransi aplikasi Anda terhadap catatan-catatan yang gagal. Ambang batas tersebut harus berupa persentase yang sesuai dari operasi yang berhasil. Anda dapat menggunakan data PutRecord metrik historis sebagai referensi untuk tingkat kegagalan yang dapat diterima. Anda juga harus mempertimbangkan percobaan ulang saat menyetel ambang batas ini karena catatan yang gagal dapat Anda coba lagi.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: LESS_THAN_THRESHOLD

PutRecords.FailedRecords

Dimensi: StreamName

Deskripsi alarm: Alarm ini akan mendeteksi apakah jumlah `PutRecords` yang mengalami kegagalan sudah melebihi ambang batas, atau tidak. Aliran Data Kinesis akan berusaha untuk

memproses semua catatan dalam setiap permintaan `PutRecords`, tetapi satu kegagalan catatan yang terjadi tidak akan menghentikan pemrosesan catatan berikutnya. Alasan utama yang mengakibatkan terjadinya kegagalan ini adalah dilampauinya throughput aliran atau serpihan individu. Penyebab umumnya adalah lonjakan lalu lintas dan latensi jaringan yang menyebabkan catatan tiba ke aliran dengan tidak merata. Anda harus mendeteksi catatan-catatan yang gagal diproses dan mencobanya kembali dalam panggilan berikutnya. Lihat Menangani Kegagalan Saat Menggunakan PutRecords untuk detail selengkapnya.

Maksud: Alarm ini dapat mendeteksi kegagalan yang terjadi secara terus-menerus saat Anda menggunakan operasi batch untuk menempatkan catatan ke aliran Anda. Jika Anda menyetel alarm pada metrik ini, maka Anda dapat secara proaktif mendeteksi peningkatan catatan yang mengalami kegagalan, memungkinkan Anda melakukan tindakan secara tepat waktu untuk mengatasi masalah mendasar dan memastikan proses penyerapan data berjalan dengan lancar dan andal.

Statistik: Jumlah

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda harus menetapkan ambang batas ini dengan jumlah catatan yang mengalami kegagalan yang mencerminkan toleransi aplikasi untuk catatan yang gagal. Anda dapat menggunakan data historis sebagai acuan untuk nilai kegagalan yang dapat diterima. Anda juga harus mempertimbangkan percobaan ulang saat menyetel ambang batas karena catatan yang gagal dapat dicoba lagi dalam panggilan berikutnya PutRecords .

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

ReadProvisionedThroughputExceeded

Dimensi: StreamName

Deskripsi alarm: Alarm akan melacak jumlah catatan yang mengakibatkan throttling pada kapasitas throughput baca. Jika Anda menyadari bahwa Anda dibatasi secara terus-menerus, maka Anda harus mempertimbangkan untuk menambahkan lebih banyak serpihan ke aliran Anda untuk meningkatkan throughput baca yang disediakan. Jika ada lebih dari satu aplikasi

konsumen yang berjalan di aliran, dan aplikasi-aplikasi itu berbagi batas `GetRecords`, kami menyarankan agar Anda mendaftarkan aplikasi konsumen baru melalui Enhanced Fan-Out. Jika dengan menambahkan lebih banyak serpihan tidak membuat jumlah throttling menurun, maka Anda mungkin memiliki serpihan "panas" yang sedang dibaca lebih dari pecahan lainnya. Anda perlu mengaktifkan pemantauan yang disempurnakan, dan kemudian menemukan serpihan "panas" itu, dan memisahkannya.

Maksud: Alarm ini dapat mendeteksi apakah konsumen mengalami throttling saat melebihi throughput baca yang Anda sediakan (ditentukan oleh jumlah serpihan yang Anda miliki). Dalam hal ini, Anda tidak akan dapat membaca dari aliran, dan aliran dapat mulai melakukan pencadangan.

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Biasanya permintaan-permintaan yang mengalami throttling dapat dicoba ulang dan karenanya mengatur ambang batasnya dengan nol akan membuat alarm menjadi terlalu sensitif. Namun demikian, throttling yang terjadi secara terus-menerus dapat memengaruhi pembacaan dari aliran dan akan memicu alarm. Anda harus menetapkan ambang batasnya dengan persentase yang sesuai dengan permintaan yang mengalami throttling untuk aplikasi dan mencoba lagi konfigurasi.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

SubscribeToShardEvent.MillisBehindLatest

Dimensi:StreamName, ConsumerName

Deskripsi alarm: Alarm ini akan mendeteksi apakah penundaan pemrosesan catatan dalam aplikasi telah melanggar ambang batas, atau tidak. Masalah-masalah sementara, seperti kegagalan operasi API ke aplikasi hilir, akan dapat menyebabkan peningkatan metrik yang terjadi secara tiba-tiba. Anda harus menyelidiki apakah hal itu terjadi secara terus-menerus, atau tidak. Penyebab umumnya adalah konsumen tidak memproses catatan dengan cukup cepat karena sumber daya fisik yang tidak memadai atau logika pemrosesan catatan yang belum

diskalakan dengan peningkatan throughput aliran. Pemblokiran panggilan di jalur kritis sering kali menjadi penyebab terjadinya perlambatan dalam pemrosesan catatan. Anda dapat meningkatkan paralelisme Anda dengan meningkatkan jumlah serpihan. Anda juga harus mengonfirmasi bahwa simpul pemrosesan yang mendasarinya memiliki sumber daya fisik yang memadai saat permintaan mencapai puncaknya.

Maksud: Alarm ini dapat mendeteksi penundaan dalam berlangganan peristiwa serpihan aliran. Hal ini menunjukkan bahwa terjadi kelambatan pemrosesan dan dapat membantu Anda mengidentifikasi potensi masalah yang mungkin terjadi pada performa aplikasi konsumen atau kesehatan aliran secara keseluruhan. Ketika kelambatan pemrosesan menjadi semakin parah, Anda harus menyelidiki dan mengatasi kemacetan atau ketidakefisienan aplikasi konsumen untuk menjamin pelaksanaan pemrosesan data waktu nyata dan meminimalkan backlog data.

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Nilai ambang batas yang kami sarankan untuk alarm ini sangat bergantung pada penundaan yang dapat ditoleransi oleh aplikasi Anda. Anda perlu meninjau persyaratan aplikasi Anda dan melakukan analisis tren historis, dan kemudian pilih ambang batas yang sesuai. Ketika SubscribeToShard panggilan berhasil, konsumen Anda mulai menerima SubscribeToShardEvent acara melalui koneksi persisten hingga 5 menit, setelah itu Anda perlu menelepon SubscribeToShard lagi untuk memperbarui langganan jika Anda ingin terus menerima catatan.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

WriteProvisionedThroughputExceeded

Dimensi: StreamName

Deskripsi alarm: Alarm ini akan mendeteksi ketika jumlah catatan yang mengakibatkan throttling kapasitas throughput penulisan mencapai ambang batasnya. Ketika produsen Anda melebihi throughput penulisan yang disediakan (ditentukan oleh jumlah serpihan yang Anda miliki), mereka akan mengalami throttling dan Anda tidak akan dapat memasukkan catatan ke aliran.

Untuk mengatasi throttling yang terjadi secara terus-menerus, Anda harus mempertimbangkan untuk menambahkan serpihan ke aliran Anda. Hal ini menaikkan throughput penulisan yang Anda sediakan dan throttling terjadi lagi di masa depan. Anda juga harus mempertimbangkan pilihan kunci partisi saat menyerap catatan. Kunci partisi acak lebih disukai karena kunci ini akan menyebarkan catatan secara merata di seluruh serpihan aliran, bila memungkinkan.

Maksud: Alarm ini dapat mendeteksi apakah produsen Anda ditolak, atau tidak, untuk menulis catatan karena terjadi throttling aliran atau serpihan. Jika aliran Anda berada dalam mode Provisioned, maka dengan menyetel alarm ini akan membantu Anda secara proaktif dalam mengambil tindakan-tindakan saat aliran data mencapai batasnya, dan memungkinkan Anda mengoptimalkan kapasitas yang disediakan atau melakukan tindakan-tindakan penskalaan yang sesuai untuk menghindari terjadinya kehilangan data dan menjaga kelancaran pemrosesan data.

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Biasanya permintaan yang mengalami throttling dapat dicoba lagi, jadi dengan menyetel ambang batas dengan nilai nol akan membuat alarm menjadi terlalu sensitif. Namun demikian, throttling yang terjadi secara terus-menerus akan dapat memengaruhi penulisan ke aliran, dan Anda harus mengatur ambang batas alarm untuk mendeteksi hal ini. Anda harus menetapkan ambang batasnya dengan persentase yang sesuai dengan permintaan yang mengalami throttling untuk aplikasi dan mencoba lagi konfigurasi.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

Lambda

ClaimedAccountConcurrency

Dimensi: Tidak ada

Deskripsi alarm: Alarm ini membantu memantau apakah konkurensi fungsi Lambda Anda mendekati batas konkurensi tingkat Wilayah akun Anda. Sebuah fungsi akan mulai mengalami

throttling jika mencapai batas konkurensinya. Anda dapat mengambil tindakan-tindakan berikut untuk menghindari terjadinya throttling.

1. Minta peningkatan konkurensi di Wilayah ini.
2. Identifikasi dan kurangi konkurensi cadangan yang tidak terpakai atau konkurensi yang disediakan.
3. Identifikasi masalah kinerja dalam fungsi untuk meningkatkan kecepatan pemrosesan dan karenanya meningkatkan throughput.
4. Tingkatkan ukuran batch fungsi, sehingga lebih banyak pesan diproses oleh setiap pemanggilan fungsi.

Maksud: Alarm ini dapat mendeteksi secara proaktif jika konkurensi fungsi Lambda Anda mendekati kuota konkurensi tingkat Wilayah akun Anda, sehingga Anda dapat menindaklanjutinya. Fungsi dibatasi jika `ClaimedAccountConcurrency` mencapai kuota konkurensi tingkat Wilayah akun. Jika Anda menggunakan Reserved Concurrency (RC) atau Provisioned Concurrency (PC), alarm ini memberi Anda lebih banyak visibilitas tentang pemanfaatan konkurensi daripada alarm yang digunakan. `ConcurrentExecutions`

Statistik: Maksimum

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda harus menghitung nilai sekitar 90% dari kuota konkurensi yang ditetapkan untuk akun di Wilayah, dan menggunakan hasilnya sebagai nilai ambang batas. Secara default, akun Anda memiliki kuota konkurensi sebesar 1.000 di semua fungsi di sebuah Wilayah. Namun, Anda harus memeriksa kuota akun Anda dari dasbor Service Quotas.

Periode: 60

Titik data untuk alarm: 10

Periode evaluasi: 10

Operator Perbandingan: GREATER_THAN_THRESHOLD

Kesalahan

Dimensi: FunctionName

Deskripsi alarm: Alarm ini akan mendeteksi terjadinya jumlah kesalahan yang tinggi. Kesalahan tersebut mencakup pengecualian yang dibuat oleh kode serta pengecualian yang dibuat oleh

runtime Lambda. Anda dapat memeriksa log yang terkait dengan fungsi tersebut untuk melakukan diagnosis masalah.

Maksud: Alarm ini akan membantu mendeteksi jumlah kesalahan yang tinggi dalam invokasi fungsi.

Statistik: Jumlah

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda harus menetapkan ambang batas dengan angka yang lebih besar dari nol. Nilai yang tepat dapat bergantung pada toleransi terhadap kesalahan yang diterapkan dalam aplikasi Anda. Anda juga harus memahami kekritisan invokasi yang ditangani oleh fungsi tersebut. Untuk beberapa aplikasi, kesalahan apa pun mungkin tidak akan dapat diterima, sedangkan untuk aplikasi yang lain mungkin memungkinkan margin kesalahan tertentu.

Periode: 60

Titik data untuk alarm: 3

Periode evaluasi: 3

Operator Perbandingan: GREATER_THAN_THRESHOLD

Pembatasan

Dimensi: FunctionName

Deskripsi alarm: Alarm ini akan mendeteksi sejumlah besar permintaan invokasi yang mengalami throttling. Throttling terjadi ketika tidak ada konkurensi yang tersedia untuk menaikkan skala. Ada beberapa pendekatan yang bisa dilakukan untuk mengatasi masalah ini. 1) Minta peningkatan konkurensi dari AWS Support di Wilayah ini. 2) Mengidentifikasi masalah performa yang terjadi dalam fungsi tersebut untuk meningkatkan kecepatan pemrosesan dan karenanya juga akan meningkatkan throughput. 3) Meningkatkan ukuran batch fungsi, sehingga lebih banyak pesan yang bisa diproses oleh masing-masing invokasi fungsi.

Maksud: Alarm ini akan membantu Anda dalam mendeteksi sejumlah besar permintaan invokasi yang mengalami throttling untuk fungsi Lambda. Penting bagi Anda untuk mengetahui apakah permintaan terus-menerus ditolak karena terjadinya throttling dan apakah Anda perlu meningkatkan performa fungsi Lambda atau meningkatkan kapasitas konkurensi untuk menghindari terjadinya throttling yang sulit diatasi.

Statistik: Jumlah

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda harus menetapkan ambang batas dengan angka yang lebih besar dari nol. Nilai pasti dari ambang batas ini dapat bergantung pada toleransi aplikasi. Anda harus menetapkan ambang batas tersebut sesuai dengan persyaratan penggunaan dan penskalaan fungsinya.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

Durasi

Dimensi: FunctionName

Deskripsi alarm: Alarm ini akan mendeteksi waktu durasi yang lama untuk mengolah suatu peristiwa dengan fungsi Lambda. Durasi yang lama tersebut mungkin saja terjadi karena adanya perubahan kode fungsi yang membuat fungsi membutuhkan waktu yang lebih lama untuk dijalankan, atau dependensi fungsi tersebut memakan waktu yang lebih lama.

Maksud: Alarm ini dapat mendeteksi durasi yang berjalan lama dari sebuah fungsi Lambda. Durasi runtime yang tinggi menjadi indikasi bahwa suatu fungsi memerlukan waktu yang lebih lama untuk invokasi, dan hal ini juga dapat memengaruhi kapasitas invokasi konkurensi apabila Lambda menangani jumlah peristiwa yang lebih tinggi. Sangat penting bagi Anda untuk mengetahui apakah fungsi Lambda tersebut terus-menerus memakan waktu eksekusi yang lebih lama dari yang diharapkan, atau tidak.

Statistik: p90

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Ambang batas untuk durasi akan tergantung pada aplikasi dan beban kerja Anda serta persyaratan performa Anda. Untuk persyaratan performa tinggi, Anda perlu menetapkan ambang batasnya dengan waktu yang lebih singkat untuk melihat apakah fungsi tersebut sesuai dengan harapan. Anda juga dapat melakukan analisis data historis

untuk metrik durasi untuk melihat apakah waktu yang dibutuhkan sesuai dengan ekspektasi performa yang diharapkan dari fungsi tersebut, dan kemudian Anda juga perlu mengatur ambang batasnya dengan waktu yang lebih lama daripada waktu rata-rata sesuai riwayatnya. Anda harus memastikan untuk mengatur ambang batas tersebut lebih rendah dari batas waktu fungsi yang dikonfigurasi.

Periode: 60

Titik data untuk alarm: 15

Periode evaluasi: 15

Operator Perbandingan: GREATER_THAN_THRESHOLD

ConcurrentExecutions

Dimensi: FunctionName

Deskripsi alarm: Alarm ini akan membantu Anda memantau apakah konkurensi fungsi sudah mendekati batas konkurensi tingkat Wilayah yang ditetapkan untuk akun Anda. Sebuah fungsi akan mulai mengalami throttling jika mencapai batas konkurensinya. Anda dapat mengambil tindakan-tindakan berikut untuk menghindari terjadinya throttling.

1. Minta peningkatan konkurensi di Wilayah ini.
2. Identifikasi masalah kinerja dalam fungsi untuk meningkatkan kecepatan pemrosesan dan karenanya meningkatkan throughput.
3. Tingkatkan ukuran batch fungsi, sehingga lebih banyak pesan diproses oleh setiap pemanggilan fungsi.

Untuk mendapatkan visibilitas yang lebih baik pada konkurensi cadangan dan pemanfaatan konkurensi yang disediakan, setel alarm pada metrik baru sebagai gantinya. `ClaimedAccountConcurrency`

Maksud: Alarm ini dapat mendeteksi secara proaktif apakah konkurensi fungsi sudah mendekati kuota konkurensi tingkat Wilayah yang ditetapkan untuk akun Anda, sehingga Anda dapat menindaklanjutinya. Sebuah fungsi mengalami throttling jika fungsi tersebut mencapai kuota konkurensi tingkat Wilayah yang ditetapkan untuk akun tersebut.

Statistik: Maksimum

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda harus menetapkan ambang batasnya menjadi sekitar 90% dari kuota konkurensi yang ditetapkan untuk akun di Wilayah. Secara default, akun Anda memiliki kuota konkurensi sebesar 1.000 di semua fungsi di sebuah Wilayah. Namun, Anda dapat memeriksa kuota akun Anda, karena dapat ditingkatkan dengan menghubungi AWS dukungan.

Periode: 60

Titik data untuk alarm: 10

Periode evaluasi: 10

Operator Perbandingan: GREATER_THAN_THRESHOLD

Wawasan Lambda

Kami menyarankan Anda untuk menyetel alarm-alarm praktik terbaik untuk metrik Wawasan Lambda berikut ini.

memory_utilization

Dimensi: function_name

Deskripsi alarm: Alarm ini digunakan untuk mendeteksi apakah pemanfaatan memori dari sebuah fungsi lambda sudah mendekati batas yang dikonfigurasikan untuknya. Untuk pemecahan masalah, Anda dapat mencoba untuk 1) Mengoptimalkan kode Anda. 2) Mengukur alokasi memori Anda dengan benar dengan memperkirakan kebutuhan memori yang Anda perlukan secara akurat. Anda dapat melihat Lambda Power Tuning untuk mendapatkan informasi selengkapnya tengang hal ini. 3) Menggunakan penyatuan koneksi. Silakan lihat Menggunakan Proksi Amazon RDS dengan Lambda untuk melakukan penyatuan koneksi untuk basis data RDS. 4) Anda juga dapat mempertimbangkan untuk merancang fungsi Anda dengan tujuan untuk mencegah penyimpanan sejumlah besar data dalam memori di antara invokasi.

Maksud: Alarm ini digunakan untuk mendeteksi apakah pemanfaatan memori untuk fungsi Lambda sudah mendekati batas yang dikonfigurasikan.

Statistik: Rata-rata

Saran Ambang Batas: 90,0

Pembenaran Ambang Batas:Anda perlu mengatur ambang batas menjadi 90% untuk mendapatkan peringatan ketika pemanfaatan memori melampaui 90% dari memori yang

dialokasikan. Anda dapat menyesuaikan nilai ambang batas ini dengan nilai yang lebih rendah jika Anda memiliki kekhawatiran terhadap beban kerja untuk pemanfaatan memori. Anda juga dapat memeriksa data historis untuk metrik ini dan menetapkan ambang batas sesuai dengan data historis tersebut.

Periode: 60

Titik data untuk alarm: 10

Periode evaluasi: 10

ComparisonOperator: LEBIH BESAR_THAN_THRESHOLD

Amazon VPC (**`AWS/NATGateway`**)

ErrorPortAllocation

Dimensi: NatGatewayId

Deskripsi alarm: Alarm ini akan membantu Anda dalam mendeteksi ketika NAT Gateway tidak dapat memberikan alokasi port untuk koneksi baru. Untuk mengatasi permasalahan ini, Anda bisa lihat Mengatasi kesalahan alokasi port pada NAT Gateway.

Maksud: Alarm ini akan digunakan untuk mendeteksi apakah NAT gateway tidak dapat mengalokasikan port sumber.

Statistik: Jumlah

Ambang batas yang disarankan: 0,0

Pembenaran ambang batas: Jika nilai ErrorPortAllocation lebih besar dari nol, itu berarti terlalu banyak koneksi bersamaan ke satu tujuan populer terbuka melalui NatGateway.

Periode: 60

Titik data untuk alarm: 15

Periode evaluasi: 15

Operator Perbandingan: GREATER_THAN_THRESHOLD

PacketsDropCount

Dimensi: NatGatewayId

Deskripsi alarm: Alarm ini akan membantu Anda dalam mendeteksi apakah paket sudah dijatuhkan oleh NAT Gateway, atau tidak. Ini mungkin terjadi karena masalah dengan NAT Gateway, jadi periksa dasbor kesehatan AWS layanan untuk status AWS NAT Gateway di Wilayah Anda. Hal ini akan dapat membantu Anda dalam menghubungkan masalah jaringan yang terjadi yang berkaitan dengan lalu lintas dengan menggunakan NAT gateway.

Maksud: Alarm ini akan digunakan untuk mendeteksi apakah paket sudah dijatuhkan oleh NAT Gateway, atau tidak.

Statistik: Jumlah

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda harus menghitung nilai 0,01 persen dari total lalu lintas di NAT Gateway dan menggunakan hasil perhitungan itu sebagai nilai ambang batas. Anda juga perlu menggunakan data historis lalu lintas pada NAT Gateway untuk menentukan ambang batas tersebut.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

AWS Tautan Pribadi (**`AWS/PrivateLinkEndpoints`**)

PacketsDropped

Dimensi: Id VPC, Id Titik Akhir VPC, Jenis Titik Akhir, Id Subnet, Nama Layanan

Deskripsi alarm: Alarm ini akan membantu Anda dalam mendeteksi apakah titik akhir atau layanan titik akhir sedang dalam kondisi tidak sehat dengan memantau jumlah paket yang dijatuhkan oleh titik akhir. Perlu Anda perhatikan bahwa paket yang lebih besar dari 8500 byte yang tiba di titik akhir VPC akan dijatuhkan. Untuk pemecahan masalah, silakan lihat masalah konektivitas antara sebuah titik akhir VPC antarmuka dan sebuah layanan titik akhir.

Maksud: Alarm ini akan digunakan untuk mendeteksi apakah titik akhir atau layanan titik akhir sedang dalam kondisi tidak sehat.

Statistik: Jumlah

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda perlu menetapkan ambang batas ini sesuai dengan kasus penggunaan Anda. Jika Anda ingin mengetahui status titik akhir atau layanan titik akhir yang sedang dalam kondisi tidak sehat, maka Anda harus menetapkan ambang batas dengan nilai rendah sehingga Anda akan mendapatkan kesempatan untuk memperbaiki masalah sebelum Anda mengalami kehilangan data dalam jumlah besar. Anda juga dapat menggunakan data historis untuk memahami toleransi terhadap paket-paket yang dijatuhkan dan menetapkan ambang batas sesuai dengan data historis tersebut.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

AWS Tautan Pribadi (**`AWS/PrivateLinkServices`**)

RstPacketsSent

Dimensi: Id Layanan, Penyeimbang Beban Arn, Az

Deskripsi alarm: Alarm ini akan membantu Anda dalam mendeteksi target yang sedang dalam kondisi tidak sehat dari layanan titik akhir berdasarkan jumlah paket reset yang dikirim ke titik akhir. Saat Anda men-debug kesalahan koneksi dengan konsumen layanan Anda, Anda dapat memvalidasi apakah layanan mengatur ulang koneksi dengan RstPacketsSent metrik, atau jika ada hal lain yang gagal di jalur jaringan.

Maksud: Alarm ini akan digunakan untuk mendeteksi target yang sedang dalam kondisi tidak sehat dari layanan titik akhir.

Statistik: Jumlah

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Ambang batas ini akan bergantung pada kasus penggunaan Anda. Jika kasus penggunaan Anda dapat memberikan toleransi pada target yang sedang dalam kondisi

tidak sehat, maka Anda dapat menetapkan ambang batas dengan nilai yang tinggi. Jika kasus penggunaan tersebut tidak dapat memberikan toleransi pada target yang tidak sehat, maka Anda dapat menetapkan ambang batas dengan nilai yang sangat rendah.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

Amazon RDS

CPUUtilization

Dimensi: DB InstanceIdentifier

Deskripsi alarm: Alarm ini akan membantu Anda dalam memantau pemanfaatan CPU yang berada dalam level yang tinggi secara konsisten. Pemanfaatan CPU mengukur waktu non-idle. Pertimbangkan untuk menggunakan Pemantauan yang Ditingkatkan atau Wawasan Performa untuk meninjau waktu tunggu mana yang menghabiskan sebagian besar waktu CPU (`guest`, `irq`, `wait`, `nice` dan sebagainya) untuk MariaDB, MySQL, Oracle, dan PostgreSQL. Kemudian evaluasi kueri-kueri mana yang mengkonsumsi jumlah CPU yang paling tinggi. Jika Anda tidak dapat menyetel beban kerja Anda, pertimbangkanlah untuk pindah ke kelas instans DB yang lebih besar.

Maksud: Alarm ini digunakan untuk mendeteksi pemanfaatan CPU yang berada dalam level yang tinggi secara konsisten untuk mencegah waktu respons dan batas waktu yang sangat tinggi. Jika Anda ingin memeriksa micro-bursting yang terjadi pada pemanfaatan CPU, Anda dapat mengatur waktu evaluasi alarm yang lebih rendah.

Statistik: Rata-rata

Ambang batas yang disarankan: 90,0

Pembenaran ambang batas: Lonjakan acak dalam konsumsi CPU mungkin tidak akan menghambat performa basis data, tetapi pemanfaatan CPU yang tinggi yang terus-menerus dapat menghambat permintaan basis data yang akan datang. Bergantung pada beban kerja basis data secara keseluruhan, CPU tinggi pada instans RDS/Aurora Anda dapat menurunkan performa secara keseluruhan.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

DatabaseConnections

Dimensi: DB InstanceIdentifier

Deskripsi alarm: Alarm ini mendeteksi koneksi dalam jumlah besar. Tinjau koneksi-koneksi yang ada dan hentikan semua koneksi yang berada dalam keadaan `sleep` atau koneksi yang tidak ditutup dengan benar. Pertimbangkan untuk menggunakan penyatuan koneksi untuk membatasi jumlah koneksi yang baru. Atau, tingkatkan ukuran instans DB untuk menggunakan kelas yang memiliki lebih banyak memori dan karenanya memiliki nilai default yang lebih tinggi untuk `max_connections` atau tingkatkan nilai `max_connections` di RDS dan Aurora MySQL dan PostgreSQL untuk kelas saat ini jika hal itu dapat mendukung beban kerja Anda.

Maksud: Alarm ini digunakan untuk membantu mencegah koneksi-koneksi yang ditolak ketika jumlah maksimum koneksi DB tercapai. Alarm ini tidak disarankan jika Anda sering mengubah kelas instans DB, karena hal itu akan mengubah memori dan jumlah koneksi maksimum default.

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Jumlah koneksi yang diperkenankan bergantung pada ukuran kelas instans DB Anda dan parameter spesifik mesin basis data yang terkait dengan proses/koneksi. Anda harus menghitung nilai antara 90-95% dari jumlah maksimum koneksi untuk basis data Anda dan menggunakan hasil tersebut sebagai nilai ambang batas.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

EBS% ByteBalance

Dimensi: DB InstanceIdentifier

Deskripsi alarm: Alarm ini akan membantu Anda dalam memantau persentase kredit throughput yang tersisa yang berada dalam level yang rendah. Untuk pemecahan masalah, silakan periksa masalah latensi di RDS.

Maksud: Alarm ini digunakan untuk mendeteksi persentase sisa kredit throughput di bucket burst yang berada dalam level rendah. Persentase saldo byte yang rendah dapat menyebabkan terjadinya masalah-masalah kemacetan throughput. Alarm ini tidak disarankan untuk instans Aurora PostgreSQL.

Statistik: Rata-rata

Ambang batas yang disarankan: 10,0

Pembenaran ambang batas: Saldo kredit throughput di bawah 10% dianggap buruk dan Anda harus menetapkan ambang batas yang semestinya. Anda juga dapat menetapkan sebuah ambang batas yang lebih rendah jika aplikasi Anda dapat mentolerir throughput yang lebih rendah untuk beban kerja tersebut.

Periode: 60

Titik data untuk alarm: 3

Periode evaluasi: 3

Operator Perbandingan: LESS_THAN_THRESHOLD

EBSIOBalance%

Dimensi: DB InstanceIdentifier

Deskripsi alarm: Alarm ini akan membantu Anda dalam memantau persentase kredit IOPS yang tersisa yang berada dalam level yang rendah. Untuk pemecahan masalah, silakan lihat masalah latensi di RDS.

Maksud: Alarm ini digunakan untuk mendeteksi persentase sisa kredit I/O di bucket burst yang berada dalam level rendah. Persentase saldo byte yang rendah dapat menyebabkan terjadinya masalah-masalah kemacetan IOPS. Alarm ini tidak disarankan untuk instans Aurora.

Statistik: Rata-rata

Ambang batas yang disarankan: 10,0

Pembenaran ambang batas: Saldo kredit IOPS di bawah 10% dianggap buruk dan Anda dapat menetapkan ambang batas yang semestinya. Anda juga dapat menetapkan sebuah ambang batas yang lebih rendah jika aplikasi Anda dapat mentolerir IOPS yang lebih rendah untuk beban kerja tersebut.

Periode: 60

Titik data untuk alarm: 3

Periode evaluasi: 3

Operator Perbandingan: LESS_THAN_THRESHOLD

FreeableMemory

Dimensi: DB InstanceIdentifier

Deskripsi alarm: Alarm ini akan membantu Anda dalam memantau memori yang bisa dibebaskan yang berada dalam level yang rendah yang dapat berarti bahwa ada lonjakan koneksi basis data atau instans Anda mungkin berada di bawah tekanan memori tinggi. Periksa tekanan memori dengan memantau CloudWatch metrik untuk `SwapUsage` `selain. `FreeableMemory` Jika penggunaan memori instans sering berada dalam level yang terlalu tinggi, hal ini menunjukkan bahwa Anda harus memeriksa beban kerja Anda atau meningkatkan kelas instans Anda. Untuk instans DB pembaca Aurora, pertimbangkan untuk menambahkan instans DB pembaca tambahan ke klaster tersebut. Untuk informasi tentang pemecahan masalah Aurora, lihat masalah memori yang dapat dibebaskan.

Maksud: Alarm ini digunakan untuk membantu mencegah terjadinya kehabisan memori yang dapat mengakibatkan koneksi ditolak.

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Bergantung pada beban kerja dan kelas instans, nilai-nilai yang berbeda untuk ambang batas dapat dibenarkan. Idealnya, memori yang tersedia tidak boleh berada di bawah 25% dari total memori dalam waktu yang lama. Untuk Aurora, Anda dapat mengatur ambang batasnya mendekati 5%, karena metrik mendekati 0, itu artinya instans DB telah dinaikkan skalanya sebisa mungkin. Anda dapat menganalisis perilaku historis metrik ini untuk menentukan tingkat ambang batas yang masuk akal.

Periode: 60

Titik data untuk alarm: 15

Periode evaluasi: 15

Operator Perbandingan: LESS_THAN_THRESHOLD

FreeLocalStorage

Dimensi: DB InstanceIdentifier

Deskripsi alarm: Alarm ini akan membantu Anda dalam memantau penyimpanan lokal bebas dalam level yang rendah. Aurora PostgreSQL-Compatible Edition menggunakan penyimpanan lokal untuk menyimpan log kesalahan dan file sementara. Aurora MySQL menggunakan penyimpanan lokal untuk menyimpan log kesalahan, log umum, log kueri lambat, log audit, dan tabel-tabel sementara non-InnoDB. Volume penyimpanan lokal ini didukung oleh Amazon EBS dan dapat diperluas dengan menggunakan kelas instans DB yang lebih besar. Untuk pemecahan masalah, periksa Aurora PostgreSQL-Compatible dan MySQL-Compatible.

Maksud: Alarm ini digunakan untuk mendeteksi seberapa dekat instans Aurora DB untuk mencapai batas penyimpanan lokal, jika Anda tidak menggunakan Aurora Serverless v2 atau yang lebih tinggi. Penyimpanan lokal dapat mencapai kapasitas saat Anda menyimpan data non-persisten, seperti tabel sementara dan file log, di penyimpanan lokal tersebut. Alarm ini dapat mencegah out-of-space kesalahan yang terjadi ketika instans DB Anda kehabisan penyimpanan lokal.

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda harus menghitung sekitar 10% -20% dari jumlah penyimpanan yang tersedia berdasarkan kecepatan dan tren penggunaan volume, dan kemudian menggunakan hasil itu sebagai nilai ambang batas untuk secara proaktif mengambil tindakan sebelum volume tersebut mencapai batasnya.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: LESS_THAN_THRESHOLD

FreeStorageSpace

Dimensi: DB InstanceIdentifier

Deskripsi alarm: Alarm ini berfungsi untuk jumlah ruang penyimpanan yang tersedia yang rendah. Pertimbangkan untuk menaikkan skala penyimpanan basis data Anda jika Anda sering kali mendekati batas kapasitas penyimpanan. Sertakan beberapa buffer untuk mengakomodasi peningkatan permintaan yang tidak terduga dari aplikasi Anda. Atau, pertimbangkan untuk mengaktifkan penskalaan otomatis penyimpanan RDS. Selain itu, pertimbangkan untuk membebaskan lebih banyak ruang dengan menghapus data dan log yang tidak terpakai atau sudah usang. Untuk informasi lebih lanjut, silakan periksa dokumen RDS kehabisan penyimpanan dan dokumen masalah penyimpanan PostgreSQL.

Maksud: Alarm ini akan membantu Anda mencegah masalah-masalah penyimpanan penuh. Hal ini dapat mencegah terjadinya downtime ketika instans basis data Anda kehabisan penyimpanan. Kami tidak menyarankan Anda untuk menggunakan alarm ini jika Anda mengaktifkan penskalaan otomatis penyimpanan, atau jika Anda sering kali mengubah kapasitas penyimpanan instans basis data Anda.

Statistik: Minimum

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Nilai ambang batas akan tergantung pada ruang penyimpanan yang saat ini dialokasikan. Biasanya, Anda harus menghitung nilai 10 persen dari ruang penyimpanan yang dialokasikan dan menggunakan hasil itu sebagai nilai ambang batasnya.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: LESS_THAN_THRESHOLD

MaximumUsedTransactionID

Dimensi: DB InstanceIdentifier

Deskripsi alarm: Alarm ini akan membantu Anda dalam mencegah penutupan ID transaksi untuk PostgreSQL. Lihat langkah-langkah pemecahan masalah yang diuraikan di blog ini untuk menyelidiki dan menyelesaikan masalah. Anda juga dapat merujuk ke blog ini untuk

membiasakan diri Anda lebih jauh dengan konsep autovacuum, masalah umum, dan praktik terbaik.

Maksud: Alarm ini digunakan untuk membantu mencegah penutupan ID transaksi untuk PostgreSQL.

Statistik: Rata-rata

Ambang batas yang disarankan: 1.0E9

Pembenaran ambang batas: Menetapkan ambang batas ini menjadi 1 miliar akan memberi Anda waktu untuk menyelidiki masalahnya. Nilai default autovacuum_freeze_max_age adalah 200 juta. Jika usia transaksi tertua adalah 1 miliar, maka autovacuum akan mengalami masalah dalam menjaga ambang batas ini di bawah target 200 juta ID transaksi.

Periode: 60

Titik data untuk alarm: 1

Periode evaluasi: 1

Operator Perbandingan: GREATER_THAN_THRESHOLD

ReadLatency

Dimensi: DB InstanceIdentifier

Deskripsi alarm: Alarm ini akan membantu Anda dalam memantau latensi baca yang tinggi. Jika latensi penyimpanan berada dalam level yang tinggi, hal itu karena beban kerja melebihi batas sumber daya. Anda dapat meninjau pemanfaatan I/O relatif terhadap instans dan konfigurasi penyimpanan yang dialokasikan. Lihat pemecahan masalah latensi volume Amazon EBS yang disebabkan oleh kemacetan IOPS. Untuk Aurora, Anda dapat beralih ke kelas instans yang memiliki konfigurasi penyimpanan I/O-Optimized. Lihat Perencanaan I/O di Aurora untuk mendapatkan panduannya.

Maksud: Alarm ini digunakan untuk mendeteksi latensi baca tinggi. Disk basis data biasanya memiliki latensi baca/tulis yang rendah, tetapi mereka dapat memiliki masalah yang dapat menyebabkan terjadinya operasi latensi yang tinggi.

Statistik: p90

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Nilai ambang batas yang disarankan untuk alarm ini sangat tergantung pada kasus penggunaan Anda. Latensi-latensi baca yang lebih tinggi dari 20 milidetik kemungkinan menjadi penyebab terjadinya penyelidikan. Anda juga dapat menetapkan ambang batas yang lebih tinggi jika aplikasi Anda dapat memiliki latensi yang lebih tinggi untuk operasi baca. Tinjau kekritisan dan persyaratan latensi baca dan kemudian lakukan analisis perilaku historis dari metrik ini untuk menentukan tingkat ambang batas yang masuk akal.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

ReplicaLag

Dimensi: DB InstanceIdentifier

Deskripsi alarm: Alarm ini akan membantu Anda dalam memahami jumlah detik sebuah replika berada di belakang instans utama. Replika Baca PostgreSQL melaporkan keterlambatan replikasi hingga lima menit jika tidak ada transaksi pengguna yang dilakukan pada instans basis data sumber. Ketika ReplicaLag metrik mencapai 0, replika telah menyusul instans DB utama. Jika ReplicaLag metrik mengembalikan -1, maka replikasi saat ini tidak aktif. Untuk panduan terkait PostgreSQL RDS, lihat praktik terbaik replikasi dan untuk `ReplicaLag` pemecahan masalah dan kesalahan terkait, lihat pemecahan masalah. ReplicaLag

Maksud: Alarm ini dapat mendeteksi lag replika yang mencerminkan kehilangan data yang dapat terjadi jika ada kegagalan pada instans primernya. Jika replika tersebut terlalu jauh di belakang instans primer dan instans primer tersebut gagal, maka replika akan kehilangan data yang ada di instans primer.

Statistik: Maksimum

Ambang batas yang disarankan: 60,0

Pembenaran ambang batas: Biasanya, keterlambatan yang dapat diterima tergantung pada aplikasi. Kami merekomendasikan tidak lebih dari 60 detik.

Periode: 60

Titik data untuk alarm: 10

Periode evaluasi: 10

Operator Perbandingan: GREATER_THAN_THRESHOLD

WriteLatency

Dimensi: DB InstanceIdentifier

Deskripsi alarm: Alarm ini akan membantu Anda dalam memantau latensi tulis yang tinggi. Jika latensi penyimpanan berada dalam level yang tinggi, hal itu karena beban kerja melebihi batas sumber daya. Anda dapat meninjau pemanfaatan I/O relatif terhadap instans dan konfigurasi penyimpanan yang dialokasikan. Lihat pemecahan masalah latensi volume Amazon EBS yang disebabkan oleh kemacetan IOPS. Untuk Aurora, Anda dapat beralih ke kelas instans yang memiliki konfigurasi penyimpanan I/O-Optimized. Lihat Perencanaan I/O di Aurora untuk mendapatkan panduannya.

Maksud: Alarm ini digunakan untuk mendeteksi latensi tulis yang tinggi. Meskipun disk basis data biasanya memiliki latensi baca/tulis yang rendah, mereka mungkin mengalami masalah yang menyebabkan operasi latensi yang tinggi. Pemantauan ini akan memastikan kepada Anda latensi disk serendah yang diharapkan.

Statistik: p90

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Nilai ambang batas yang disarankan untuk alarm ini sangat tergantung pada kasus penggunaan Anda. Latensi-latensi tulis yang lebih tinggi dari 20 milidetik kemungkinan menjadi penyebab terjadinya penyelidikan. Anda juga dapat menetapkan ambang batas yang lebih tinggi jika aplikasi Anda dapat memiliki latensi yang lebih tinggi untuk operasi tulis. Tinjau kekritisan dan persyaratan latensi tulis dan kemudian lakukan analisis perilaku historis dari metrik ini untuk menentukan tingkat ambang batas yang masuk akal.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

DBLoad

Dimensi: DB InstanceIdentifier

Deskripsi alarm: Alarm ini akan membantu Anda dalam memantau beban DB yang tinggi. Jika jumlah proses melebihi jumlah vCPU, maka proses ini akan mulai mengantre. Ketika antrean meningkat, performa akan terpengaruh. Jika muatan DB sering melampaui vCPU maksimum dan status tunggu utamanya adalah CPU, maka CPU akan kelebihan muatan. Dalam hal ini, Anda dapat memantau `CPUUtilization`, `DBLoadCPU` dan tugas yang diantrekan di Wawasan Performa/Pemantauan yang Ditingkatkan. Anda mungkin ingin melakukan throttling koneksi ke instans, menyesuaikan kueri semua SQL dengan muatan CPU yang tinggi, atau mempertimbangkan kelas instans yang lebih besar. Instans yang tinggi dan konsisten dari setiap status tunggu menunjukkan bahwa mungkin terjadi kemacetan atau masalah ketidakcocokan sumber daya yang perlu diselesaikan.

Maksud: Alarm ini digunakan untuk mendeteksi muatan DB yang tinggi. Beban DB yang tinggi dapat menyebabkan terjadinya masalah performa pada instans DB. Alarm ini tidak berlaku untuk instans DB nirserver.

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Nilai vCPU maksimum ditentukan oleh jumlah inti vCPU (CPU virtual) untuk instans DB Anda. Bergantung pada vCPU maksimum, nilai-nilai yang berbeda untuk ambang batas tersebut bisa dibenarkan. Idealnya, muatan DB tidak boleh melebihi garis vCPU.

Periode: 60

Titik data untuk alarm: 15

Periode evaluasi: 15

Operator Perbandingan: GREATER_THAN_THRESHOLD

AuroraVolumeBytesLeftTotal

Dimensi: DB ClusterIdentifier

Deskripsi alarm: Alarm ini akan membantu Anda dalam memantau volume total sisa yang rendah. Ketika total volume yang tersisa mencapai batas ukuran, cluster melaporkan out-of-space kesalahan. Penyimpanan Aurora secara otomatis menskalakan dengan data dalam volume klaster dan bisa berekspansi hingga 128 TiB atau 64 TiB tergantung pada versi mesin DB. Pertimbangkan untuk mengurangi penyimpanan dengan menghapus tabel dan basis data yang tidak lagi Anda perlukan. Untuk informasi lebih lanjut, silakan periksa penskalaan penyimpanan.

Maksud: Alarm ini digunakan untuk mendeteksi seberapa dekat klaster Aurora dengan batas ukuran volume tersebut. Alarm ini dapat mencegah out-of-space kesalahan yang terjadi ketika cluster Anda kehabisan ruang. Alarm ini direkomendasikan hanya untuk Aurora MySQL.

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda harus menghitung 10% -20% dari batas ukuran yang sebenarnya berdasarkan kecepatan dan tren kenaikan penggunaan volume, dan kemudian menggunakan hasil itu sebagai nilai ambang batas untuk secara proaktif mengambil tindakan sebelum volume tersebut mencapai batasnya.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: LESS_THAN_THRESHOLD

AuroraBinlogReplicaLag

Dimensi: DBClusterIdentifier, peran = penulis

Deskripsi alarm: Alarm ini akan membantu Anda dalam memantau status kesalahan replikasi instans penulis Aurora. Untuk informasi selengkapnya, lihat Mereplikasi kluster DB MySQL Aurora di seluruh Wilayah. AWS Untuk pemecahan masalah, silakan lihat Masalah replikasi Aurora MySQL.

Maksud: Alarm ini digunakan untuk mendeteksi apakah instans penulis berada dalam status kesalahan, atau tidak, dan tidak dapat mereplikasi sumbernya. Alarm ini direkomendasikan hanya untuk Aurora MySQL.

Statistik: Rata-rata

Ambang batas yang disarankan: -1,0

Pembenaran ambang batas: Kami menyarankan Anda menggunakan -1 sebagai nilai ambang batas karena Aurora MySQL akan menerbitkan nilai ini jika replika berada dalam status kesalahan.

Periode: 60

Titik data untuk alarm: 2

Periode evaluasi: 2

Operator Perbandingan: LESS_THAN_OR_EQUAL_TO_THRESHOLD

BlockedTransactions

Dimensi: DB InstanceIdentifier

Deskripsi alarm: Alarm ini akan membantu Anda dalam memantau jumlah transaksi yang diblokir yang sedang dalam level yang tinggi dalam sebuah instans Aurora DB. Transaksi-transaksi yang diblokir dapat berakhir dengan rollback atau commit. Konkurensi yang berada dalam level yang tinggi, idle dalam transaksi, atau transaksi yang berjalan lama dapat menyebabkan terjadinya pemblokiran transaksi. Untuk pemecahan masalah, silakan lihat dokumentasi Aurora MySQL.

Maksud: Alarm ini digunakan untuk mendeteksi jumlah transaksi yang diblokir dalam sebuah instans Aurora DB untuk mencegah terjadinya kemunduran transaksi dan penurunan performa.

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda harus menghitung 5% dari semua transaksi instans Anda dengan menggunakan metrik `ActiveTransactions` dan menggunakan hasil itu sebagai nilai ambang batas. Anda juga dapat meninjau kekritisan dan persyaratan transaksi yang diblokir dan menganalisis perilaku historis metrik ini untuk menentukan tingkat ambang batas yang masuk akal.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

BufferCacheHitRatio

Dimensi: DB InstanceIdentifier

Deskripsi alarm: Alarm ini akan membantu Anda dalam memantau rasio hit cache yang berada dalam level yang rendah secara konsisten dari klaster Aurora. Rasio hit yang rendah menunjukkan bahwa kueri Anda pada instans DB ini sering kali masuk ke disk. Untuk pemecahan masalah, lakukan investigasi pada beban kerja Anda untuk melihat kueri mana yang menyebabkan perilaku ini, dan kemudian lihat dokumen rekomendasi RAM instans DB.

Maksud: Alarm ini digunakan untuk mendeteksi rasio hit cache yang berada dalam level rendah secara konsisten untuk mencegah terjadinya penurunan performa berkelanjutan pada instans Aurora.

Statistik: Rata-rata

Ambang batas yang disarankan: 80,0

Pembenaran ambang batas: Anda dapat mengatur ambang batas untuk rasio hit cache buffer menjadi 80%. Namun demikian, Anda dapat menyesuaikan nilai ini berdasarkan tingkat performa dan karakteristik beban kerja yang dapat diterima.

Periode: 60

Titik data untuk alarm: 10

Periode evaluasi: 10

Operator Perbandingan: LESS_THAN_THRESHOLD

EngineUptime

Dimensi: DBClusterIdentifier, peran = penulis

Deskripsi alarm: Alarm ini akan membantu Anda dalam memantau downtime rendah dari instans DB penulis. Instans DB penulis dapat turun karena adanya reboot, pemeliharaan, peningkatan, atau failover. Ketika waktu aktif mencapai 0 karena terjadi failover dalam klaster tersebut, dan klaster itu memiliki satu atau beberapa Replika Aurora, maka Replika Aurora akan dipromosikan ke instans penulis primer selama peristiwa kegagalan. Untuk meningkatkan ketersediaan klaster DB Anda, pertimbangkan untuk membuat setidaknya satu atau beberapa Replika Aurora di dua Zona Ketersediaan yang berbeda. Untuk informasi lebih lanjut, silakan periksa faktor-faktor yang memengaruhi downtime Aurora.

Maksud: Alarm ini digunakan untuk mendeteksi apakah instans DB penulis Aurora sedang berada dalam keadaan downtime. Hal ini dapat mencegah terjadinya kegagalan jangka panjang dalam instans penulis karena crash atau failover.

Statistik: Rata-rata

Ambang batas yang disarankan: 0,0

Pembenaran ambang batas: Peristiwa kegagalan mengakibatkan terjadinya interupsi dalam waktu singkat, di mana pada saat itu operasi baca dan tulis gagal dengan pengecualian. Namun demikian, layanan biasanya akan pulih dalam waktu kurang dari 60 detik, dan sering kali kurang dari 30 detik.

Periode: 60

Titik data untuk alarm: 2

Periode evaluasi: 2

Operator Perbandingan: LESS_THAN_OR_EQUAL_TO_THRESHOLD

RollbackSegmentHistoryListLength

Dimensi: DB InstanceIdentifier

Deskripsi alarm: Alarm ini akan membantu Anda dalam memantau panjang riwayat segmen rollback yang berada dalam level tinggi secara konsisten dari sebuah instans Aurora. Panjang daftar riwayat InnoDB yang tinggi menunjukkan bahwa sejumlah besar versi baris lama, kueri, dan shutdown basis data menjadi lebih lambat. Untuk informasi selengkapnya dan pemecahan masalah, silakan lihat dokumentasi panjang daftar riwayat InnoDB meningkat secara signifikan.

Maksud: Alarm ini digunakan untuk mendeteksi panjang riwayat segmen rollback yang berada dalam level tinggi secara konsisten. Hal ini dapat membantu Anda mencegah penurunan performa berkelanjutan dan penggunaan CPU yang tinggi yang terjadi di instans Aurora. Alarm ini direkomendasikan hanya untuk Aurora MySQL.

Statistik: Rata-rata

Ambang batas yang disarankan: 1000000,0

Pembenaran ambang batas: Menetapkan ambang batas ini menjadi 1 juta akan memberi Anda waktu untuk menyelidiki masalahnya. Namun demikian, Anda dapat menyesuaikan nilai ini berdasarkan tingkat performa dan karakteristik beban kerja yang dapat diterima.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

StorageNetworkThroughput

Dimensi: DBClusterIdentifier, peran = penulis

Deskripsi alarm: Alarm ini akan membantu Anda dalam memantau throughput jaringan penyimpanan yang tinggi. Jika throughput jaringan penyimpanan melewati total bandwidth jaringan dari instans EC2, hal ini akan dapat menyebabkan latensi baca dan tulis yang tinggi dan kemudian dapat menyebabkan penurunan performa. Anda dapat memeriksa jenis instans EC2 dari AWS Console. Untuk pemecahan masalah, silakan periksa setiap perubahan pada latensi tulis/baca dan evaluasi apakah Anda juga menekan alarm pada metrik ini. Jika itu masalahnya, lakukan evaluasi pada pola beban kerja Anda selama alarm dipicu. Hal ini dapat membantu Anda dalam mengidentifikasi apakah Anda dapat mengoptimalkan beban kerja kami untuk mengurangi jumlah total lalu lintas jaringan. Jika ini tidak memungkinkan, maka Anda mungkin perlu mempertimbangkan untuk menskalakan instans Anda.

Maksud: Alarm ini digunakan untuk mendeteksi throughput jaringan penyimpanan yang tinggi. Mendeteksi throughput yang tinggi dapat mencegah terjadinya penurunan paket jaringan dan penurunan performa.

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda harus menghitung sekitar 80%-90% dari total bandwidth jaringan tipe instans EC2, dan kemudian menggunakan hasil itu sebagai nilai ambang batas untuk secara proaktif mengambil tindakan sebelum paket jaringan terpengaruh. Anda juga dapat meninjau kekritisan dan persyaratan throughput jaringan penyimpanan dan melakukan analisis terhadap perilaku historis metrik ini untuk menentukan tingkat ambang batas yang masuk akal.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

Amazon Route 53 Public Data Plane

HealthCheckStatus

Dimensi: HealthCheckId

Deskripsi alarm: Alarm ini akan membantu Anda dalam mendeteksi titik akhir yang sedang dalam kondisi tidak sehat sesuai pemeriksa kondisi. Untuk memahami alasan kegagalan yang mengakibatkan status tidak sehat tersebut, Anda perlu menggunakan tab Pemeriksaan Kondisi di Konsol Pemeriksaan Kondisi Route 53 untuk melihat status dari masing-masing Wilayah serta kegagalan pemeriksaan kondisi terakhir yang mengalami kegagalan. Tab status tersebut juga dapat menampilkan alasan yang mengakibatkan titik akhir dilaporkan tidak sehat. Silakan lihat langkah-langkah pemecahan masalah.

Maksud: Alarm ini menggunakan pemeriksa kondisi Route53 untuk mendeteksi titik akhir yang sedang dalam kondisi tidak sehat.

Statistik: Rata-rata

Ambang batas yang disarankan: 1,0

Pembenaran ambang batas: Status titik akhir dilaporkan dengan nilai 1 saat sedang dalam kondisi yang sehat. Jika nilainya kurang dari 1, berarti kondisinya tidak sehat.

Periode: 60

Titik data untuk alarm: 3

Periode evaluasi: 3

Operator Perbandingan: LESS_THAN_THRESHOLD

Amazon S3

4xxErrors

Dimensi:BucketName, FilterId

Deskripsi alarm: Alarm ini akan membantu kami dalam melaporkan jumlah total kode status kesalahan 4xx yang dibuat sebagai respons atas permintaan klien. Kode kesalahan 403 mungkin

menunjukkan adanya kebijakan IAM yang salah, dan kode kesalahan 404 mungkin menunjukkan bahwa aplikasi klien yang berperilaku salah, begitu contohnya. Mengaktifkan pencatatan log akses server S3 sementara waktu akan membantu Anda dalam menentukan dari mana asal masalah yang terjadi dengan menggunakan bidang status HTTP dan Kode Kesalahan. Agar Anda dapat lebih memahami tentang kode kesalahan tersebut, silakan lihat Respons Kesalahan.

Maksud: Alarm ini akan digunakan untuk membuat garis dasar untuk tingkat kesalahan 4xx tipikal sehingga Anda dapat melihat kelainan apa pun yang terjadi yang mungkin menjadi indikasi dari sebuah masalah penyiapan.

Statistik: Rata-rata

Ambang batas yang disarankan: 0,05

Pembenaran ambang batas: Ambang batas yang disarankan adalah untuk mendeteksi apakah lebih dari 5% dari total permintaan telah mendapatkan kesalahan 4XX. Kesalahan 4XX yang sering terjadi harus Anda waspadai. Namun demikian, jika Anda menetapkan nilai yang sangat rendah untuk ambang batas ini, hal itu akan dapat menyebabkan alarm menjadi terlalu sensitif. Anda juga dapat menyetel ambang batas ini agar sesuai dengan beban permintaan, dengan memperhitungkan tingkat kesalahan 4XX yang dapat diterima. Anda juga dapat menganalisis data historis untuk menemukan tingkat kesalahan yang dapat diterima untuk beban kerja aplikasi, dan kemudian Anda dapat menyetel ambang batas yang sesuai.

Periode: 60

Titik data untuk alarm: 15

Periode evaluasi: 15

Operator Perbandingan: GREATER_THAN_THRESHOLD

5xxErrors

Dimensi:BucketName, FilterId

Deskripsi alarm: Alarm ini akan membantu Anda dalam mendeteksi sejumlah besar kesalahan yang terjadi pada sisi server. Kesalahan ini menunjukkan bahwa ada klien yang sudah membuat permintaan dan tidak dapat diselesaikan oleh server. Hal ini dapat membantu Anda untuk mengkorelasikan masalah yang dihadapi aplikasi Anda karena S3. Untuk informasi selengkapnya guna membantu Anda menangani atau mengurangi kejadian kesalahan secara efisien, silakan

lihat Mengoptimalkan pola desain performa. Kesalahan mungkin juga disebabkan oleh masalah yang terjadi pada S3, Anda perlu memeriksa dasbor kesehatan layanan AWS untuk mengetahui status Amazon S3 di Wilayah Anda.

Maksud: Alarm ini dapat membantu Anda dalam mendeteksi apakah aplikasi sedang mengalami masalah yang diakibatkan oleh kesalahan 5xx.

Statistik: Rata-rata

Ambang batas yang disarankan: 0,05

Pembenaran ambang batas: Kami menyarankan agar pengaturan ambang batas dilakukan untuk mendeteksi apakah lebih dari 5% dari total permintaan mendapatkan 5XXError, atau tidak. Namun demikian, Anda dapat menyetel ambang batas tersebut agar sesuai dengan lalu lintas permintaan, dan sesuai dengan tingkat kesalahan yang dapat diterima. Anda juga dapat menganalisis data historis untuk mengetahui tingkat kesalahan yang dapat diterima untuk beban kerja aplikasi, dan kemudian Anda dapat menyetel ambang batas sesuai dengan data historis itu.

Periode: 60

Titik data untuk alarm: 15

Periode evaluasi: 15

Operator Perbandingan: GREATER_THAN_THRESHOLD

OperationsFailedReplication

Dimensi:SourceBucket, DestinationBucket, RuleId

Deskripsi alarm: Alarm ini akan membantu Anda dalam memahami terjadinya kegagalan replikasi. Metrik ini akan melacak status objek baru yang direplikasi dengan menggunakan S3 CRR atau S3 SRR, dan juga melacak objek yang ada yang direplikasi dengan menggunakan replikasi batch S3. Silakan lihat Pemecahan masalah replikasi untuk mendapatkan informasi selengkapnya.

Maksud: Alarm ini digunakan untuk mendeteksi apakah ada operasi replikasi yang mengalami kegagalan.

Statistik: Maksimum

Ambang batas yang disarankan: 0,0

Pembenaran ambang batas: Metrik ini memancarkan nilai 0 untuk operasi yang berhasil dilakukan, dan tidak memancarkan apa-apa ketika tidak ada operasi replikasi yang dilakukan dalam satu menit. Ketika metrik memancarkan nilai yang lebih besar dari 0, artinya operasi replikasi yang dilakukan tidak berhasil.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

S3ObjectLambda

4xxErrors

Dimensi:AccessPointName, DataSource ARN

Deskripsi alarm: Alarm ini akan membantu kami melaporkan jumlah total kode status kesalahan 4xx yang dibuat sebagai respons atas permintaan klien. Mengaktifkan pencatatan log akses server S3 sementara waktu akan membantu Anda dalam menentukan dari mana asal masalah yang terjadi dengan menggunakan bidang status HTTP dan Kode Kesalahan.

Maksud: Alarm ini akan digunakan untuk membuat garis dasar untuk tingkat kesalahan 4xx tipikal sehingga Anda dapat melihat kelainan apa pun yang terjadi yang mungkin menjadi indikasi dari sebuah masalah penyiapan.

Statistik: Rata-rata

Ambang batas yang disarankan: 0,05

Pembenaran ambang batas: Kami menyarankan agar pengaturan ambang batas dilakukan untuk mendeteksi apakah lebih dari 5% dari total permintaan mendapatkan 4xxError, atau tidak. Kesalahan 4XX yang sering terjadi harus Anda waspadai. Namun demikian, jika Anda menetapkan nilai yang sangat rendah untuk ambang batas ini, hal itu akan dapat menyebabkan alarm menjadi terlalu sensitif. Anda juga dapat menyetel ambang batas ini agar sesuai dengan beban permintaan, dengan memperhitungkan tingkat kesalahan 4XX yang dapat diterima. Anda juga dapat menganalisis data historis untuk menemukan tingkat kesalahan yang dapat diterima untuk beban kerja aplikasi, dan kemudian Anda dapat menyetel ambang batas yang sesuai.

Periode: 60

Titik data untuk alarm: 15

Periode evaluasi: 15

Operator Perbandingan: GREATER_THAN_THRESHOLD

5xxErrors

Dimensi:AccessPointName, DataSource ARN

Deskripsi alarm: Alarm ini akan membantu Anda mendeteksi sejumlah besar kesalahan yang terjadi pada sisi server. Kesalahan ini menunjukkan bahwa ada klien yang sudah membuat permintaan dan tidak dapat diselesaikan oleh server. Kesalahan-kesalahan ini mungkin juga disebabkan oleh masalah yang terjadi pada S3, jadi Anda perlu memeriksa dasbor kesehatan layanan AWS untuk mengetahui status Amazon S3 di Wilayah Anda. Hal ini dapat membantu Anda untuk mengkorelasikan masalah yang dihadapi aplikasi Anda karena S3. Untuk informasi selengkapnya guna membantu Anda menangani atau mengurangi kejadian-kejadian kesalahan ini secara efisien, silakan lihat Mengoptimalkan pola desain performa.

Maksud: Alarm ini dapat membantu Anda dalam mendeteksi apakah aplikasi sedang mengalami masalah yang diakibatkan oleh kesalahan 5xx.

Statistik: Rata-rata

Ambang batas yang disarankan: 0,05

Pembenaran ambang batas: Kami menyarankan agar pengaturan ambang batas dilakukan untuk mendeteksi apakah lebih dari 5% dari total permintaan mendapatkan kesalahan 5XX, atau tidak. Namun demikian, Anda dapat menyetel ambang batas tersebut agar sesuai dengan lalu lintas permintaan, dan sesuai dengan tingkat kesalahan yang dapat diterima. Anda juga dapat menganalisis data historis untuk mengetahui tingkat kesalahan yang dapat diterima untuk beban kerja aplikasi, dan kemudian Anda dapat menyetel ambang batas sesuai dengan data historis itu.

Periode: 60

Titik data untuk alarm: 15

Periode evaluasi: 15

Operator Perbandingan: GREATER_THAN_THRESHOLD

LambdaResponse4xx

Dimensi:AccessPointName, DataSource ARN

Deskripsi alarm: Alarm ini akan membantu Anda dalam mendeteksi dan melakukan diagnosis kegagalan (500s) dalam panggilan ke S3 Object Lambda. Kesalahan-kesalahan ini dapat disebabkan oleh terjadinya kesalahan atau salah konfigurasi dalam fungsi Lambda yang bertanggung jawab untuk memberikan respons atas permintaan Anda. Menyelidiki Aliran CloudWatch Log dari fungsi Lambda yang terkait dengan Object Lambda Access Point dapat membantu Anda menentukan asal masalah berdasarkan respons dari S3 Object Lambda.

Maksud: Alarm ini digunakan untuk mendeteksi kesalahan klien 4xx untuk WriteGetObjectResponse panggilan.

Statistik: Rata-rata

Ambang batas yang disarankan: 0,05

Pembenaran ambang batas: Kami menyarankan agar pengaturan ambang batas dilakukan untuk mendeteksi apakah lebih dari 5% dari total permintaan mendapatkan 4xxError, atau tidak. Kesalahan 4XX yang sering terjadi harus Anda waspadai. Namun demikian, jika Anda menetapkan nilai yang sangat rendah untuk ambang batas ini, hal itu akan dapat menyebabkan alarm menjadi terlalu sensitif. Anda juga dapat menyetel ambang batas ini agar sesuai dengan beban permintaan, dengan memperhitungkan tingkat kesalahan 4XX yang dapat diterima. Anda juga dapat menganalisis data historis untuk menemukan tingkat kesalahan yang dapat diterima untuk beban kerja aplikasi, dan kemudian Anda dapat menyetel ambang batas yang sesuai.

Periode: 60

Titik data untuk alarm: 15

Periode evaluasi: 15

Operator Perbandingan: GREATER_THAN_THRESHOLD

Amazon SNS

NumberOfMessagesPublished

Dimensi: TopicName

Deskripsi alarm: Alarm ini dapat mendeteksi apakah jumlah pesan SNS yang diterbitkan terlalu rendah, atau tidak. Untuk pemecahan masalah, Anda perlu memeriksa penyebab yang mengakibatkan penerbit mengirim lalu lintas dalam jumlah lebih sedikit.

Maksud: Alarm ini akan membantu Anda secara proaktif memantau dan mendeteksi terjadinya penurunan yang signifikan dalam penerbitan notifikasi. Hal ini akan membantu Anda dalam mengidentifikasi masalah yang mungkin terjadi dengan aplikasi atau proses bisnis Anda, sehingga Anda dapat mengambil tindakan-tindakan yang tepat untuk mempertahankan aliran notifikasi yang Anda harapkan. Anda harus membuat alarm ini jika Anda memperkirakan bahwa sistem Anda akan melayani lalu lintas dengan jumlah minimum.

Statistik: Jumlah

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Jumlah pesan yang diterbitkan harus sesuai dengan perkiraan jumlah pesan yang dipublikasikan untuk aplikasi Anda. Anda juga dapat melakukan analisis data historis, tren, dan lalu lintas untuk menemukan ambang batas yang sesuai.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: LESS_THAN_THRESHOLD

NumberOfNotificationsDelivered

Dimensi: TopicName

Deskripsi alarm: Alarm ini dapat mendeteksi apakah jumlah pesan SNS yang dikirim terlalu rendah, atau tidak. Hal ini bisa jadi karena terjadinya penghentian berlangganan titik akhir yang dilakukan secara tidak disengaja, atau karena ada peristiwa SNS yang menyebabkan pesan mengalami penundaan.

Maksud: Alarm ini akan membantu Anda mendeteksi penurunan volume pesan yang dikirimkan. Anda harus membuat alarm ini jika Anda memperkirakan bahwa sistem Anda akan melayani lalu lintas dengan jumlah minimum.

Statistik: Jumlah

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Jumlah pesan yang dikirim harus sejalan dengan perkiraan jumlah pesan yang dihasilkan dan jumlah konsumen. Anda juga dapat melakukan analisis data historis, tren, dan lalu lintas untuk menemukan ambang batas yang sesuai.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: LESS_THAN_THRESHOLD

NumberOfNotificationsFailed

Dimensi: TopicName

Deskripsi alarm: Alarm ini dapat mendeteksi apakah jumlah pesan SNS yang gagal sudah terlalu tinggi, atau tidak. Untuk memecahkan masalah pemberitahuan yang gagal, aktifkan pencatatan ke CloudWatch Log. Dengan melakukan pemeriksaan log, hal itu dapat membantu Anda menemukan pelanggan mana yang mengalami kegagalan, dan menemukan kode status yang mereka kembalikan.

Maksud: Alarm ini akan membantu Anda secara proaktif menemukan masalah yang terjadi dengan pengiriman notifikasi dan mengambil tindakan yang tepat untuk mengatasi masalah ini.

Statistik: Jumlah

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Nilai ambang batas yang disarankan untuk alarm ini sangat bergantung pada dampak notifikasi yang mengalami kegagalan. Anda juga perlu meninjau SLA yang diberikan kepada pengguna akhir Anda, toleransi kesalahan dan kekritisan notifikasi dan analisis data historis, dan kemudian memilih ambang batas yang sesuai. Jumlah notifikasi yang mengalami kegagalan harus 0 untuk topik yang hanya memiliki langganan SQS, Lambda, atau Firehose.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

NumberOfNotificationsFilteredOut-InvalidAttributes

Dimensi: TopicName

Deskripsi alarm: Alarm ini akan membantu Anda dalam memantau dan menyelesaikan masalah-masalah yang mungkin terjadi terhadap penerbit atau pelanggan. Anda perlu memeriksa apakah penerbit menerbitkan pesan dengan atribut yang tidak valid atau apakah filter yang tidak pantas diterapkan ke pelanggan. Anda juga dapat menganalisis CloudWatch Log untuk membantu menemukan akar penyebab masalah.

Maksud: Alarm ini digunakan untuk mendeteksi apakah pesan yang dipublikasikan tidak valid atau apakah filter yang tidak pantas telah diterapkan ke pelanggan.

Statistik: Jumlah

Ambang batas yang disarankan: 0,0

Pembenaran ambang batas: Atribut yang tidak valid, hampir selalu menjadi kesalahan yang diakibatkan oleh penerbit. Kami menyarankan Anda untuk menetapkan ambang batasnya dengan nilai 0 karena atribut yang tidak valid tidak diharapkan ada dalam sistem yang sehat.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

NumberOfNotificationsFilteredOut-InvalidMessageBody

Dimensi: TopicName

Deskripsi alarm: Alarm ini akan membantu Anda dalam memantau dan menyelesaikan masalah-masalah yang mungkin terjadi terhadap penerbit atau pelanggan. Anda perlu memeriksa apakah penerbit menerbitkan sebuah pesan yang memiliki badan pesan yang tidak valid, atau apakah filter yang tidak pantas telah diterapkan ke pelanggan. Anda juga dapat menganalisis CloudWatch Log untuk membantu menemukan akar penyebab masalah.

Maksud: Alarm ini digunakan untuk mendeteksi apakah pesan yang dipublikasikan tidak valid atau apakah filter yang tidak pantas telah diterapkan ke pelanggan.

Statistik: Jumlah

Ambang batas yang disarankan: 0,0

Pembenaran ambang batas: Badan pesan yang tidak valid biasanya merupakan kesalahan yang diakibatkan oleh penerbit. Kami menyarankan Anda untuk menetapkan ambang batasnya dengan nilai 0 karena badan pesan yang tidak valid tidak diharapkan ada dalam sistem yang sehat.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

NumberOfNotificationsRedrivenToDlq

Dimensi: TopicName

Deskripsi alarm: Alarm ini akan membantu Anda memantau jumlah pesan yang sudah dipindahkan ke antrean surat mati.

Maksud: Alarm ini digunakan untuk mendeteksi pesan-pesan yang sudah dipindahkan ke antrean surat mati. Kami menyarankan Anda agar Anda membuat alarm ini ketika SNS digabungkan dengan SQS, Lambda atau Firehose.

Statistik: Jumlah

Ambang batas yang disarankan: 0,0

Pembenaran ambang batas: Dalam sebuah sistem yang sehat yang terdiri dari semua jenis pelanggan, pesan tidak boleh dipindahkan ke antrean surat mati. Kami menyarankan Anda agar Anda mendapatkan notifikasi jika ada pesan yang masuk ke dalam antrean, sehingga Anda dapat mengidentifikasi dan mengatasi akar penyebabnya, dan hal ini berpotensi mengarahkan ulang pesan-pesan yang ada dalam antrean surat mati untuk mencegah terjadinya kehilangan data.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

NumberOfNotificationsFailedToRedriveToDlq

Dimensi: TopicName

Deskripsi alarm: Alarm ini akan membantu Anda dalam memantau pesan-pesan yang tidak dapat dipindahkan ke antrean surat mati. Anda perlu memeriksa apakah antrean surat mati Anda ada dan apakah hal itu sudah dikonfigurasi dengan benar. Selain itu, Anda juga perlu memverifikasi bahwa SNS memiliki izin untuk mengakses antrean surat mati. Silakan lihat dokumentasi antrean surat mati untuk mempelajari hal ini lebih lanjut.

Maksud: Alarm ini digunakan untuk mendeteksi pesan-pesan yang tidak dapat dipindahkan ke antrean surat mati.

Statistik: Jumlah

Ambang batas yang disarankan: 0,0

Pembenaran ambang batas: Hampir selalu menjadi kesalahan jika pesan tidak dapat dipindahkan ke antrean surat mati. Disarankan untuk mengatur ambang batasnya dengan 0, yang berarti semua pesan yang mengalami kegagalan dalam prosesnya harus dapat mencapai antrean surat mati ketika antrean telah dikonfigurasi.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

SMS MonthToDateSpent USD

Dimensi: TopicName

Deskripsi alarm: Alarm ini akan membantu Anda memantau apakah Anda memiliki kuota yang cukup pada akun Anda sehingga SNS dapat mengirimkan pesan. Jika Anda sudah mencapai kuota Anda, maka SNS tidak akan dapat mengirimkan pesan SMS. Untuk informasi tentang pengaturan kuota belanja SMS bulanan Anda, atau untuk informasi tentang meminta peningkatan kuota belanja dengan AWS, lihat Menyetel preferensi pesan SMS.

Maksud: Alarm ini digunakan untuk mendeteksi apakah Anda memiliki kuota yang cukup yang ada di akun Anda agar pesan-pesan SMS Anda bisa dikirim dengan sukses.

Statistik: Maksimum

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda perlu menetapkan ambang batasnya sesuai dengan kuota (Batas pengeluaran akun) untuk akun. Anda harus memilih ambang batas yang memberi tahu Anda cukup awal apakah Anda telah mencapai batas kuota Anda sehingga Anda punya waktu untuk meminta dilakukan kenaikan.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

SMS SuccessRate

Dimensi: TopicName

Deskripsi alarm: Alarm ini akan membantu Anda memantau tingkat kegagalan pengiriman pesan SMS. Anda dapat mengatur Cloudwatch Logs untuk memahami sifat yang ada pada kegagalan dan mengambil tindakan berdasarkan hal itu.

Maksud: Alarm ini digunakan untuk mendeteksi kejadian gagalnya pengiriman pesan SMS.

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Anda perlu menetapkan ambang batas untuk alarm ini sesuai dengan toleransi Anda untuk pengiriman pesan SMS yang mengalami kegagalan.

Periode: 60

Titik data untuk alarm: 5

Periode evaluasi: 5

Operator Perbandingan: GREATER_THAN_THRESHOLD

Amazon SQS

ApproximateAgeOfOldestMessage

Dimensi: QueueName

Deskripsi alarm: Alarm ini mengawasi usia pesan tertua yang ada dalam antrean. Anda dapat menggunakan alarm ini untuk memantau apakah para pelanggan Anda memproses pesan SQS dengan kecepatan yang diinginkan. Anda juga perlu mempertimbangkan untuk meningkatkan jumlah konsumen atau throughput konsumen untuk mengurangi usia pesan. Metrik ini dapat digunakan dengan dikombinasikan dengan `ApproximateNumberOfMessagesVisible` untuk menentukan seberapa besar antrean backlog dan seberapa cepat pesan yang sedang diproses. Untuk mencegah pesan dihapus sebelum diproses, Anda perlu mempertimbangkan untuk mengonfigurasi antrean surat mati untuk mengesampingkan pesan pil racun yang mungkin terjadi.

Maksud: Alarm ini digunakan untuk mendeteksi apakah usia pesan tertua dalam QueueName antrian terlalu tinggi. Usia yang tinggi dapat menjadi indikasi bahwa pesan tidak diproses dengan cukup cepat atau bahwa ada beberapa pesan pil racun yang terjebak dalam antrean dan tidak dapat diproses.

Statistik: Maksimum

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Nilai ambang batas yang disarankan untuk alarm ini sangat bergantung pada waktu pemrosesan pesan yang Anda harapkan. Anda dapat menggunakan data historis untuk menghitung waktu pemrosesan pesan rata-rata, dan kemudian menetapkan ambang batasnya menjadi 50% lebih tinggi dari waktu pemrosesan pesan SQS maksimum yang diharapkan oleh konsumen antrean.

Periode: 60

Titik data untuk alarm: 15

Periode evaluasi: 15

Operator Perbandingan: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

ApproximateNumberOfMessagesNotVisible

Dimensi: QueueName

Deskripsi alarm: Alarm ini akan membantu Anda mendeteksi sejumlah besar pesan yang sedang berjalan sehubungan dengan `QueueName`. Untuk pemecahan masalah, Anda perlu memeriksa penurunan backlog pesan.

Maksud: Alarm ini digunakan untuk mendeteksi sejumlah besar pesan sedang berjalan yang ada dalam antrean. Jika konsumen tidak menghapus pesan-pesan yang ada dalam periode batas waktu visibilitas, ketika antrean disurvei, pesan-pesan tersebut akan muncul kembali dalam antrean. Untuk antrean FIFO, bisa ada maksimal 20.000 pesan sedang berjalan. Jika Anda mencapai kuota ini, maka SQS tidak akan mengembalikan pesan kesalahan. Antrean FIFO memeriksa 20k pesan pertama untuk menentukan grup pesan yang tersedia. Hal ini artinya bahwa jika Anda memiliki backlog pesan dalam satu grup pesan, maka Anda tidak akan dapat menggunakan pesan-pesan dari grup pesan lain yang dikirim ke antrean di lain waktu hingga Anda berhasil mengkonsumsi pesan-pesan dari backlog tersebut.

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Nilai ambang batas yang disarankan untuk alarm ini sangat tergantung pada perkiraan jumlah pesan sedang berjalan. Anda dapat menggunakan data historis untuk menghitung perkiraan jumlah pesan maksimum sedang berjalan dan menetapkan ambang batasnya menjadi 50% di atas nilai ini. Jika konsumen antrean sedang memproses tetapi tidak menghapus pesan-pesan dari antrean tersebut, maka jumlah ini akan meningkat secara tiba-tiba.

Periode: 60

Titik data untuk alarm: 15

Periode evaluasi: 15

Operator Perbandingan: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

ApproximateNumberOfMessagesVisible

Dimensi: QueueName

Deskripsi alarm: Alarm ini akan mengawasi backlog antrean pesan yang menjadi lebih besar dari yang diharapkan, yang mana hal ini menunjukkan bahwa konsumen terlalu lambat atau jumlah konsumen tidak memadai. Anda perlu mempertimbangkan untuk menaikkan jumlah konsumen atau mempercepat konsumen Anda, jika alarm ini beralih statusnya menjadi ALARM.

Maksud: Alarm ini digunakan untuk mendeteksi apakah jumlah pesan dari antrean aktif terlalu tinggi dan apakah konsumen memproses pesan dengan lambat atau tidak tersedia jumlah konsumen yang memadai untuk memproses pesan-pesan tersebut.

Statistik: Rata-rata

Ambang batas yang disarankan: Tergantung pada situasi Anda

Pembenaran ambang batas: Jumlah pesan yang terlihat sangat tinggi menunjukkan bahwa pesan-pesan tersebut tidak diproses oleh konsumen dengan kecepatan yang diharapkan. Anda juga perlu mempertimbangkan data historis saat Anda menetapkan ambang batas ini.

Periode: 60

Titik data untuk alarm: 15

Periode evaluasi: 15

Operator Perbandingan: GREATER_THAN_OR_EQUAL_TO_THRESHOLD

NumberOfMessagesSent

Dimensi: QueueName

Deskripsi alarm: Alarm ini akan membantu Anda mendeteksi apakah tidak ada pesan yang dikirim dari produsen sehubungan dengan `QueueName`. Untuk pemecahan masalah, Anda perlu memeriksa alasan yang menyebabkan produsen tersebut tidak mengirim pesan.

Maksud: Alarm ini digunakan untuk mendeteksi ketika produsen berhenti mengirim pesan.

Statistik: Jumlah

Ambang batas yang disarankan: 0,0

Pembenaran ambang batas: Jika jumlah pesan yang dikirim adalah 0, artinya produsen tidak mengirim pesan apa pun. Jika antrian ini memiliki TPS rendah, tingkatkan jumlahnya.
EvaluationPeriods

Periode: 60

Titik data untuk alarm: 15

Periode evaluasi: 15

Operator Perbandingan: LESS_THAN_OR_EQUAL_TO_THRESHOLD

AWS VPN

TunnelState

Dimensi: VpnId

Deskripsi alarm: Alarm ini akan membantu Anda memahami apakah satu atau beberapa terowongan sedang berada dalam status DOWN. Untuk memecahkan masalah ini, silakan lihat Pemecahan masalah terowongan VPN.

Maksud: Alarm ini digunakan untuk mendeteksi jika setidaknya ada satu terowongan yang statusnya beralih menjadi DOWN untuk VPN ini, sehingga Anda dapat memecahkan masalah yang dialami oleh VPN yang terkena dampaknya. Alarm ini akan selalu berada dalam status ALARM untuk jaringan yang dikonfigurasi untuk memiliki hanya satu terowongan.

Statistik: Minimum

Ambang batas yang disarankan: 1,0

Pembenaran ambang batas: Nilai ambang batas yang kurang dari 1 menunjukkan bahwa setidaknya ada satu terowongan yang statusnya DOWN.

Periode: 300

Titik data untuk alarm: 3

Periode evaluasi: 3

Operator Perbandingan: LESS_THAN_THRESHOLD

TunnelState

Dimensi: TunnelIpAddress

Deskripsi alarm: Alarm ini akan membantu Anda mengetahui jika terowongan ini berada dalam status DOWN. Untuk memecahkan masalah ini, silakan lihat Pemecahan masalah terowongan VPN.

Maksud: Alarm ini digunakan untuk mendeteksi apakah terowongan dalam status DOWN atau tidak, sehingga Anda dapat menyelesaikan masalah-masalah yang dialami VPN yang terkena

dampaknya. Alarm ini akan selalu berada dalam status ALARM untuk jaringan yang dikonfigurasi untuk memiliki hanya satu terowongan.

Statistik: Minimum

Ambang batas yang disarankan: 1,0

Pembenaran ambang batas: Nilai ambang batas yang kurang dari 1 menunjukkan bahwa terowongan sedang berada dalam status DOWN.

Periode: 300

Titik data untuk alarm: 3

Periode evaluasi: 3

Operator Perbandingan: LESS_THAN_THRESHOLD

Membuat alarm untuk metrik

Langkah-langkah di bagian berikut menjelaskan cara membuat CloudWatch alarm pada metrik.

Buat CloudWatch alarm berdasarkan ambang statis

Anda memilih CloudWatch metrik untuk menonton alarm, dan ambang batas untuk metrik itu. Alarm tersebut statusnya akan beralih menjadi status ALARM saat metrik melanggar ambang batas untuk jumlah periode evaluasi yang ditentukan.

Jika Anda membuat alarm di akun yang diatur sebagai akun pemantauan dalam pengamatan CloudWatch lintas akun, Anda dapat mengatur alarm untuk menonton metrik di akun sumber yang ditautkan ke akun pemantauan ini. Untuk informasi selengkapnya, lihat CloudWatch observabilitas lintas akun.

Cara membuat sebuah alarm berdasarkan sebuah metrik tunggal

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, pilih Alarm, Semua alarm.
3. Pilih Buat alarm.
4. Pilih Pilih Metrik.
5. Lakukan salah satu langkah berikut:

- Anda harus memilih namespace layanan yang memuat metrik yang Anda inginkan. Kemudian Anda harus memilih pilihan yang tampaknya akan mempersempit pilihan. Ketika ada daftar metrik yang muncul, pilihlah kotak centang yang ada di samping metrik yang Anda inginkan.
- Pada kotak pencarian, masukkan nama metrik, ID akun, label akun, dimensi, atau ID sumber daya. Kemudian, Anda harus memilih salah satu hasil pencarian dan melanjutkan hingga daftar metrik muncul. Pilihlah kotak centang yang ada di samping metrik yang Anda inginkan.

6. Pilih tab Metrik bergrafik.

   a. Pada Statistik, pilih salah satu statistik atau persentil yang telah ditentukan sebelumnya, atau masukkan persentil yang Anda kehendaki (misalnya, **`p95.45`**).

   b. Pada Periode, silakan pilih periode evaluasi untuk alarm tersebut. Saat Anda melakukan evaluasi alarm, masing-masing periode akan digabungkan menjadi satu titik data.

      Anda juga dapat memilih apakah keterangan sumbu y muncul di bagian kiri atau kanan ketika Anda membuat alarm. Pilihan ini hanya bisa digunakan ketika Anda sedang membuat alarm.

   c. Pilih Pilih Metrik.

      Halaman Tentukan metrik dan ketentuan ditampilkan, di sana ditampilkan grafik dan informasi lain tentang metrik dan statistik yang Anda pilih.

7. Pada Ketentuan, tentukan hal-hal berikut:

   a. Setiap kali ***`metrik`*** diukur, tentukan apakah metrik harus lebih besar dari, kurang dari, atau sama dengan ambang batas. Di bawah dari..., tentukan nilai ambang batas.

   b. Pilih Konfigurasi tambahan. Untuk Titik data alarm, tentukan berapa banyak periode evaluasi (titik data) yang harus ada dalam status `ALARM` untuk memicu alarm. Jika kedua nilai di sini cocok, Anda membuat alarm yang beralih ke status `ALARM` jika terjadi pelanggaran selama sebanyak itu dalam periode berturut-turut.

      Untuk membuat alarm M dari N, tentukan angka yang lebih rendah untuk nilai pertama dibandingkan dengan nilai yang Anda tentukan untuk nilai kedua. Untuk informasi selengkapnya, lihat Melakukan evaluasi alarm.

   c. Untuk Perlakuan data yang hilang, pilih cara alarm berperilaku ketika beberapa titik data hilang. Untuk informasi selengkapnya, lihat Mengkonfigurasi bagaimana CloudWatch alarm memperlakukan data yang hilang.

d. Jika alarm menggunakan sebuah persentil sebagai statistik yang dipantau, maka kotak Persentil dengan sampel rendah akan ditampilkan. Pilih apakah Anda akan melakukan evaluasi atau mengabaikan kasus yang memiliki tingkat sampel yang kecil. Jika Anda memilih abaikan (status alarm tidak berubah), maka status alarm saat ini akan tetap dipertahankan ketika ukuran sampel terlalu kecil. Untuk informasi selengkapnya, lihat CloudWatch Alarm berbasis persentil dan sampel data rendah.

8. Pilih Berikutnya.
9. Pada Notifikasi, silakan pilih topik SNS yang akan diberitahukan sebagai notifikasi ketika alarm berada dalam status `ALARM`, status `OK`, atau status `INSUFFICIENT_DATA`.

   Agar alarm mengirimkan beberapa notifikasi untuk status alarm yang sama atau untuk status alarm yang berbeda, silakan pilih Tambahkan notifikasi.

   Dalam pengamatan CloudWatch lintas akun, Anda dapat memilih untuk mengirimkan notifikasi ke beberapa AWS akun. Sebagai contoh, ke akun pemantauan dan akun sumber.

   Agar alarm tidak mengirim notifikasi, silakan pilih Hapus.
10. Agar alarm dapat melakukan tindakan penskalaan otomatis, EC2, Lambda atau Systems Manager, pilih tombol yang sesuai dan pilih status alarm serta tindakan yang harus dilakukan. Alarm dapat melakukan tindakan Systems Manager hanya ketika masuk ke status ALARM. Untuk informasi selengkapnya tentang tindakan Systems Manager, lihat Mengkonfigurasi CloudWatch untuk membuat OpsItems dari alarm dan pembuatan Insiden.

    > **Note**
    >
    > Untuk membuat alarm yang melakukan tindakan SSM Incident Manager, Anda harus memiliki izin tertentu. Untuk informasi selengkapnya, lihat contoh kebijakan berbasis identitas untuk Manajer Insiden AWS Systems Manager.

11. Setelah selesai, silakan pilih Berikutnya.
12. Masukkan sebuah nama dan deskripsi untuk alarm tersebut. Nama tersebut harus menggunakan karakter UTF-8, dan tidak dapat berisi karakter kontrol ASCII. Deskripsi dapat mencakup pemformatan penurunan harga, yang hanya ditampilkan di tab Detail alarm di CloudWatch konsol. Penurunan harga dapat Anda gunakan untuk menambahkan tautan ke runbook atau sumber daya internal lainnya. Lalu pilih Berikutnya.
13. Pada Pratinjau dan buat, konfirmasikan bahwa informasi dan ketentuannya sudah sesuai keinginan Anda, kemudian pilih Buat alarm.

Anda juga dapat menambahkan alarm ke sebuah dasbor. Untuk informasi selengkapnya, lihat Menambahkan atau menghapus widget alarm dari CloudWatch dasbor .

Buat CloudWatch alarm berdasarkan ekspresi matematika metrik

Untuk membuat alarm berdasarkan ekspresi matematika metrik, pilih satu atau beberapa CloudWatch metrik yang akan digunakan dalam ekspresi. Kemudian, Anda juga perlu menentukan periode ekspresi, ambang batas, dan evaluasi.

Anda tidak dapat membuat sebuah alarm berdasarkan ekspresi SEARCH. Hal ini karena ekspresi pencarian mengembalikan beberapa deret waktu, dan sebuah alarm yang dibuat berdasarkan ekspresi matematika hanya dapat mengawasi satu deret waktu.

Cara membuat sebuah alarm yang didasarkan pada ekspresi matematika metrik

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Alarm, dan kemudian pilih Semua alarm.
3. Pilih Buat alarm.
4. Pilih Pilih Metrik, dan kemudian lakukan salah satu tindakan berikut ini:
   - Pilih namespace dari menu geser-turun namespace AWS atau menu geser-turun namespace kustom. Setelah memilih namespace, Anda kemudian harus memilih opsi hingga daftar metrik muncul, di mana Anda dapat memilih kotak centang yang ada di samping metrik yang benar.
   - Anda bisa menggunakan kotak pencarian untuk menemukan metrik, ID akun, dimensi, atau ID sumber daya. Setelah memasukkan ID metrik, dimensi, atau sumber daya, Anda kemudian harus memilih opsi hingga ada daftar metrik yang muncul, di mana Anda bisa memilih kotak centang yang ada di samping metrik yang benar.
5. (Opsional) Jika Anda ingin menambahkan metrik lainnya pada ekspresi matematika metrik, maka Anda dapat menggunakan kotak pencarian untuk menemukan metrik tertentu. Anda dapat menambahkan sebanyak 10 metrik pada sebuah ekspresi matematika metrik.
6. Seleksi tab Metrik bergrafik. Untuk setiap metrik yang sebelumnya Anda tambahkan, Anda perlu melakukan tindakan-tindakan berikut ini:

   a. Pada kolom Statistik, silakan pilih menu geser-turun. Pada menu geser-turun, silakan pilih salah satu statistik atau persentil yang telah ditentukan sebelumnya. Gunakan kotak pencarian yang ada di menu geser-turun untuk menentukan persentil kustom.

b. Pada kolom Periode, pilih menu geser-turun. Pada menu geser-turun tersebut, Anda harus memilih salah satu periode evaluasi yang telah ditentukan sebelumnya.

   Saat membuat alarm, Anda juga dapat menentukan apakah legenda sumbu Y akan dimunculkan di sisi kiri atau kanan grafik Anda.

> **Note**
>
> Saat CloudWatch mengevaluasi alarm, periode digabungkan menjadi titik data tunggal.

7. Pilih menu geser-turun Tambahkan matematika, kemudian pilih Mulai dengan ekspresi kosong dari daftar ekspresi matematika metrik yang telah ditentukan sebelumnya.

   Setelah Anda memilih Mulai dengan ekspresi kosong, sebuah kotak ekspresi matematika akan ditampilkan di mana Anda bisa menerapkan atau mengedit ekspresi matematika di sana.

8. Pada kotak ekspresi matematika tersebut, Anda masukkan ekspresi matematika Anda, dan kemudian pilih Terapkan.

   Setelah Anda memilih Terapkan, kolom ID akan ditampilkan di samping kolom Label.

   Agar Anda bisa menggunakan metrik atau hasil ekspresi matematika metrik lain sebagai bagian dari rumus ekspresi matematika Anda saat ini, Anda harus menggunakan nilai yang ditampilkan di kolom ID. Untuk mengubah nilai ID, Anda memilih pen-and-paper ikon di sebelah nilai saat ini. Nilai baru yang Anda masukkan harus dimulai dengan huruf kecil dan dapat mencakup angka, huruf, dan simbol garis bawah. Dengan mengubah nilai ID menjadi sebuah nama yang lebih signifikan akan membuat grafik alarm Anda menjadi lebih mudah dipahami.

   Untuk informasi mengenai fungsi-fungsi yang tersedia untuk matematika metrik, silakan lihat Sintaks dan fungsi matematika metrik.

9. (Opsional) Anda bisa menambahkan lebih banyak ekspresi matematika, dengan menggunakan metrik dan hasil ekspresi matematika lainnya dalam rumus ekspresi matematika yang baru.

10. Ketika Anda memiliki ekspresi yang akan digunakan untuk alarm, Anda harus menghapus kotak centang yang ada di sebelah kiri ekspresi lain dan setiap metrik yang ada di halaman tersebut. Hanya kotak centang yang di samping ekspresi saja yang digunakan untuk alarm yang harus dipilih. Ekspresi yang Anda pilih untuk alarm tersebut harus menghasilkan satu rangkaian waktu dan hanya menampilkan satu garis pada grafik. Kemudian pilih Pilih metrik.

Halaman Tentukan metrik dan ketentuan ditampilkan, di sana akan ditampilkan grafik dan informasi lain tentang ekspresi matematika yang Anda pilih.

11. Untuk Setiap kali ***ekspresi*** adalah, Anda harus menentukan apakah metrik harus lebih besar dari, kurang dari, atau sama dengan ambang batas. Di bawah dari..., tentukan nilai ambang batas.
12. Pilih Konfigurasi tambahan. Untuk Titik data alarm, tentukan berapa banyak periode evaluasi (titik data) yang harus ada dalam status `ALARM` untuk memicu alarm. Jika kedua nilai di sini cocok, Anda membuat alarm yang beralih ke status `ALARM` jika terjadi pelanggaran selama sebanyak itu dalam periode berturut-turut.

    Untuk membuat alarm M dari N, tentukan angka yang lebih rendah untuk nilai pertama dibandingkan dengan nilai yang Anda tentukan untuk nilai kedua. Untuk informasi selengkapnya, lihat Melakukan evaluasi alarm.
13. Untuk Perlakuan data yang hilang, pilih cara alarm berperilaku ketika beberapa titik data hilang. Untuk informasi selengkapnya, lihat Mengkonfigurasi bagaimana CloudWatch alarm memperlakukan data yang hilang.
14. Pilih Berikutnya.
15. Pada Notifikasi, silakan pilih topik SNS yang akan diberitahukan sebagai notifikasi ketika alarm berada dalam status `ALARM`, status `OK`, atau status `INSUFFICIENT_DATA`.

    Agar alarm mengirimkan beberapa notifikasi untuk status alarm yang sama atau untuk status alarm yang berbeda, silakan pilih Tambahkan notifikasi.

    Agar alarm tidak mengirim notifikasi, silakan pilih Hapus.
16. Agar alarm dapat melakukan tindakan penskalaan otomatis, EC2, Lambda atau Systems Manager, pilih tombol yang sesuai dan pilih status alarm serta tindakan yang harus dilakukan. Jika Anda memilih sebuah fungsi Lambda sebagai tindakan alarm, maka Anda menentukan nama fungsi atau ARN, dan Anda dapat memilih versi tertentu dari fungsi tersebut secara opsional.

    Alarm dapat melakukan tindakan Systems Manager hanya ketika masuk ke status ALARM. Untuk informasi selengkapnya tentang tindakan Systems Manager, lihat Mengkonfigurasi CloudWatch untuk membuat OpsItems dari alarm dan pembuatan Insiden.

> **Note**
>
> Untuk membuat alarm yang melakukan tindakan SSM Incident Manager, Anda harus memiliki izin tertentu. Untuk informasi selengkapnya, lihat contoh kebijakan berbasis identitas untuk Manajer Insiden AWS Systems Manager.

17. Setelah selesai, silakan pilih Berikutnya.
18. Masukkan sebuah nama dan deskripsi untuk alarm tersebut. Lalu pilih Berikutnya.

    Nama tersebut harus menggunakan karakter UTF-8, dan tidak dapat berisi karakter kontrol ASCII. Deskripsi dapat mencakup pemformatan penurunan harga, yang hanya ditampilkan di tab Detail alarm di CloudWatch konsol. Penurunan harga dapat Anda gunakan untuk menambahkan tautan ke runbook atau sumber daya internal lainnya.
19. Pada Pratinjau dan buat, konfirmasikan bahwa informasi dan ketentuannya sudah sesuai keinginan Anda, kemudian pilih Buat alarm.

Anda juga dapat menambahkan alarm ke sebuah dasbor. Untuk informasi selengkapnya, lihat Menambahkan atau menghapus widget alarm dari CloudWatch dasbor .

Membuat CloudWatch alarm berdasarkan kueri Wawasan Metrik

Anda juga dapat membuat sebuah alarm berdasarkan kueri Wawasan Metrik apa pun yang menampilkan rangkaian waktu tunggal. Ini dapat sangat berguna untuk membuat alarm dinamis yang mengamati metrik agregat di seluruh armada infrastruktur atau aplikasi Anda. Setelah alarm dibuat, alarm tersebut akan secara dinamis menyesuaikan diri ketika ada sumber daya yang ditambahkan atau dihapus dari armada. Sebagai contoh, Anda dapat membuat sebuah alarm yang mengawasi ulitisasi CPU dari semua instans Anda, dan alarm tersebut secara dinamis akan menyesuaikan diri ketika Anda menambahkan atau menghapus instans.

Untuk instruksi selengkapnya, silakan lihat Membuat alarm pada kueri Metrics Insights.

Membuat sebuah alarm berdasarkan pada sumber data yang terhubung

Anda dapat membuat alarm yang melihat metrik dari sumber data yang tidak ada. CloudWatch Untuk informasi selengkapnya mengenai cara membuat koneksi ke sumber data lainnya, silakan lihat Metrik kueri dari sumber data lain.

Cara membuat sebuah alarm berdasarkan metrik dari sumber data yang telah Anda sambungkan

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik, Semua metrik.
3. Pilih tab Kueri multi sumber.
4. Untuk Sumber data, silakan pilih sumber data yang ingin Anda gunakan.
5. Pembuat kueri akan meminta Anda untuk memberikan informasi yang diperlukan untuk kueri agar bisa mengambil metrik yang akan digunakan untuk alarm. Alur kerjanya akan berbeda untuk masing-masing sumber data, dan akan disesuaikan dengan sumber data. Sebagai contoh, untuk Layanan Terkelola Amazon untuk Prometheus dan sumber data Prometheus, kotak editor kueri PromQL dengan pembantu kueri akan ditampilkan.
6. Setelah Anda selesai membuat konsep kueri tersebut, silakan pilih Buat grafik kueri.
7. Jika grafik sampel sudah terlihat seperti yang Anda harapkan, silakan pilih Buat alarm.
8. Kemudian akan muncul halaman Tentukan metrik dan kondisi. Jika kueri yang Anda gunakan menghasilkan lebih dari satu deret waktu, maka Anda akan muncul banner peringatan di bagian atas halaman. Jika banner peringatan itu muncul, silakan pilih fungsi yang akan digunakan untuk menggabungkan deret waktu dalam fungsi Agregasi.
9. (Opsional) Tambahkan sebuah Label untuk alarm.
10. Untuk ***your-metric-name***Kapan pun.. , pilih Lebih Besar, Lebih Hebat/Sama, Lebih Rendah/Sama, atau Lebih Rendah. Kemudian untuk dari . . ., masukkan angka untuk nilai ambang batas Anda.
11. Pilih Konfigurasi tambahan. Untuk Titik data alarm, tentukan berapa banyak periode evaluasi (titik data) yang harus ada dalam status `ALARM` untuk memicu alarm. Jika kedua nilai di sini cocok, Anda membuat alarm yang beralih ke status `ALARM` jika terjadi pelanggaran selama sebanyak itu dalam periode berturut-turut.

    Untuk membuat sebuah alarm M dari N, Anda harus menentukan angka untuk nilai pertama dengan nilai yang lebih rendah dari angka untuk nilai kedua. Untuk informasi selengkapnya, lihat Melakukan evaluasi alarm.
12. Untuk Perlakuan data yang hilang, pilih perilaku alarm ketika ada beberapa titik data yang hilang. Untuk informasi selengkapnya, lihat Mengkonfigurasi bagaimana CloudWatch alarm memperlakukan data yang hilang.
13. Pilih Berikutnya.
14. Untuk Notifikasi, Anda perlu menentukan topik Amazon SNS yang akan mendapatkan notifikasi saat alarm statusnya beralih menjadi `ALARM`, `OK` atau `INSUFFICIENT_DATA`.

a. (Opsional) Untuk mengirimkan beberapa notifikasi untuk status alarm yang sama atau status alarm yang berbeda, silakan pilih Tambahkan notifikasi.

> **Note**
>
> Kami menyarankan Anda untuk menyetel alarm untuk mengambil tindakan ketika alarm beralih statusnya menjadi data tidak mencukupi selain ketika beralih status menjadi Alarm. Hal ini dilakukan karena banyak masalah dengan fungsi Lambda yang terhubung ke sumber data yang dapat menyebabkan alarm beralih statusnya menjadi Data tidak mencukupi.

b. (Opsional) Jika tidak ingin mengirimkan notifikasi Amazon SNS, silakan pilih Hapus.

15. Agar alarm dapat melakukan tindakan penskalaan otomatis, EC2, Lambda atau Systems Manager, pilih tombol yang sesuai dan pilih status alarm serta tindakan yang harus dilakukan. Jika Anda memilih sebuah fungsi Lambda sebagai tindakan alarm, maka Anda menentukan nama fungsi atau ARN, dan Anda dapat memilih versi tertentu dari fungsi tersebut secara opsional.

    Alarm dapat melakukan tindakan Systems Manager hanya ketika masuk ke status ALARM. Untuk informasi selengkapnya tentang tindakan Systems Manager, lihat Mengkonfigurasi CloudWatch untuk membuat OpsItems dari alarm dan pembuatan Insiden.

    > **Note**
    >
    > Untuk membuat alarm yang melakukan tindakan SSM Incident Manager, Anda harus memiliki izin tertentu. Untuk informasi selengkapnya, lihat contoh kebijakan berbasis identitas untuk Manajer Insiden AWS Systems Manager.

16. Pilih Berikutnya.
17. Pada Nama dan deskripsi, Anda harus memasukkan nama dan deskripsi untuk alarm Anda, dan kemudian pilih Berikutnya. Nama tersebut harus menggunakan karakter UTF-8, dan tidak dapat berisi karakter kontrol ASCII. Deskripsi dapat mencakup pemformatan penurunan harga, yang hanya ditampilkan di tab Detail alarm di CloudWatch konsol. Penurunan harga dapat Anda gunakan untuk menambahkan tautan ke runbook atau sumber daya internal lainnya.

> **Tip**
>
> Nama alarm harus menggunakan karakter UTF-8 saja. Nama tersebut tidak boleh memuat karakter kontrol ASCII.

18. Pada Pratinjau dan buat, silakan Anda konfirmasi bahwa informasi dan kondisi alarm Anda sudah benar, dan kemudian pilih Buat alarm.

Detail mengenai alarm untuk sumber data yang terhubung

- Ketika CloudWatch mengevaluasi alarm, ia melakukannya setiap menit, bahkan jika periode untuk alarm lebih dari satu menit. Agar alarm berfungsi, fungsi Lambda harus dapat mengembalikan daftar stempel waktu yang dimulai setiap menit, tidak hanya pada kelipatan panjang periode. Stempel waktu ini harus diberi jarak satu periode.

  Oleh karena itu, jika sumber data yang dikueri oleh Lambda hanya dapat mengembalikan stempel waktu yang merupakan kelipatan dari panjang periode, maka fungsi tersebut harus "mengambil sampel ulang" data yang diambil agar sesuai dengan stempel waktu yang diharapkan oleh permintaan `GetMetricData`.

  Sebagai contoh, sebuah alarm dengan periode lima menit akan dievaluasi setiap menit dengan menggunakan jendela lima menit yang bergeser satu menit setiap kali. Dalam kasus ini:

  - Untuk evaluasi alarm pada 12:15:00, CloudWatch mengharapkan titik data dengan stempel waktu,, dan. `12:00:00` `12:05:00` `12:10:00`
  - Kemudian untuk evaluasi alarm pada 12:16:00, CloudWatch mengharapkan titik data dengan stempel waktu,, dan. `12:01:00` `12:06:00` `12:11:00`
- Saat CloudWatch mengevaluasi alarm, titik data apa pun yang dikembalikan oleh fungsi Lambda yang tidak sejajar dengan stempel waktu yang diharapkan akan dijatuhkan, dan alarm dievaluasi menggunakan titik data yang diharapkan yang tersisa. Sebagai contoh, ketika alarm dievaluasi pada `12:15:00`, itu akan mengharapkan data dengan stempel waktu `12:00:00`, `12:05:00`, dan `12:10:00`. Jika menerima data dengan stempel waktu`12:00:00`,,`12:05:00`, dan `12:06:0012:10:00`, data dari `12:06:00` dijatuhkan dan CloudWatch mengevaluasi alarm menggunakan stempel waktu lainnya.

  Kemudian untuk evaluasi berikutnya pada `12:16:00`, ia akan mengharapkan data dengan stempel waktu `12:01:00`, `12:06:00`, dan `12:11:00`. Jika ia hanya memiliki data dengan stempel waktu

`12:00:00`, `12:05:00`, dan `12:10:00`, maka semua titik data ini akan diabaikan pada 12:16:00 dan alarm akan beralih status sesuai dengan pengaturan yang Anda tetapkan untuk alarm ketika menangani data yang hilang. Untuk informasi selengkapnya, lihat Melakukan evaluasi alarm.

- Kami menyarankan Anda untuk membuat alarm ini agar melakukan tindakan saat beralih statusnya menjadi `INSUFFICIENT_DATA`, karena beberapa kasus penggunaan fungsi Lambda yang gagal akan mengubah status alarm menjadi `INSUFFICIENT_DATA` terlepas dari pengaturan yang Anda tentukan untuk alarm ketika menangani data yang hilang.
- Jika fungsi Lambda mengembalikan kesalahan atau mengembalikan sebagian data:
  - Jika ada permasalahan izin ketika memanggil fungsi Lambda, alarm akan mulai memiliki transisi data yang hilang sesuai dengan pengaturan yang Anda tentukan untuk alarm ketika menangani data yang hilang saat Anda membuatnya.
  - Jika fungsi Lambda mengembalikan `'StatusCode' = 'PartialData'`, artinya evaluasi alarm mengalami kegagalan, dan alarm tersebut beralih statusnya menjadi `INSUFFICIENT_DATA` setelah tiga upaya, Hal ini memakan waktu sekitar tiga menit.
  - Kesalahan-kesalahan lain yang berasal dari fungsi Lambda akan menyebabkan alarm beralih statusnya menjadi `INSUFFICIENT_DATA`.
- Jika metrik yang diminta oleh fungsi Lambda mengalami beberapa penundaan sehingga titik data terakhir selalu hilang, maka Anda harus menggunakan sebuah solusi untuk menyelesaikannya. Anda dapat membuat sebuah alarm M dari N atau dapat menaikkan periode evaluasi alarm. Untuk informasi selengkapnya tentang alarm M dari N, silakan lihat Melakukan evaluasi alarm.

Buat CloudWatch alarm berdasarkan deteksi anomali

Anda dapat membuat alarm berdasarkan deteksi CloudWatch anomali, yang menganalisis data metrik masa lalu dan membuat model nilai yang diharapkan. Nilai-nilai yang diharapkan tersebut mempertimbangkan pola per jam, harian, dan mingguan dalam metrik.

Anda menetapkan nilai untuk ambang deteksi anomali, dan CloudWatch menggunakan ambang batas ini dengan model untuk menentukan rentang nilai “normal” untuk metrik. Nilai yang lebih tinggi untuk ambang batas akan menghasilkan pita yang lebih tebal dari nilai "normal".

Anda dapat memilih apakah alarm akan dipicu ketika nilai metrik berada di atas nilai pita yang diharapkan, di bawah nilai pita, atau bisa di atas atau di bawah pita.

Anda juga dapat membuat alarm deteksi anomali pada metrik tunggal dan output dari ekspresi matematika metrik. Anda dapat menggunakan ekspresi-ekspresi ini untuk membuat grafik yang menggambarkan pita deteksi anomali.

Di akun yang disiapkan sebagai akun pemantauan untuk pengamatan CloudWatch lintas akun, Anda dapat membuat detektor anomali pada metrik di akun sumber selain metrik di akun pemantauan.

Untuk informasi selengkapnya, lihat Menggunakan CloudWatch deteksi anomali.

> **Note**
>
> Jika Anda sudah menggunakan deteksi anomali untuk tujuan visualisasi pada sebuah metrik yang ada di konsol Metrik dan Anda membuat sebuah alarm deteksi anomali pada metrik yang sama, maka ambang batas yang Anda tetapkan untuk alarm tersebut tidak akan mengubah ambang batas yang sudah Anda tetapkan untuk visualisasi. Untuk informasi selengkapnya, lihat Membuat grafik.

Cara membuat sebuah alarm berdasarkan deteksi anomali

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, pilih Alarm, Semua alarm.
3. Pilih Buat alarm.
4. Pilih Pilih Metrik.
5. Lakukan salah satu hal berikut:
   - Pilih namespace layanan yang berisi metrik Anda, dan kemudian lanjutkan dengan memilih pilihan-pilihan yang tampaknya akan mempersempit opsi Anda. Ketika ada daftar metrik yang ditampilkan, silakan pilih kotak centang yang ada di samping metrik Anda.
   - Pada kotak pencarian, Anda harus memasukkan nama metrik, dimensi, atau ID sumber daya. Kemudian, Anda harus memilih salah satu hasilnya, dan melanjutkan dengan memilih pilihan saat muncul hingga daftar metrik ditampilkan. Centang kotak yang ada di samping metrik Anda.
6. Pilih Metrik grafik.

   a. (Opsional) Untuk Statistik, pilih dropdown, lalu pilih salah satu statistik atau persentil yang telah ditentukan. Anda dapat menggunakan kotak pencarian pada menu geser-turun untuk menentukan sebuah persentil kustom, misalnya **`p95.45`**.

b. (Opsional) Untuk Periode, pilih dropdown, lalu pilih salah satu periode evaluasi yang telah ditentukan.

> **Note**
>
> Saat CloudWatch mengevaluasi alarm Anda, itu mengumpulkan periode menjadi satu titik data. Untuk sebuah alarm deteksi anomali, periode evaluasinya adalah satu menit atau lebih.

7. Pilih Berikutnya.
8. Pada Ketentuan, tentukan hal-hal berikut:

a. Pilih Deteksi anomali.

Jika model untuk metrik dan statistik ini sudah ada, CloudWatch menampilkan pratinjau pita deteksi anomali dalam grafik di bagian atas layar. Setelah membuat alarm, diperlukan waktu hingga 15 menit agar pita deteksi anomali aktual ditampilkan dalam grafik. Sebelum itu, pita yang Anda lihat tersebut adalah perkiraan dari pita deteksi anomali.

> **Tip**
>
> Untuk melihat grafik pada bagian atas layar dalam kerangka waktu yang lebih lama, silakan pilih Sunting pada kanan atas layar.

Jika model untuk metrik dan statistik ini belum ada, CloudWatch buat pita deteksi anomali setelah Anda selesai membuat alarm. Untuk model-model yang baru, diperlukan waktu hingga 3 jam agar pita deteksi anomali aktual ditampilkan pada grafik Anda. Untuk melatih model baru tersebut, diperlukan waktu hingga dua minggu, sehingga pita deteksi anomali akan dapat menunjukkan nilai yang diharapkan dengan lebih akurat.

b. Untuk Jika ***metrik*** adalah, tentukan kapan alarm harus terpicu. Sebagai contoh, ketika metrik lebih besar dari, lebih rendah dari, atau di luar pita (di kedua arah).

c. Untuk Ambang batas deteksi anomali, silakan pilih angka yang akan Anda gunakan untuk ambang batas deteksi anomali. Angka yang lebih tinggi akan menciptakan pita yang lebih tebal dari nilai "normal" sehingga ia akan menjadi lebih toleran terhadap perubahan-perubahan metrik. Angka yang lebih rendah akan menciptakan pita yang lebih tipis yang

akan membuat beralih status menjadi `ALARM` dengan penyimpangan metrik yang lebih kecil. Angka tersebut tidak harus berupa bilangan bulat.

d. Pilih Konfigurasi tambahan. Untuk Titik data alarm, tentukan berapa banyak periode evaluasi (titik data) yang harus ada dalam status `ALARM` untuk memicu alarm. Jika kedua nilai di sini cocok, Anda membuat alarm yang beralih ke status `ALARM` jika terjadi pelanggaran selama sebanyak itu dalam periode berturut-turut.

   Untuk membuat sebuah alarm M dari N, Anda harus menentukan angka untuk nilai pertama dengan nilai yang lebih rendah dari angka untuk nilai kedua. Untuk informasi selengkapnya, lihat Melakukan evaluasi alarm.

e. Untuk Perlakuan data yang hilang, pilih perilaku alarm ketika ada beberapa titik data yang hilang. Untuk informasi selengkapnya, lihat Mengkonfigurasi bagaimana CloudWatch alarm memperlakukan data yang hilang.

f. Jika alarm menggunakan sebuah persentil sebagai statistik yang dipantau, maka kotak Persentil dengan sampel rendah akan ditampilkan. Pilih apakah Anda akan melakukan evaluasi atau mengabaikan kasus yang memiliki tingkat sampel yang kecil. Jika Anda memilih Abaikan (status alarm tidak berubah), maka status alarm saat ini akan tetap dipertahankan ketika ukuran sampel terlalu kecil. Untuk informasi selengkapnya, lihat CloudWatch Alarm berbasis persentil dan sampel data rendah.

9. Pilih Berikutnya.

10. Pada Notifikasi, silakan pilih topik SNS yang akan diberitahukan sebagai notifikasi ketika alarm berada dalam status `ALARM`, status `OK`, atau status `INSUFFICIENT_DATA`.

    Untuk mengirimkan beberapa notifikasi untuk status alarm yang sama atau status alarm yang berbeda, silakan pilih Tambahkan notifikasi.

    Pilih Hapus jika Anda tidak ingin alarm tersebut mengirim notifikasi.

11. Anda dapat mengatur alarm untuk melakukan tindakan EC2 atau menjalankan fungsi Lambda saat perubahan status, atau untuk membuat Systems Manager OpsItem atau insiden saat masuk ke status ALARM. Untuk melakukan hal ini, silakan pilih tombol yang sesuai lalu pilih status alarm dan tindakan yang harus dilakukan.

    Jika Anda memilih sebuah fungsi Lambda sebagai tindakan alarm, maka Anda menentukan nama fungsi atau ARN, dan Anda dapat memilih versi tertentu dari fungsi tersebut secara opsional.

Untuk informasi selengkapnya tentang tindakan Systems Manager, lihat Mengkonfigurasi CloudWatch untuk membuat OpsItems dari alarm dan pembuatan Insiden.

> **Note**
>
> Untuk membuat alarm yang melakukan tindakan Manajer Insiden AWS Systems Manager , Anda harus memiliki izin-izin tertentu. Untuk informasi selengkapnya, lihat contoh kebijakan berbasis identitas untuk Manajer Insiden AWS Systems Manager.

12. Pilih Berikutnya.
13. Pada Nama dan deskripsi, Anda harus memasukkan nama dan deskripsi untuk alarm Anda, dan kemudian pilih Berikutnya. Nama tersebut harus menggunakan karakter UTF-8, dan tidak dapat berisi karakter kontrol ASCII. Deskripsi dapat mencakup pemformatan penurunan harga, yang hanya ditampilkan di tab Detail alarm di CloudWatch konsol. Penurunan harga dapat Anda gunakan untuk menambahkan tautan ke runbook atau sumber daya internal lainnya.

> **Tip**
>
> Nama alarm tersebut harus menggunakan karakter UTF-8, dan tidak dapat berisi karakter kontrol ASCII

14. Pada Pratinjau dan buat, silakan Anda konfirmasi bahwa informasi dan kondisi alarm Anda sudah benar, dan kemudian pilih Buat alarm.

Memodifikasi sebuah model deteksi anomali

Setelah Anda membuat sebuah alarm, Anda dapat menyesuaikan model deteksi anomalinya. Anda dapat mengecualikan periode waktu tertentu sehingga periode tersebut tidak digunakan dalam pembuatan model. Penting bagi Anda untuk mengecualikan peristiwa-peristiwa tidak biasa seperti gangguan sistem, deployment, dan hari libur dari data pelatihan. Anda juga dapat menentukan apakah akan menyesuaikan model untuk perubahan Waktu Musim Panas.

Cara menyesuaikan model deteksi anomali untuk sebuah alarm

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Alarm, Semua alarm.
3. Pilih nama alarm. Jika perlu, Anda bisa menggunakan kotak pencarian untuk menemukan alarm.

4. Pilih Analisis, Dalam metrik.
5. Di kolom Detail, silakan pilih ANOMALY_DETECTION_BAND, Sunting model deteksi anomali.
6. Untuk mengecualikan periode waktu agar tidak digunakan untuk membuat model, Anda bisa pilih ikon kalender berdasarkan Tanggal akhir. Kemudian, pilih atau masukkan hari dan waktu yang akan dikecualikan dari pelatihan, dan pilih Terapkan.
7. Jika metrik tersebut peka terhadap perubahan Waktu Musim Panas, silakan pilih zona waktu yang sesuai pada kotak Zona waktu metrik.
8. Pilih Perbarui.

Menghapus sebuah model deteksi anomali

Menggunakan deteksi anomali untuk sebuah alarm akan menambah biaya. Sebagai praktik terbaik, jika alarm Anda tidak lagi membutuhkan model deteksi anomali, maka Anda harus menghapus alarm tersebut terlebih dahulu dan kemudian menghapus modelnya. Ketika alarm deteksi anomali dievaluasi, setiap detektor anomali yang hilang akan dibuat atas nama Anda. Jika Anda menghapus model tersebut tanpa menghapus alarm, maka alarm akan secara otomatis membuat ulang model yang dihapus tersebut.

Untuk menghapus alarm

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Alarm, Semua Alarm.
3. Pilih nama alarm.
4. Pilih Tindakan, Hapus.
5. Di kotak konfirmasi, pilih Hapus.

Cara menghapus sebuah model deteksi anomali yang digunakan untuk sebuah alarm

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik, dan kemudian pilih Semua metrik.
3. Pilih Jelajah, dan kemudian pilih metrik yang menyertakan model deteksi anomali. Anda dapat mencari metrik Anda melalui kotak pencarian atau memilih metrik Anda dengan memilih melalui opsi-opsinya.
    - (Opsional) Jika Anda menggunakan antarmuka asli, silakan pilih Semua metrik, kemudian pilih metrik yang menyertakan model deteksi anomali tersebut. Anda dapat mencari metrik

Anda melalui kotak pencarian atau memilih metrik Anda dengan memilih melalui opsi-opsinya.

4. Pilih Metrik bergrafik.
5. Pada tab Metrik bergrafik, silakan pilih nama model deteksi anomali yang ingin Anda hapus, dan kemudian pilih Hapus model deteksi anomali.
   - (Opsional) Jika Anda menggunakan antarmuka asli, silakan pilih Sunting model. Anda akan diarahkan ke layar baru. Pada layar baru tersebut, silakan pilih Hapus model, dan kemudian pilih Hapus.

Membuat alarm pada log

Langkah-langkah di bagian berikut menjelaskan cara membuat CloudWatch alarm pada log.

Buat CloudWatch alarm berdasarkan filter metrik grup log

Prosedur yang diuraikan pada bagian ini akan menjelaskan cara membuat alarm berdasarkan filter metrik grup log. Dengan filter metrik, Anda dapat mencari istilah dan pola dalam data log saat data dikirim CloudWatch. Untuk informasi selengkapnya, lihat Membuat metrik dari peristiwa log menggunakan filter di Panduan Pengguna CloudWatch Log Amazon. Sebelum Anda membuat sebuah alarm berdasarkan filter metrik grup log, Anda harus terlebih dahulu menyelesaikan tindakan-tindakan berikut:

- Membuat sebuah grup log. Untuk informasi selengkapnya, lihat Bekerja dengan grup log dan aliran log di Panduan Pengguna CloudWatch Log Amazon.
- Membuat sebuah filter metrik. Untuk informasi selengkapnya, lihat Membuat filter metrik untuk grup log di Panduan Pengguna CloudWatch Log Amazon.

Cara membuat sebuah alarm berdasarkan pada filter metrik grup log

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Dari panel navigasi, silakan pilih Log, kemudian pilih Grup log.
3. Pilih grup log yang menyertakan filter metrik Anda.
4. Pilih Filter metrik.
5. Pada tab filter metrik, silakan pilih kotak untuk filter metrik yang ingin Anda gunakan sebagai dasar alarm Anda.

6. Pilih Buat alarm.
7. (Opsional) Pada Metrik, sunting Nama metrik, Statistik, dan Periode.
8. Pada Ketentuan, tentukan hal-hal berikut:
   a. Untuk Jenis ambang batas, silakan pilih Statis atau Deteksi anomali.
   b. Untuk ***your-metric-name***Kapan pun.. , pilih Lebih Besar, Lebih Hebat/Sama, Lebih Rendah/Sama, atau Lebih Rendah.
   c. Kemudian untuk dari . . ., silakan masukkan angka untuk nilai ambang batas Anda.
9. Pilih Konfigurasi tambahan.
   a. Untuk Titik data untuk alarm, Anda harus menentukan berapa banyak titik data yang memicu alarm Anda sehingga statusnya beralih menjadi `ALARM`. Jika Anda menentukan nilai yang cocok, maka alarm Anda akan beralih statusnya menjadi `ALARM` jika banyak periode berturut-turut yang melanggar. Untuk membuat sebuah alarm M dari N, Anda harus menentukan angka untuk nilai pertama dengan nilai yang lebih rendah dari angka untuk nilai yang kedua. Untuk informasi selengkapnya, lihat Menggunakan CloudWatch alarm Amazon.
   b. Untuk Perlakuan data yang hilang, silakan pilih opsi untuk menentukan cara menangani data yang hilang saat alarm Anda dievaluasi.
10. Pilih Berikutnya.
11. Untuk Notifikasi, Anda harus menentukan topik Amazon SNS yang akan mendapatkan notifikasi saat alarm Anda berada dalam status `ALARM`, `OK`, atau `INSUFFICIENT_DATA`.
    a. (Opsional) Untuk mengirimkan beberapa notifikasi untuk status alarm yang sama atau status alarm yang berbeda, silakan pilih Tambahkan notifikasi.
    b. (Opsional) Jika tidak ingin mengirimkan notifikasi, silakan pilih Hapus.
12. Agar alarm dapat melakukan tindakan penskalaan otomatis, EC2, Lambda atau Systems Manager, pilih tombol yang sesuai dan pilih status alarm serta tindakan yang harus dilakukan. Jika Anda memilih sebuah fungsi Lambda sebagai tindakan alarm, maka Anda menentukan nama fungsi atau ARN, dan Anda dapat memilih versi tertentu dari fungsi tersebut secara opsional.

    Alarm dapat melakukan tindakan Systems Manager hanya ketika masuk ke status ALARM. Untuk informasi selengkapnya tentang tindakan Systems Manager, lihat Mengkonfigurasi CloudWatch untuk membuat OpsItems dari alarm dan pembuatan Insiden.

> **Note**
>
> Untuk membuat alarm yang melakukan tindakan SSM Incident Manager, Anda harus memiliki izin tertentu. Untuk informasi selengkapnya, lihat contoh kebijakan berbasis identitas untuk Manajer Insiden AWS Systems Manager.

13. Pilih Berikutnya.
14. Untuk Nama dan deskripsi, Anda harus memasukkan nama dan deskripsi untuk alarm Anda. Nama tersebut harus menggunakan karakter UTF-8, dan tidak dapat berisi karakter kontrol ASCII. Deskripsi dapat mencakup pemformatan penurunan harga, yang hanya ditampilkan di tab Detail alarm di CloudWatch konsol. Penurunan harga dapat Anda gunakan untuk menambahkan tautan ke runbook atau sumber daya internal lainnya.
15. Untuk Pratinjau dan buat, silakan periksa apakah konfigurasi Anda sudah benar, dan pilih Buat alarm.

Menggabungkan alarm

Dengan CloudWatch, Anda dapat menggabungkan beberapa alarm menjadi satu alarm komposit untuk membuat indikator kesehatan agregat yang diringkas di seluruh aplikasi atau kelompok sumber daya. Alarm gabungan adalah alarm-alarm yang menentukan statusnya dengan memantau status alarm lainnya. Anda harus menentukan aturan-aturan untuk menggabungkan status alarm yang dipantau dengan menggunakan logika Boolean.

Anda dapat menggunakan alarm-alarm gabungan untuk mengurangi kebisingan alarm dengan mengambil tindakan-tindakan hanya pada tingkat agregat. Sebagai contoh, Anda dapat membuat sebuah alarm gabungan untuk mengirim notifikasi ke tim server web Anda jika ada alarm yang terkait dengan server web Anda yang memicu. Ketika salah satu alarm tersebut statusnya beralih menjadi ALARM, alarm gabungan akan beralih sendiri statusnya menjadi ALARM dan mengirimkan notifikasi ke tim Anda. Jika alarm-alarm lain yang terkait dengan server web Anda juga statusnya beralih menjadi ALARM, maka tim Anda tidak akan kelebihan beban dengan notifikasi baru karena alarm gabungan tersebut telah memberi tahu mereka tentang situasi yang ada.

Anda juga dapat menggunakan alarm gabungan untuk membuat kondisi-kondisi peringatan yang kompleks dan akan melakukan tindakan-tindakan hanya ketika banyak kondisi berbeda yang terpenuhi. Sebagai contoh, Anda dapat membuat sebuah alarm gabungan yang menggabungkan

sebuah alarm CPU dan alarm memori, dan hanya akan memberikan notifikasi kepada tim Anda jika CPU dan alarm memori telah dipicu.

Menggunakan alarm gabungan

Saat Anda menggunakan alarm-alarm gabungan, maka Anda memiliki dua pilihan:

- Anda bisa mengonfigurasikan tindakan yang ingin Anda lakukan hanya pada tingkat alarm gabungan, dan membuat alarm terpantau yang mendasari tanpa tindakan
- Anda bisa mengonfigurasikan serangkaian tindakan yang berbeda pada tingkat alarm gabungan. Sebagai contoh, tindakan alarm gabungan dapat melibatkan tim yang berbeda jika terjadi masalah yang meluas.

Alarm-alarm gabungan hanya akan dapat melakukan tindakan-tindakan berikut ini:

- Memberikan notifikasi topik Amazon SNS
- Menginvokasi fungsi Lambda
- Buat OpsItems di Pusat Ops Systems Manager
- Membuat insiden di Systems Manager Incident Manager

> **Note**
>
> Semua alarm yang mendasari pada alarm gabungan Anda harus berada di akun yang sama dan Wilayah yang sama dengan alarm gabungan Anda. Namun, jika Anda mengatur alarm gabungan di akun pemantauan observabilitas CloudWatch lintas akun, alarm yang mendasarinya dapat menonton metrik di akun sumber yang berbeda dan di akun pemantauan itu sendiri. Untuk informasi selengkapnya, lihat CloudWatch observabilitas lintas akun.
>
> Alarm gabungan tunggal dapat memantau 100 alarm yang mendasarinya, dan 150 alarm gabungan dapat memantau satu alarm yang mendasarinya.

Ekspresi aturan

Semua alarm gabungan memuat ekspresi-ekspresi aturan. Ekspresi aturan akan memberitahu alarm gabungan tentang alarm lain mana yang harus dipantau dan menentukan statusnya dari sana. Ekspresi aturan dapat merujuk ke alarm metrik dan alarm gabungan. Saat Anda menjadikan sebuah

alarm sebagai referensi dalam sebuah ekspresi aturan, Anda menentukan fungsi ke alarm yang menentukan dalam status mana dari tiga status berikut alarm itu akan beralih:

- ALARM

  ALARM ("alarm-name atau alarm-ARN") adalah BETUL jika alarm tersebut berada dalam status ALARM.

- OK

  OK ("alarm-name atau alarm-ARN") adalah BETUL jika alarm tersebut berada dalam status OK.

- INSUFFICIENT_DATA

  INSUFFICIENT_DATA ("alarm-name or alarm-ARN") adalah BETUL jika alarm yang disebutkan berada dalam status INSUFFICIENT_DATA.

> **Note**
>
> BETUL selalu mengevaluasi menjadi BETUL, dan SALAH selalu mengevaluasi menjadi SALAH.

Contoh ekspresi

Parameter permintaan `AlarmRule` mendukung penggunaan operator logis `AND`, `OR`, dan `NOT`, sehingga Anda dapat menggabungkan beberapa fungsi menjadi satu ekspresi tunggal. Contoh ekspresi-ekspresi berikut ini menunjukkan bagaimana Anda dapat mengonfigurasi alarm yang mendasari yang ada dalam alarm gabungan Anda:

- `ALARM(CPUUtilizationTooHigh) AND ALARM(DiskReadOpsTooHigh)`

  Ekspresi tersebut menentukan bahwa alarm gabungan beralih statusnya menjadi `ALARM` hanya jika `CPUUtilizationTooHigh` dan `DiskReadOpsTooHigh` berada dalam status `ALARM`.

- `ALARM(CPUUtilizationTooHigh) AND NOT ALARM(DeploymentInProgress)`

  Ekspresi tersebut menentukan bahwa alarm gabungan beralih statusnya menjadi `ALARM` jika `CPUUtilizationTooHigh` statusnya `ALARM` dan `DeploymentInProgress` statusnya bukan `ALARM`. Ini adalah contoh alarm gabungan yang mengurangi kebisingan alarm selama jendela deployment.

- `(ALARM(CPUUtilizationTooHigh) OR ALARM(DiskReadOpsTooHigh)) AND OK(NetworkOutTooHigh)`

  Ekspresi tersebut menentukan bahwa alarm gabungan beralih statusnya menjadi `ALARM` jika `(ALARM(CPUUtilizationTooHigh)` atau `(DiskReadOpsTooHigh)` statusnya `ALARM` dan `(NetworkOutTooHigh)` statusnya `OK`. Ini adalah contoh alarm gabungan yang mengurangi kebisingan alarm dengan tidak mengirimi Anda notifikasi ketika salah satu alarm yang mendasarinya tidak berada dalam status `ALARM` saat terjadi masalah jaringan.

Topik

Membuat sebuah alarm gabungan

Langkah-langkah di bagian ini menjelaskan cara menggunakan CloudWatch konsol untuk membuat alarm komposit. Anda juga dapat menggunakan API atau AWS CLI untuk membuat alarm komposit. Untuk informasi lebih lanjut, lihat PutCompositeAlarmatau put-composite-alarm

Cara membuat sebuah alarm gabungan

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Alarm, kemudian pilih Dalam alarm.
3. Dari daftar alarm, silakan pilih kotak centang yang ada di samping setiap alarm yang sudah ada yang ingin Anda jadikan referensi dalam ekspresi aturan, dan kemudian pilih Buat alarm gabungan.
4. Pada Tentukan kondisi alarm gabungan, Anda perlu menentukan ekspresi aturan untuk alarm gabungan baru Anda.

> **Note**
>
> Secara otomatis, alarm yang Anda pilih dari daftar alarm akan tercantum dalam kotak Ketentuan. Secara default, fungsi `ALARM` telah ditetapkan untuk setiap alarm Anda, dan setiap alarm Anda akan digabungkan oleh operator logis `OR`.

Anda dapat menggunakan langkah-langkah detail berikut untuk mengubah ekspresi aturan Anda:

a. Anda dapat mengubah status yang diperlukan untuk setiap alarm Anda dari status `ALARM` menjadi `OK` atau `INSUFFICENT_DATA`.

b. Anda dapat mengubah operator logis dalam ekspresi aturan Anda dari `OR` ke `AND` atau `NOT`, dan Anda dapat menambahkan tanda kurung untuk mengelompokkan fungsi Anda.

c. Anda dapat menyertakan alarm-alarm lain dalam ekspresi aturan Anda atau menghapus alarm dari ekspresi aturan Anda.

Contoh: Ekspresi aturan dengan kondisi

```
(ALARM("CPUUtilizationTooHigh") OR
ALARM("DiskReadOpsTooHigh")) AND
OK("NetworkOutTooHigh")
```

Dalam contoh ekspresi aturan di mana alarm komposit masuk `ALARM` ketika ALARM (“CPUUtilizationTooHigh" atau ALARM (” DiskReadOpsTooHigh “) masuk `ALARM` bersamaan dengan OK (””NetworkOutTooHigh) masuk`OK`.

5. Setelah selesai, silakan pilih Berikutnya.

6. Pada Konfigurasi tindakan, Anda dapat memilih pilihan-pilihan berikut ini:

Untuk Notifikasi

- Pilih topik SNS yang ada, Buat topik SNS baru, atau Gunakan topik ARN untuk menentukan topik SNS yang akan menerima notifikasi.
- Tambahkan notifikasi, sehingga alarm Anda dapat mengirimkan beberapa notifikasi untuk status alarm yang sama atau status alarm yang berbeda.
- Hapus untuk menghentikan alarm Anda mengirim notifikasi atau melakukan tindakan.

(Opsional) Agar alarm menginvokasi fungsi Lambda saat terjadi perubahan status, pilih Tambahkan tindakan Lambda. Kemudian tentukan nama fungsi atau ARN, dan secara opsional pilih versi fungsi tertentu.

Untuk tindakan Systems Manager

- Tambahkan tindakan Systems Manager, sehingga alarm Anda dapat melakukan tindakan SSM saat alarm tersebut berada dalam status ALARM.

Untuk mempelajari lebih lanjut tentang tindakan Systems Manager, lihat Mengkonfigurasi CloudWatch untuk membuat OpsItems dari alarm di Panduan AWS Systems Manager Pengguna dan Pembuatan insiden di Panduan Pengguna Manajer Insiden. Untuk membuat alarm yang melakukan tindakan SSM Incident Manager, Anda harus memiliki izin yang benar. Untuk informasi selengkapnya, lihat contoh kebijakan berbasis identitas untuk Manajer Insiden AWS Systems Manager di Panduan Pengguna Manajer Insiden.

7. Setelah selesai, silakan pilih Berikutnya.
8. Pada Tambahkan nama dan deskripsi, Anda harus memasukkan nama alarm dan deskripsi opsional untuk alarm gabungan baru Anda. Nama tersebut harus menggunakan karakter UTF-8, dan tidak dapat berisi karakter kontrol ASCII. Deskripsi dapat mencakup pemformatan penurunan harga, yang hanya ditampilkan di tab Detail alarm di CloudWatch konsol. Penurunan harga dapat Anda gunakan untuk menambahkan tautan ke runbook atau sumber daya internal lainnya.
9. Setelah selesai, silakan pilih Berikutnya.
10. Pada Pratinjau dan buat, Anda harus mengonfirmasi informasi Anda, dan kemudian pilih Buat alarm gabungan.

> **Note**
>
> Anda dapat membuat sebuah siklus alarm gabungan, di mana satu alarm gabungan dan alarm gabungan lainnya akan saling bergantung. Jika Anda berada dalam skenario ini, maka alarm gabungan Anda akan berhenti dievaluasi, dan Anda tidak akan dapat menghapus alarm gabungan tersebut karena alarm-alarm itu bergantung satu sama lain. Cara paling mudah untuk memutus siklus dependensi antara alarm gabungan Anda adalah dengan mengubah fungsi `AlarmRule` yang ada di salah satu alarm gabungan Anda menjadi `False`.

Menekan tindakan-tindakan alarm gabungan

Karena alarm gabungan memungkinkan Anda untuk mendapatkan pandangan agregat kondisi Anda di beberapa alarm, ada situasi umum di mana alarm tersebut diharapkan dapat dipicu. Sebagai

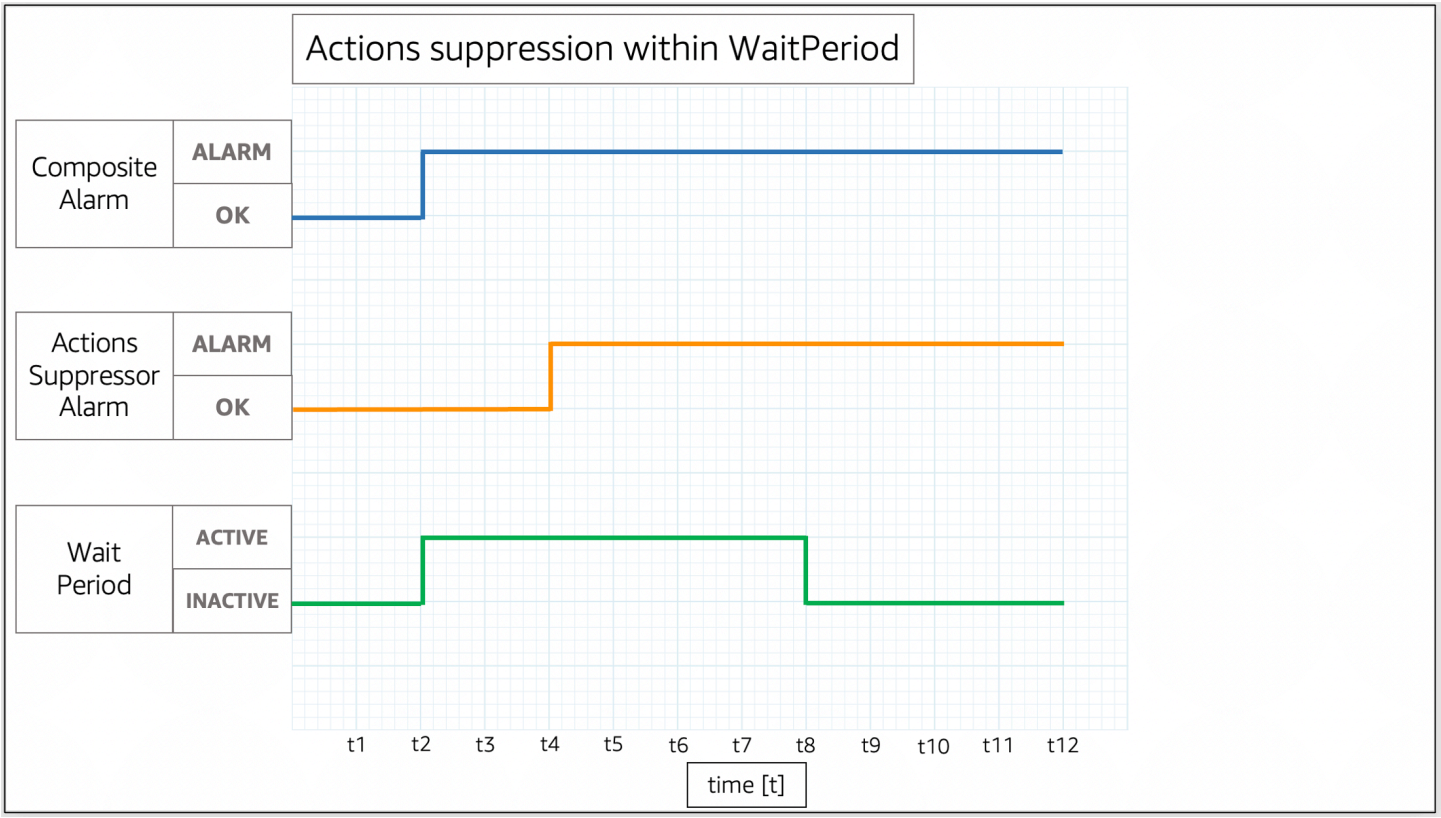
contoh, selama jendela pemeliharaan aplikasi Anda atau ketika Anda menyelidiki sebuah insiden yang sedang berlangsung. Dalam situasi seperti itu, Anda mungkin ingin menekan tindakan-tindakan dari alarm gabungan Anda, untuk mencegah notifikasi yang tidak diinginkan atau pembuatan tiket insiden baru

Dengan melakukan penekanan tindakan alarm gabungan, Anda harus mendefinisikan alarm sebagai alarm penekan. Alarm penekan akan mencegah alarm gabungan agar tidak mengambil tindakan. Sebagai contoh, Anda dapat menentukan sebuah alarm penekan yang mewakili status sumber daya pendukung. Jika sumber daya pendukung tersebut mati, maka alarm penekan akan mencegah alarm gabungan mengirim notifikasi. Penekanan tindakan alarm gabungan ini akan membantu Anda mengurangi kebisingan alarm, sehingga Anda akan menghabiskan lebih sedikit waktu untuk mengelola alarm dan lebih banyak waktu untuk fokus pada operasi Anda.

Anda menentukan alarm penekan saat Anda mengonfigurasi alarm gabungan. Alarm apa pun dapat berfungsi sebagai alarm penekan. Ketika alarm penekan berubah statusnya dari `OK` menjadi `ALARM`, alarm gabungan akan berhenti mengambil tindakan. Ketika sebuah alarm penekan berubah statusnya dari `ALARM` menjadi `OK`, alarm gabungan akan kembali melanjutkan mengambil tindakan.

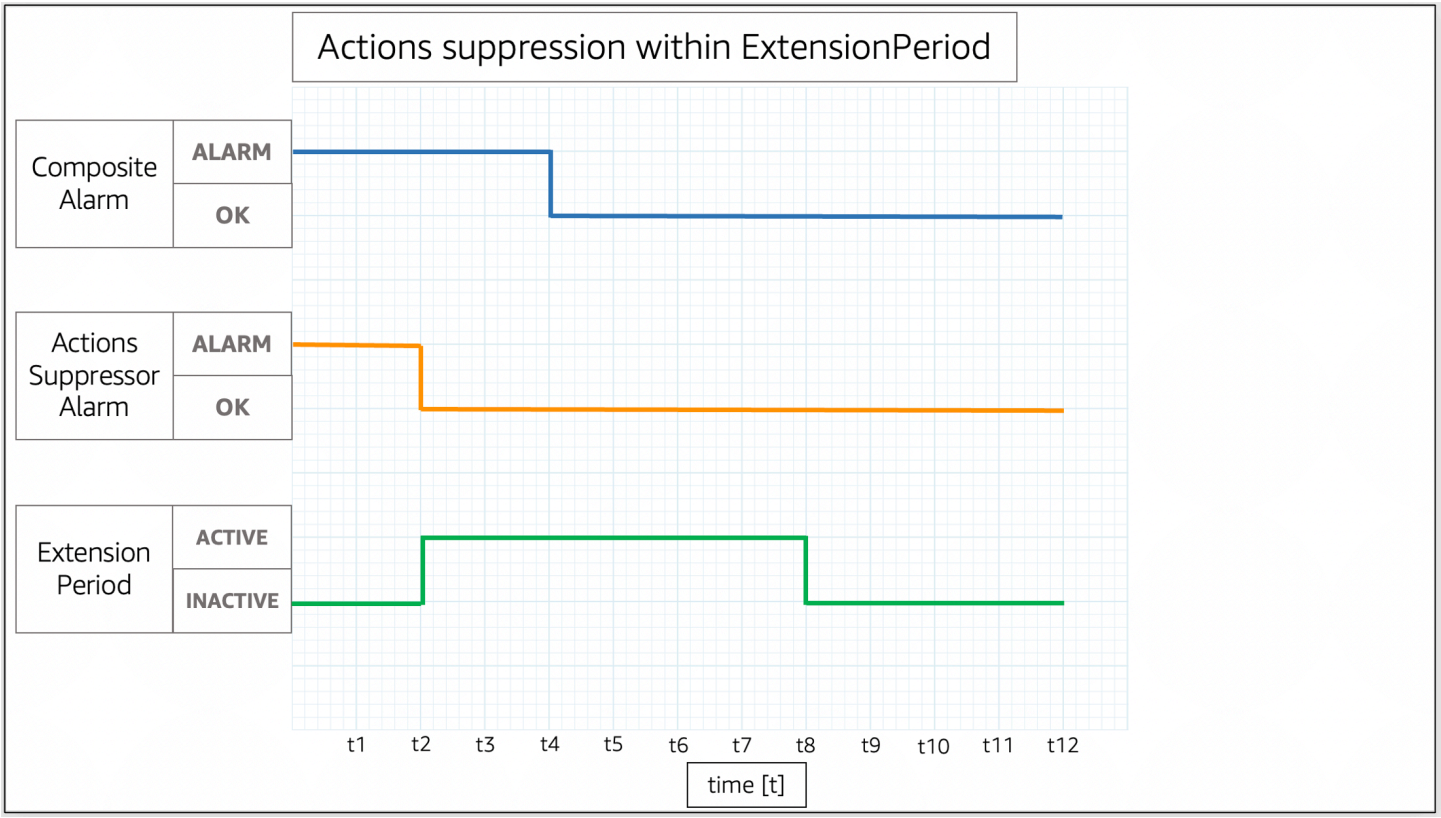
`WaitPeriod` dan **`ExtensionPeriod`**

Saat Anda menentukan sebuah alarm penekan, maka Anda harus mengatur parameter `WaitPeriod` dan `ExtensionPeriod`. Parameter-parameter ini akan mencegah alarm gabungan mengambil tindakan secara tidak terduga saat alarm penekan statusnya berubah. Gunakan `WaitPeriod` untuk mengkompensasi penundaan yang dapat terjadi ketika sebuah alarm penekan berubah statusnya dari `OK` menjadi `ALARM`. Sebagai contoh, jika alarm penekan berubah statusnya dari `OK` menjadi `ALARM` dalam 60 detik, atur `WaitPeriod` ke 60 detik.



Pada gambar tersebut, status alarm gabungan berubah dari OK menjadi ALARM di t2. Sebuah `WaitPeriod` dimulai pada t2 dan berakhir pada t8. Hal ini akan memberi waktu bagi alarm penekan untuk mengubah statusnya dari OK menjadi ALARM di t4 sebelum menekan tindakan alarm gabungan saat `WaitPeriod` kedaluwarsa pada t8.

Gunakan `ExtensionPeriod` untuk memberikan kompensasi atas penundaan apa pun yang dapat terjadi ketika alarm gabungan berubah statusnya menjadi OK mengikuti alarm penekan yang berubah menjadi OK. Sebagai contoh, jika alarm gabungan berubah statusnya menjadi OK dalam 60 detik setelah alarm penekan berubah statusnya menjadi OK, atur `ExtensionPeriod` ke 60 detik.



Pada gambar tersebut, alarm penekan berubah statusnya dari `ALARM` menjadi `OK` di t2. Sebuah `ExtensionPeriod` dimulai pada t2 dan berakhir pada t8. Hal ini akan memberi waktu bagi alarm gabungan untuk berubah statusnya dari `ALARM` menjadi `OK` sebelum `ExtensionPeriod` kedaluwarsa pada t8.

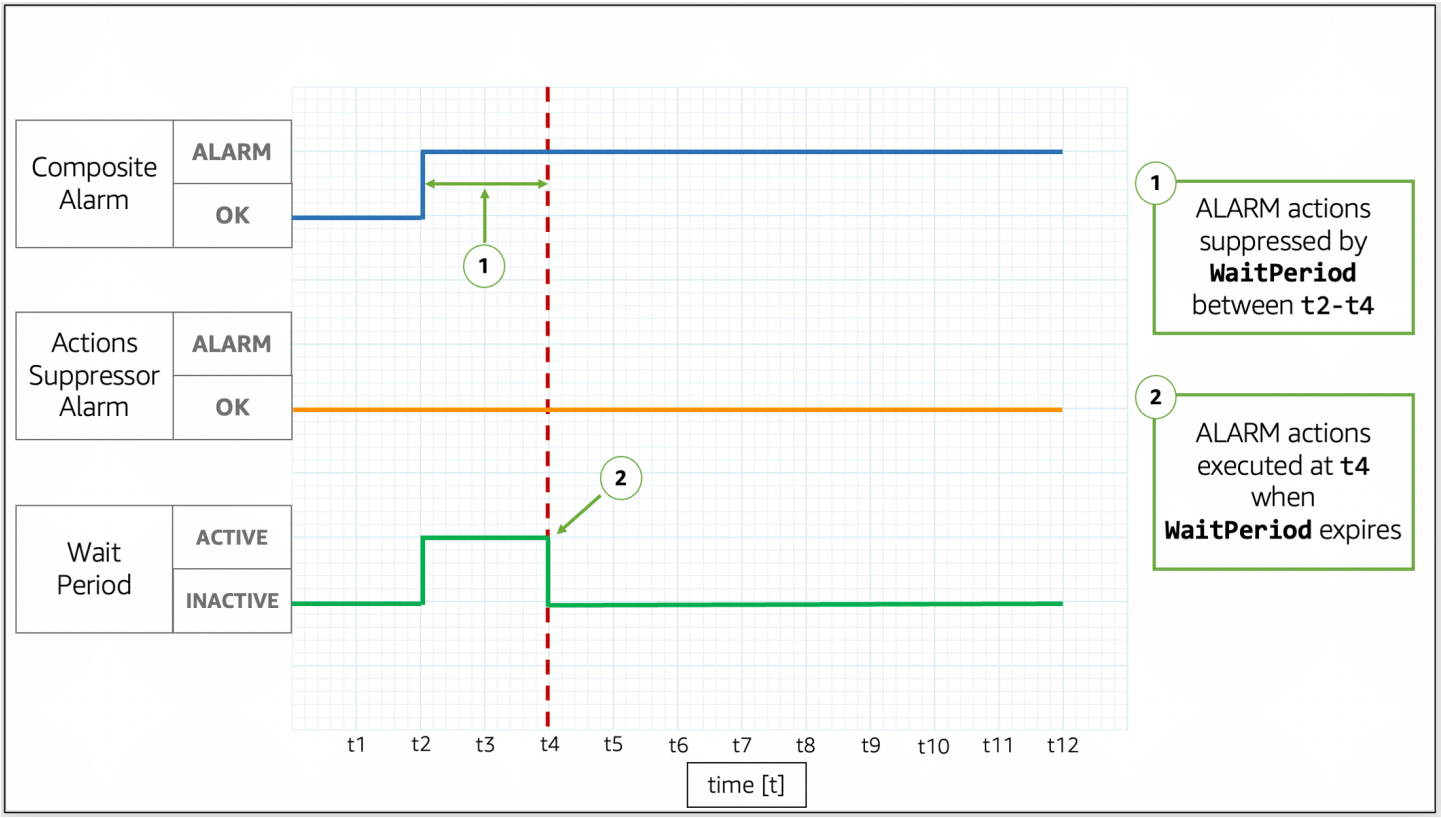
Alarm gabungan tidak akan mengambil tindakan saat `WaitPeriod` dan `ExtensionPeriod` menjadi aktif. Alarm gabungan akan mengambil tindakan yang didasarkan pada statusnya saat ini ketika `ExtensionPeriod` dan `WaitPeriod` menjadi tidak aktif. Kami menyarankan Anda menetapkan nilai untuk setiap parameter menjadi 60 detik, karena CloudWatch mengevaluasi alarm metrik setiap menit. Anda dapat menyetel parameter-parameter tersebut dengan bilangan bulat apa pun dalam hitungan detik.

Contoh berikut menjelaskan secara lebih rinci bagaimana `WaitPeriod` dan `ExtensionPeriod` mencegah alarm gabungan mengambil tindakan secara tidak terduga.

> **Note**
>
> Dalam contoh berikut ini, `WaitPeriod` dikonfigurasi sebagai 2 unit waktu, dan `ExtensionPeriod` dikonfigurasi sebagai 3 unit waktu.

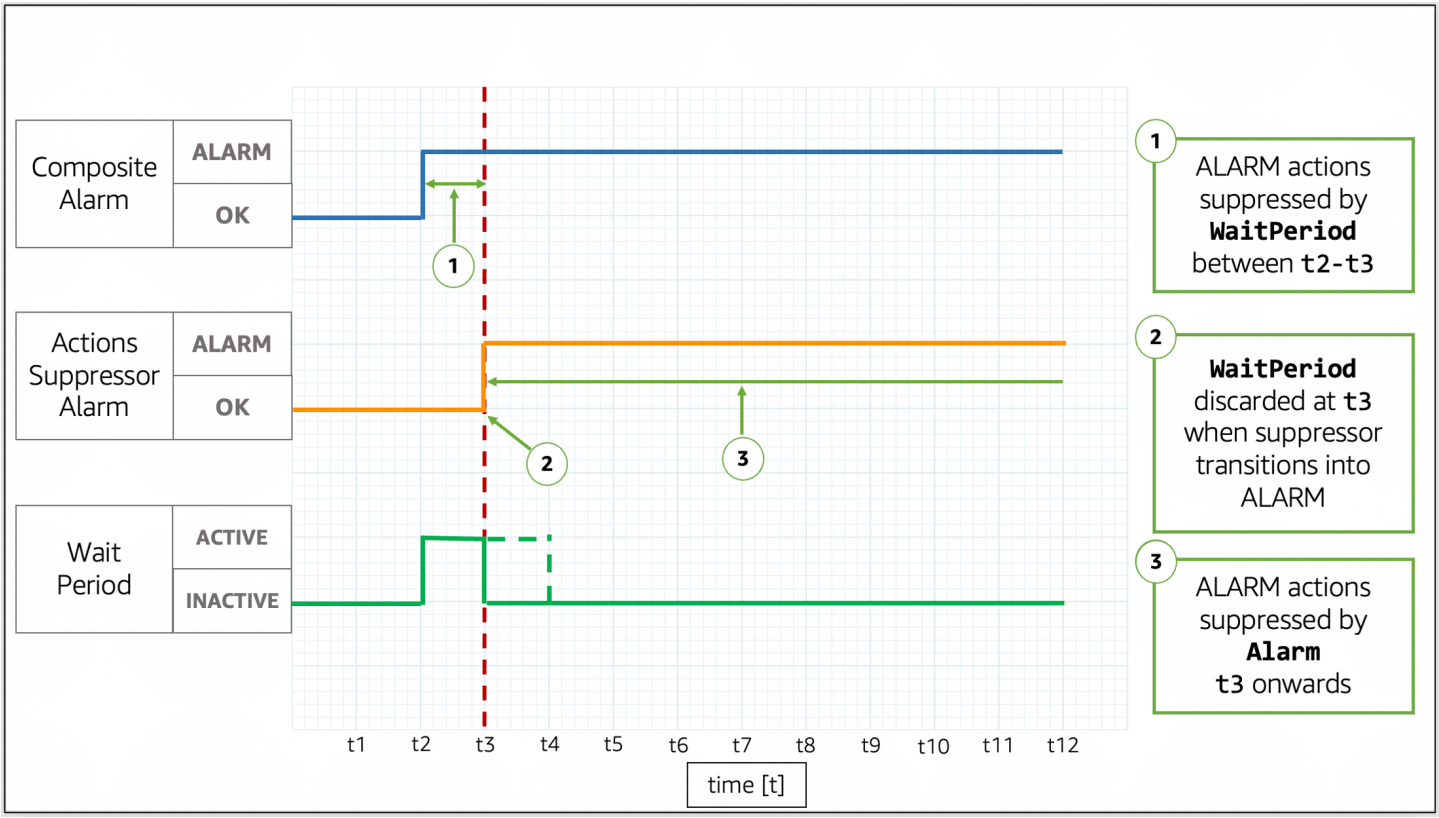
Contoh-contoh

Contoh 1: Tindakan tidak ditekan setelah **`WaitPeriod`**



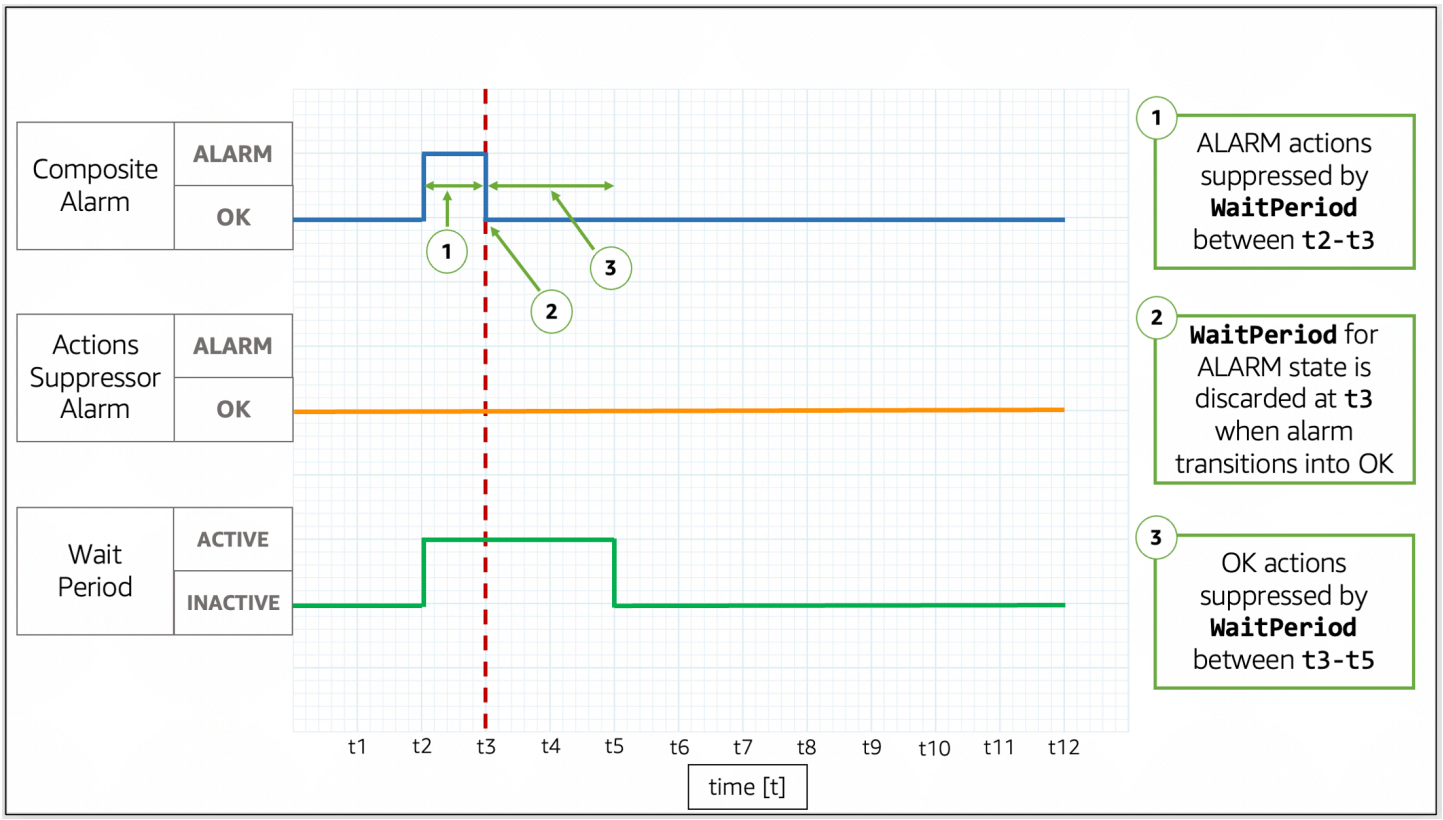
Pada gambar tersebut, alarm gabungan berubah statusnya dari `OK` menjadi `ALARM` di t2. Sebuah `WaitPeriod` dimulai pada t2 dan berakhir pada t4, sehingga ia dapat mencegah alarm gabungan agar tidak mengambil tindakan. Setelah `WaitPeriod` kedaluwarsa pada t4, alarm gabungan akan mengambil tindakan karena alarm penekan masih berada dalam status `OK`.

Contoh 2: Tindakan ditekan oleh alarm sebelum **`WaitPeriod`** mengalami kedaluwarsa



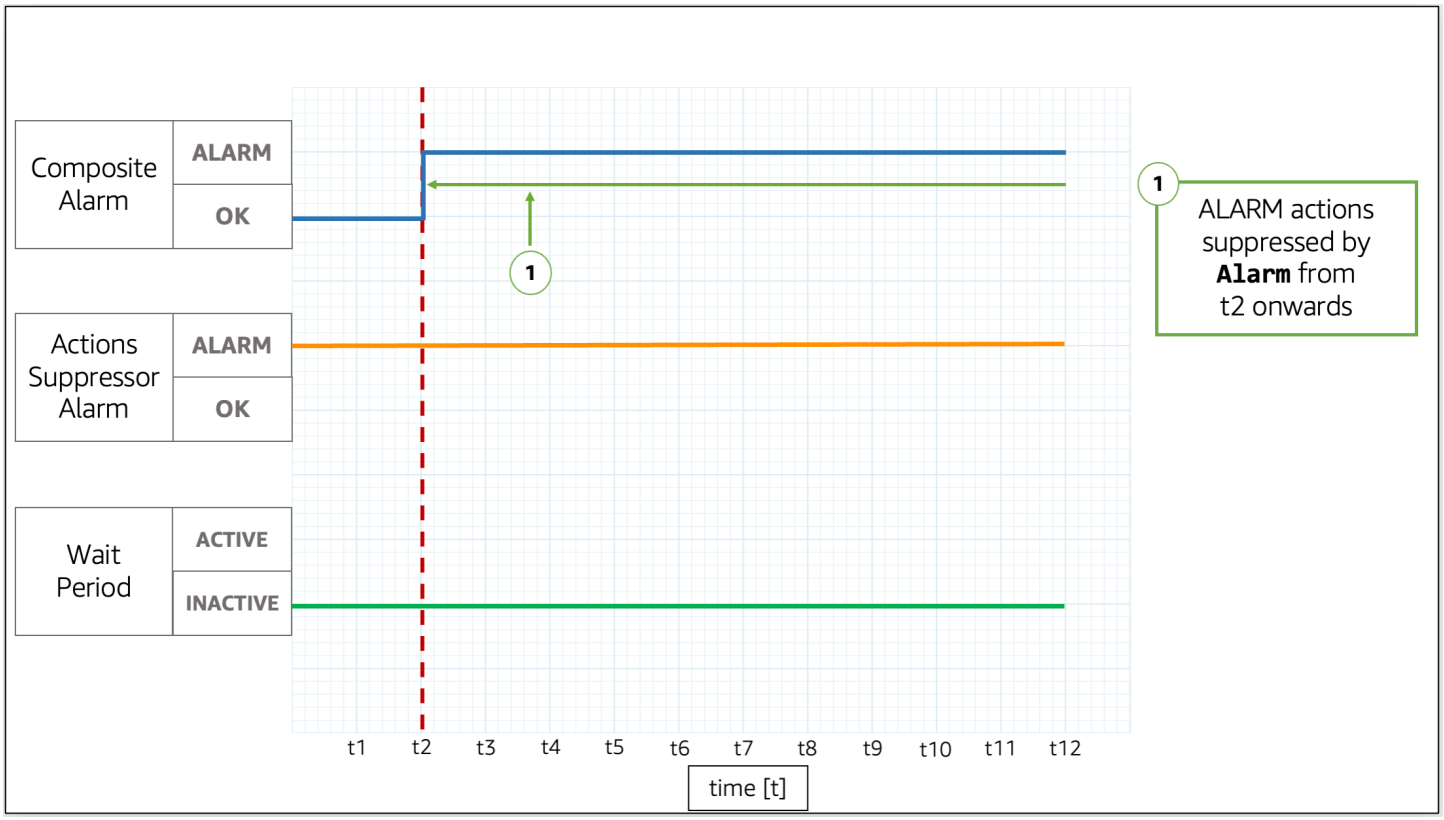
Pada gambar tersebut, alarm gabungan berubah statusnya dari OK menjadi ALARM di t2. Sebuah `WaitPeriod` dimulai pada t2 dan berakhir pada t4. Hal ini akan memberi waktu bagi alarm penekan untuk mengubah statusnya dari OK menjadi ALARM di t3. Karena alarm penekan berubah statusnya dari OK menjadi ALARM di t3, `WaitPeriod` yang dimulai pada t2 kemudian akan dibuang, dan alarm penekan sekarang menghentikan alarm gabungan agar tidak lagi melakukan tindakan.

Contoh 3: Perubahan status saat tindakan ditekan oleh **`WaitPeriod`**



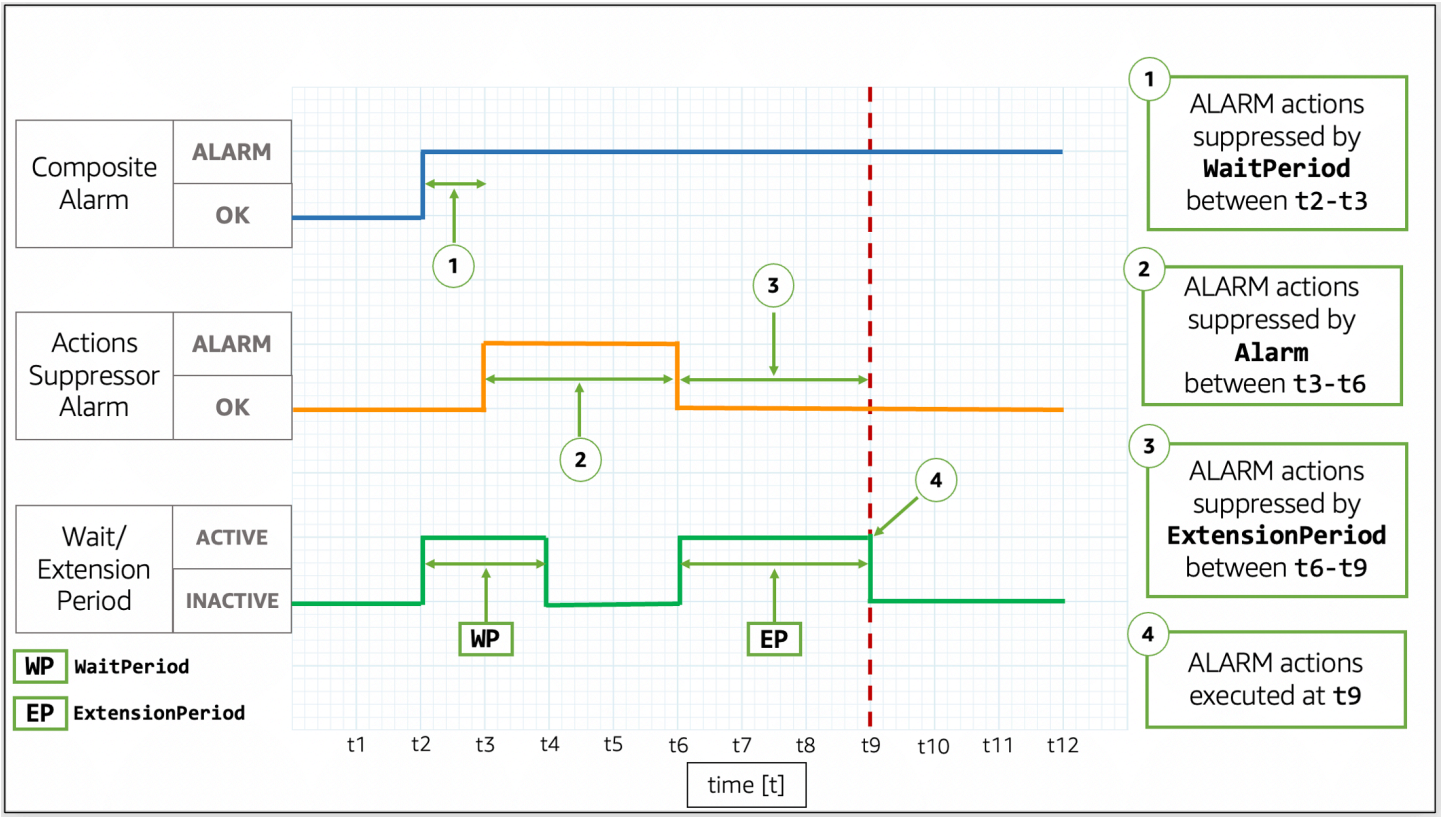
Pada gambar tersebut, alarm gabungan berubah statusnya dari OK menjadi ALARM di t2. Sebuah `WaitPeriod` dimulai pada t2 dan berakhir pada t4. Hal ini akan memberi waktu bagi alarm penekan untuk mengubah statusnya. Alarm gabungan akan mengubah kembali statusnya menjadi OK pada t3, sehingga `WaitPeriod` yang dimulai pada t2 akan dibuang. Sebuah `WaitPeriod` baru dimulai pada t3 dan berakhir pada t5. Setelah `WaitPeriod` baru kedaluwarsa pada t5, alarm gabungan akan mengambil tindakan.

Contoh 4: Perubahan status saat tindakan ditekan oleh alarm



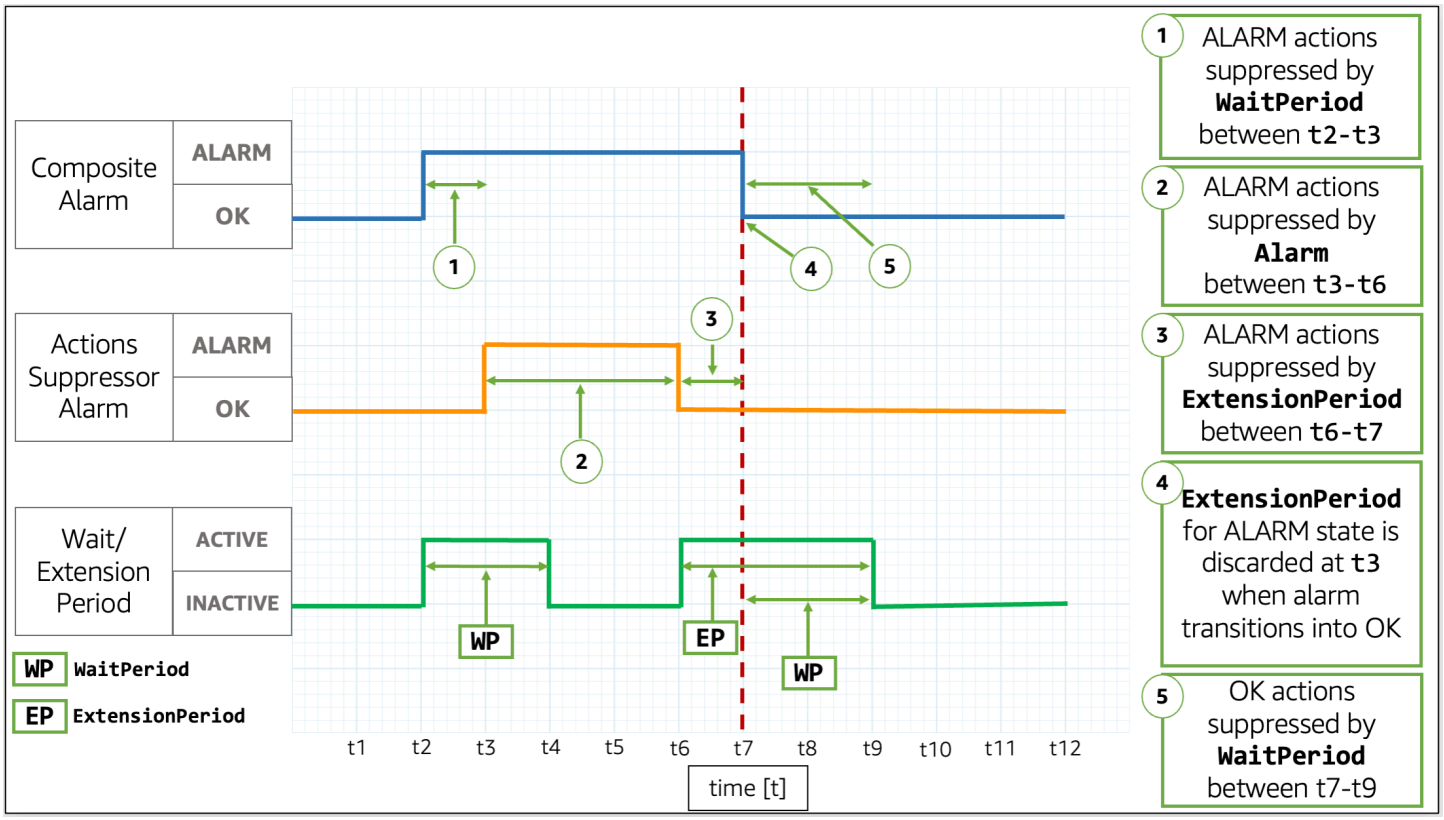
Pada gambar tersebut, alarm gabungan berubah statusnya dari `OK` menjadi `ALARM` di t2. Alarm penekan sudah berada dalam status `ALARM`. Alarm penekan tersebut akan menghentikan alarm gabungan sehingga ia tidak akan lagi mengambil tindakan.

Contoh 5: Tindakan tidak ditekan setelah **`ExtensionPeriod`**



Pada gambar tersebut, alarm gabungan berubah statusnya dari OK menjadi ALARM di t2. Sebuah `WaitPeriod` dimulai pada t2 dan berakhir pada t4. Hal ini akan memberi waktu bagi alarm penekan untuk mengubah statusnya dari OK menjadi ALARM di t3 sebelum menekan tindakan alarm gabungan hingga t6. Karena alarm penekan berubah statusnya dari OK menjadi ALARM di t3, `WaitPeriod` yang dimulai pada t2 kemudian dibuang. Pada t6, alarm penekan mengubah statusnya menjadi OK. Sebuah `ExtensionPeriod` dimulai pada t6 dan berakhir pada t9. Setelah `ExtensionPeriod` kedaluwarsa, alarm gabungan kemudian akan mengambil tindakan.

Contoh 6: Perubahan status saat tindakan ditekan oleh **`ExtensionPeriod`**



Pada gambar tersebut, alarm gabungan berubah statusnya dari OK menjadi ALARM di t2. Sebuah `WaitPeriod` dimulai pada t2 dan berakhir pada t4. Hal ini akan memberi waktu bagi alarm penekan untuk mengubah statusnya dari OK menjadi ALARM di t3 sebelum menekan tindakan alarm gabungan hingga t6. Karena alarm penekan berubah statusnya dari OK menjadi ALARM di t3, `WaitPeriod` yang dimulai pada t2 kemudian dibuang. Pada t6, alarm penekan mengubah statusnya kembali menjadi OK. Sebuah `ExtensionPeriod` dimulai pada t6 dan berakhir pada t9. Ketika alarm gabungan mengubah kembali statusnya menjadi OK di t7, alarm akan `ExtensionPeriod` kemudian dibuang, dan alarm `WaitPeriod` baru dimulai pada t7 dan berakhir pada t9.

> **Tip**
>
> Jika Anda mengganti alarm penekan tindakan tersebut, maka `WaitPeriod` atau `ExtensionPeriod` yang aktif dibuang.

Melakukan tindakan pada perubahan alarm

CloudWatch dapat memberi tahu pengguna tentang dua jenis perubahan alarm: ketika alarm berubah status, dan ketika konfigurasi alarm diperbarui.

Saat alarm melakukan evaluasi, alarm tersebut mungkin mengubah stausnya dari satu status ke status lainnya, seperti ALARM, OK atau INSUFFICIENT_DATA. Perubahan status alarm ini dapat menandakan adanya kemungkinan insiden, kembali ke normal, atau metrik yang tidak tersedia. Dalam kasus seperti itu, Anda mungkin ingin melibatkan atau memberikan notifikasi kepada para pengguna dengan menggunakan salah satu opsi berikut:

- Anda dapat mengonfigurasi alarm untuk mengirim sebuah notifikasi ke topik SNS sebagai bagian dari tindakan alarm. Topik SNS kemudian dapat dikonfigurasi untuk pesan application-to-application (A2A) serta pemberitahuan application-to-person (A2P), termasuk saluran seperti pemberitahuan email dan SMS. Semua tujuan yang Anda tentukan untuk topik SNS Anda akan menerima notifikasi alarm. Untuk informasi selengkapnya, silakan lihat destinasi peristiwa Amazon SNS.
- Anda dapat mengonfigurasi pemberitahuan untuk peristiwa perubahan status alarm. AWS Pemberitahuan Pengguna menawarkan cara asli untuk mengonfigurasi pemberitahuan tersebut dan merupakan pendekatan yang disarankan.

Selain itu, CloudWatch mengirimkan peristiwa ke Amazon EventBridge setiap kali alarm berubah status, dan saat alarm dibuat, dihapus, atau diperbarui. Anda dapat menulis EventBridge aturan untuk mengambil tindakan atau diberi tahu saat EventBridge menerima acara ini.

Topik

Memberikan notifikasi kepada pengguna tentang perubahan alarm

Bagian ini menjelaskan bagaimana Anda dapat menggunakan Pemberitahuan AWS Pengguna atau Layanan Pemberitahuan Sederhana Amazon agar pengguna diberi tahu tentang perubahan alarm.

Menyiapkan Pemberitahuan AWS Pengguna

Anda dapat menggunakan Pemberitahuan AWS Pengguna untuk mengatur saluran pengiriman agar mendapat pemberitahuan tentang perubahan status CloudWatch alarm dan peristiwa perubahan konfigurasi. Anda akan menerima sebuah notifikasi saat ada sebuah peristiwa yang cocok dengan sebuah aturan yang Anda tentukan. Anda dapat menerima notifikasi untuk peristiwa-peristiwa melalui beberapa saluran, termasuk email, notifikasi obrolan AWS Chatbot, atau notifikasi push AWS Console Mobile Application. Anda juga dapat melihat notifikasi di Pusat Notifikasi Konsol. Notifikasi

Pengguna mendukung agregasi, hal ini akan dapat mengurangi jumlah notifikasi yang Anda terima selama peristiwa-peristiwa tertentu.

Konfigurasi notifikasi yang Anda buat dengan Pemberitahuan AWS Pengguna tidak dihitung terhadap batas jumlah tindakan yang dapat Anda konfigurasikan per status alarm target. Karena Notificatinos AWS Pengguna cocok dengan peristiwa yang dipancarkan ke EventBridge Amazon, ia mengirimkan pemberitahuan untuk semua alarm di akun Anda dan Wilayah yang dipilih, kecuali jika Anda menentukan filter lanjutan untuk mengizinkan atau menolak daftar alarm atau pola tertentu.

Contoh filter lanjutan berikut cocok dengan perubahan status alarm dari OK menjadi ALARM pada alarm bernama `ServerCpuTooHigh`.

```
{
"detail": {
    "alarmName": ["ServerCpuTooHigh"],
    "previousState": { "value": ["OK"] },
    "state": { "value": ["ALARM"] }
  }
}
```

Anda dapat menggunakan salah satu properti yang diterbitkan oleh alarm dalam EventBridge acara untuk membuat filter. Untuk informasi selengkapnya, lihat Peristiwa alarm dan EventBridge.

Menyiapkan notifikasi Amazon SNS

Anda dapat menggunakan Amazon Simple Notification Service untuk mengirim pesan application-to-application (A2A) dan pesan application-to-person (A2P), termasuk pesan teks seluler (SMS) dan pesan email. Untuk informasi selengkapnya, silakan lihat destinasi peristiwa Amazon SNS.

Untuk setiap status yang dapat diambil alarm, Anda dapat mengonfigurasi alarm tersebut untuk mengirim pesan ke sebuah topik SNS. Setiap topik Amazon SNS yang Anda konfigurasi untuk sebuah status pada alarm tertentu akan terhitung dalam batasan jumlah tindakan yang dapat Anda konfigurasi untuk alarm dan status tersebut. Anda dapat mengirim pesan ke topik Amazon SNS yang sama dari alarm apa pun yang ada di akun Anda, dan menggunakan topik Amazon SNS yang sama untuk konsumen aplikasi (A2A) dan perorangan (A2P). Karena konfigurasi ini diatur pada tingkat alarm, hanya alarm yang telah Anda konfigurasikan saja yang akan mengirim pesan ke topik Amazon SNS yang Anda pilih.

Pertama, Anda harus membuat sebuah topik, lalu berlangganan padanya. Anda dapat secara opsional menerbitkan sebuah pesan uji ke topik tersebut. Sebagai contoh, silakan lihat Menyiapkan

topik Amazon SNS menggunakan AWS Management Console. Atau untuk informasi selengkapnya, silakan lihat Memulai Amazon SNS.

Atau, jika Anda berencana untuk menggunakan AWS Management Console untuk membuat CloudWatch alarm Anda, Anda dapat melewati prosedur ini karena Anda dapat membuat topik saat Anda membuat alarm.

Saat Anda membuat CloudWatch alarm, Anda dapat menambahkan tindakan untuk status target apa pun yang dimasukkan alarm. Tambahkan sebuah notifikasi Amazon SNS untuk status yang notifikasinya ingin Anda dapatkan, dan kemudian pilih topik Amazon SNS yang Anda buat di langkah sebelumnya untuk mengirim notifikasi melalui email saat alarm berada dalam status yang dipilih.

> **Note**
>
> Saat Anda membuat topik Amazon SNS, Anda memilih untuk menjadikannya topik standar atau topik FIFO. CloudWatch menjamin publikasi semua pemberitahuan alarm untuk kedua jenis topik. Namun, bahkan jika Anda menggunakan topik FIFO, dalam kasus yang jarang terjadi CloudWatch mengirimkan pemberitahuan ke topik yang rusak. Jika Anda menggunakan sebuah topik FIFO, maka alarm menyetel ID grup pesan dari notifikasi alarm menjadi hash ARN alarm tersebut.

Mencegah masalah wakil yang membingungkan

Untuk mencegah masalah keamanan deputi lintas layanan yang membingungkan, kami sarankan Anda menggunakan kunci kondisi `aws:SourceArn` dan `aws:SourceAccount` global dalam kebijakan sumber daya Amazon SNS yang memberikan izin CloudWatch untuk mengakses sumber daya Amazon SNS Anda.

Contoh kebijakan sumber daya berikut menggunakan kunci `aws:SourceArn` kondisi untuk mempersempit `SNS:Publish` izin yang akan digunakan hanya oleh CloudWatch alarm di akun yang ditentukan.

```
{
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "cloudwatch.amazonaws.com"
        },
```

```
        "Action": "SNS:Publish",
        "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
        "Condition": {
            "ArnLike": {
                "aws:SourceArn": "arn:aws:cloudwatch:us-east-2:111122223333:alarm:*"
            },
            "StringEquals": {
                "aws:SourceAccount": "111122223333"
            }
        }
    }]
}
```

Jika sebuah ARN alarm menyertakan karakter non-ASCII, maka Anda hanya bisa menggunakan kunci kondisi global `aws:SourceAccount` untuk membatasi izin.

Menyiapkan topik Amazon SNS menggunakan AWS Management Console

Pertama, Anda harus membuat sebuah topik, lalu berlangganan padanya. Anda dapat secara opsional menerbitkan sebuah pesan uji ke topik tersebut.

Cara membuat sebuah topik SNS

1. Buka konsol Amazon SNS di https://console.aws.amazon.com/sns/v3/home.
2. Pada dasbor Amazon SNS, pada Tindakan umum, silakan pilih Buat Topik.
3. Pada kotak dialog Buat topik baru, untuk Nama topik, masukkan nama untuk topik (misalnya, **`my-topic`**).
4. Pilih Buat topik.
5. Salin ARN Topik untuk tugas berikutnya (misalnya, arn:aws:sns:us-east-1:111122223333:my-topic).

Cara berlangganan ke sebuah topik SNS

1. Buka konsol Amazon SNS di https://console.aws.amazon.com/sns/v3/home.
2. Pada panel navigasi, silakan pilih Berlangganan, Buat berlangganan.
3. Pada kotak dialog Buat berlangganan, untuk ARN Topik, Anda harus menempelkan ARN topik yang sudah dibuat pada tugas sebelumnya.
4. Untuk Protokol, pilih Email.

5. Untuk Titik Akhir, Anda harus memasukkan alamat email yang dapat Anda gunakan untuk menerima notifikasi, dan kemudian pilih Buat berlangganan.
6. Dari aplikasi email Anda, buka pesan dari AWS Pemberitahuan dan konfirmasikan langganan Anda.

   Browser web Anda menampilkan respons konfirmasi dari Amazon SNS.

Cara menerbitkan sebuah pesan uji ke sebuah topik SNS

1. Buka konsol Amazon SNS di https://console.aws.amazon.com/sns/v3/home.
2. Pada panel navigasi, silakan pilih Pengguna.
3. Pada halaman Topik, silakan pilih topik dan pilih Terbitkan ke topik.
4. Pada halaman Terbitkan pesan, untuk Subjek, masukkan baris subjek untuk pesan Anda, dan untuk Pesan, masukkan pesan singkat.
5. Pilih Terbitkan Pesan.
6. Periksa email Anda untuk mengonfirmasi bahwa Anda telah menerima pesan tersebut.

Menyiapkan topik SNS menggunakan AWS CLI

Pertama, Anda perlu membuat topik SNS, dan kemudian Anda harus menerbitkan sebuah pesan secara langsung ke topik tersebut untuk menguji apakah Anda telah mengonfigurasinya dengan benar atau tidak.

Cara menyiapkan sebuah topik SNS

1. Anda harus membuat topik dengan menggunakan perintah create-topic sebagai berikut.

   ```
   aws sns create-topic --name my-topic
   ```

   Amazon SNS kemudian akan merespon dengan sebuah ARN topik dengan format berikut:

   ```
   {
       "TopicArn": "arn:aws:sns:us-east-1:111122223333:my-topic"
   }
   ```

2. Buat alamat email Anda berlangganan ke topik dengan menggunakan perintah subscribe. Jika permintaan berlangganan berhasil, maka Anda akan menerima sebuah pesan email konfirmasi.

```
aws sns subscribe --topic-arn arn:aws:sns:us-east-1:111122223333:my-topic --
protocol email --notification-endpoint my-email-address
```

Amazon SNS mengembalikan hal berikut:

```
{
    "SubscriptionArn": "pending confirmation"
}
```

3. Dari aplikasi email Anda, buka pesan dari AWS Pemberitahuan dan konfirmasikan langganan Anda.

   Browser web Anda akan menampilkan respons konfirmasi dari Amazon Simple Notification Service.

4. Periksa langganan menggunakan list-subscriptions-by-topicperintah.

```
aws sns list-subscriptions-by-topic --topic-arn arn:aws:sns:us-
east-1:111122223333:my-topic
```

Amazon SNS akan merespons dengan hal-hal berikut:

```
{
  "Subscriptions": [
    {
        "Owner": "111122223333",
        "Endpoint": "me@mycompany.com",
        "Protocol": "email",
        "TopicArn": "arn:aws:sns:us-east-1:111122223333:my-topic",
        "SubscriptionArn": "arn:aws:sns:us-east-1:111122223333:my-topic:64886986-
bf10-48fb-a2f1-dab033aa67a3"
    }
  ]
}
```

5. (Opsional) Terbitkan sebuah pesan uji ke topik tersebut dengan menggunakan perintah publish.

```
aws sns publish --message "Verification" --topic arn:aws:sns:us-
east-1:111122223333:my-topic
```

Amazon SNS akan memberikan respons seperti berikut:

```
{
    "MessageId": "42f189a0-3094-5cf6-8fd7-c2dde61a4d7d"
}
```

6. Periksa email Anda untuk mengonfirmasi bahwa Anda telah menerima pesan tersebut.

Peristiwa alarm dan EventBridge

CloudWatch mengirimkan peristiwa-peristiwa ke Amazon EventBridge kapan saja sebuah alarm CloudWatch dibuat, diperbarui, dihapus, atau mengubah status alarm. Anda dapat menggunakan EventBridge dan peristiwa-peristiwa ini untuk menulis aturan-aturan yang mengambil tindakan, seperti yang memberikan notifikasi untuk Anda, ketika ada status alarm yang berubah. Untuk informasi selengkapnya, silakan lihat Apa yang dimaksud dengan Amazon EventBridge?

CloudWatch menjamin pengiriman peristiwa-peristiwa perubahan status alarm ke EventBridge.

Contoh peristiwa-peristiwa dari CloudWatch

Bagian ini mencakup contoh peristiwa-peristiwa dari CloudWatch.

Perubahan status untuk alarm metrik tunggal

```
{
    "version": "0",
    "id": "c4c1c1c9-6542-e61b-6ef0-8c4d36933a92",
    "detail-type": "CloudWatch Alarm State Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2019-10-02T17:04:40Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServerCpuTooHigh"
    ],
    "detail": {
        "alarmName": "ServerCpuTooHigh",
        "configuration": {
            "description": "Goes into alarm when server CPU utilization is too high!",
            "metrics": [
```

```
                {
                    "id": "30b6c6b2-a864-43a2-4877-c09a1afc3b87",
                    "metricStat": {
                        "metric": {
                            "dimensions": {
                                "InstanceId": "i-12345678901234567"
                            },
                            "name": "CPUUtilization",
                            "namespace": "AWS/EC2"
                        },
                        "period": 300,
                        "stat": "Average"
                    },
                    "returnData": true
                }
            ]
        },
        "previousState": {
            "reason": "Threshold Crossed: 1 out of the last 1 datapoints
 [0.0666851903306472 (01/10/19 13:46:00)] was not greater than the threshold (50.0)
 (minimum 1 datapoint for ALARM -> OK transition).",
            "reasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2019-10-01T13:56:40.985+0000\",\"startDate\":\"2019-10-01T13:46:00.000+0000\",
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":[0.0666851903306472],
\"threshold\":50.0}",
            "timestamp": "2019-10-01T13:56:40.987+0000",
            "value": "OK"
        },
        "state": {
            "reason": "Threshold Crossed: 1 out of the last 1 datapoints
 [99.50160229693434 (02/10/19 16:59:00)] was greater than the threshold (50.0) (minimum
 1 datapoint for OK -> ALARM transition).",
            "reasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2019-10-02T17:04:40.985+0000\",\"startDate\":\"2019-10-02T16:59:00.000+0000\",
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":[99.50160229693434],
\"threshold\":50.0}",
            "timestamp": "2019-10-02T17:04:40.989+0000",
            "value": "ALARM"
        }
    }
}
```


Perubahan status untuk alarm matematika metrik

```
{
    "version": "0",
    "id": "2dde0eb1-528b-d2d5-9ca6-6d590caf2329",
    "detail-type": "CloudWatch Alarm State Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2019-10-02T17:20:48Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:TotalNetworkTrafficTooHigh"
    ],
    "detail": {
        "alarmName": "TotalNetworkTrafficTooHigh",
        "configuration": {
            "description": "Goes into alarm if total network traffic exceeds 10Kb",
            "metrics": [
                {
                    "expression": "SUM(METRICS())",
                    "id": "e1",
                    "label": "Total Network Traffic",
                    "returnData": true
                },
                {
                    "id": "m1",
                    "metricStat": {
                        "metric": {
                            "dimensions": {
                                "InstanceId": "i-12345678901234567"
                            },
                            "name": "NetworkIn",
                            "namespace": "AWS/EC2"
                        },
                        "period": 300,
                        "stat": "Maximum"
                    },
                    "returnData": false
                },
                {
                    "id": "m2",
                    "metricStat": {
                        "metric": {
                            "dimensions": {
                                "InstanceId": "i-12345678901234567"
```

```
                            },
                            "name": "NetworkOut",
                            "namespace": "AWS/EC2"
                        },
                        "period": 300,
                        "stat": "Maximum"
                    },
                    "returnData": false
                }
            ]
        },
        "previousState": {
            "reason": "Unchecked: Initial alarm creation",
            "timestamp": "2019-10-02T17:20:03.642+0000",
            "value": "INSUFFICIENT_DATA"
        },
        "state": {
            "reason": "Threshold Crossed: 1 out of the last 1 datapoints [45628.0
 (02/10/19 17:10:00)] was greater than the threshold (10000.0) (minimum 1 datapoint for
 OK -> ALARM transition).",
            "reasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2019-10-02T17:20:48.551+0000\",\"startDate\":\"2019-10-02T17:10:00.000+0000\",
\"period\":300,\"recentDatapoints\":[45628.0],\"threshold\":10000.0}",
            "timestamp": "2019-10-02T17:20:48.554+0000",
            "value": "ALARM"
        }
    }
}
```

Perubahan status untuk alarm deteksi anomali

```
{
    "version": "0",
    "id": "daafc9f1-bddd-c6c9-83af-74971fcfc4ef",
    "detail-type": "CloudWatch Alarm State Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2019-10-03T16:00:04Z",
    "region": "us-east-1",
    "resources": ["arn:aws:cloudwatch:us-east-1:123456789012:alarm:EC2 CPU Utilization
 Anomaly"],
    "detail": {
        "alarmName": "EC2 CPU Utilization Anomaly",
```

```
        "state": {
            "value": "ALARM",
            "reason": "Thresholds Crossed: 1 out of the last 1 datapoints [0.0
 (03/10/19 15:58:00)] was less than the lower thresholds [0.020599444741798756] or
 greater than the upper thresholds [0.3006915352732461] (minimum 1 datapoint for OK ->
 ALARM transition).",
            "reasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2019-10-03T16:00:04.650+0000\",\"startDate\":\"2019-10-03T15:58:00.000+0000\",
\"period\":60,\"recentDatapoints\":[0.0],\"recentLowerThresholds\":
[0.020599444741798756],\"recentUpperThresholds\":[0.3006915352732461]}",
            "timestamp": "2019-10-03T16:00:04.653+0000"
        },
        "previousState": {
            "value": "OK",
            "reason": "Thresholds Crossed: 1 out of the last 1 datapoints
 [0.166666666664241 (03/10/19 15:57:00)] was not less than the lower thresholds
 [0.0206719426210418] or not greater than the upper thresholds [0.30076870222143803]
 (minimum 1 datapoint for ALARM -> OK transition).",
            "reasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2019-10-03T15:59:04.670+0000\",\"startDate\":\"2019-10-03T15:57:00.000+0000\",
\"period\":60,\"recentDatapoints\":[0.166666666664241],\"recentLowerThresholds\":
[0.0206719426210418],\"recentUpperThresholds\":[0.30076870222143803]}",
            "timestamp": "2019-10-03T15:59:04.672+0000"
        },
        "configuration": {
            "description": "Goes into alarm if CPU Utilization is out of band",
            "metrics": [{
                "id": "m1",
                "metricStat": {
                    "metric": {
                        "namespace": "AWS/EC2",
                        "name": "CPUUtilization",
                        "dimensions": {
                            "InstanceId": "i-12345678901234567"
                        }
                    },
                    "period": 60,
                    "stat": "Average"
                },
                "returnData": true
            }, {
                "id": "ad1",
                "expression": "ANOMALY_DETECTION_BAND(m1, 0.8)",
                "label": "CPUUtilization (expected)",
```

```
                "returnData": true
            }]
        }
    }
}
```

Perubahan status untuk alarm gabungan dengan alarm penekan

```
{
    "version": "0",
    "id": "d3dfc86d-384d-24c8-0345-9f7986db0b80",
    "detail-type": "CloudWatch Alarm State Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2022-07-22T15:57:45Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServiceAggregatedAlarm"
    ],
    "detail": {
        "alarmName": "ServiceAggregatedAlarm",
        "state": {
            "actionsSuppressedBy": "WaitPeriod",
            "actionsSuppressedReason": "Actions suppressed by WaitPeriod",
            "value": "ALARM",
            "reason": "arn:aws:cloudwatch:us-
east-1:123456789012:alarm:SuppressionDemo.EventBridge.FirstChild transitioned to ALARM
 at Friday 22 July, 2022 15:57:45 UTC",
            "reasonData": "{\"triggeringAlarms\":[{\"arn\":\"arn:aws:cloudwatch:us-
east-1:123456789012:alarm:ServerCpuTooHigh\",\"state\":{\"value\":\"ALARM\",\"timestamp
\":\"2022-07-22T15:57:45.394+0000\"}}]}",
            "timestamp": "2022-07-22T15:57:45.394+0000"
        },
        "previousState": {
            "value": "OK",
            "reason": "arn:aws:cloudwatch:us-
east-1:123456789012:alarm:SuppressionDemo.EventBridge.Main was created and its alarm
 rule evaluates to OK",
            "reasonData": "{\"triggeringAlarms\":[{\"arn\":\"arn:aws:cloudwatch:us-
east-1:123456789012:alarm:TotalNetworkTrafficTooHigh\",\"state\":{\"value\":\"OK\",
\"timestamp\":\"2022-07-14T16:28:57.770+0000\"}},{\"arn\":\"arn:aws:cloudwatch:us-
east-1:123456789012:alarm:ServerCpuTooHigh\",\"state\":{\"value\":\"OK\",\"timestamp\":
\"2022-07-14T16:28:54.191+0000\"}}]}",
```

```
            "timestamp": "2022-07-22T15:56:14.552+0000"
        },
        "configuration": {
            "alarmRule": "ALARM(ServerCpuTooHigh) OR
 ALARM(TotalNetworkTrafficTooHigh)",
            "actionsSuppressor": "ServiceMaintenanceAlarm",
            "actionsSuppressorWaitPeriod": 120,
            "actionsSuppressorExtensionPeriod": 180
        }
    }
}
```

Pembuatan alarm gabungan

```
{

    "version": "0",
    "id": "91535fdd-1e9c-849d-624b-9a9f2b1d09d0",
    "detail-type": "CloudWatch Alarm Configuration Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2022-03-03T17:06:22Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServiceAggregatedAlarm"
    ],
    "detail": {
        "alarmName": "ServiceAggregatedAlarm",
        "operation": "create",
        "state": {
            "value": "INSUFFICIENT_DATA",
            "timestamp": "2022-03-03T17:06:22.289+0000"
        },
        "configuration": {
            "alarmRule": "ALARM(ServerCpuTooHigh) OR
 ALARM(TotalNetworkTrafficTooHigh)",
            "alarmName": "ServiceAggregatedAlarm",
            "description": "Aggregated monitor for instance",
            "actionsEnabled": true,
            "timestamp": "2022-03-03T17:06:22.289+0000",
            "okActions": [],
            "alarmActions": [],
            "insufficientDataActions": []
```

```
        }
    }
}
```

Pembuatan alarm gabungan dengan alarm penekan

```
{
    "version": "0",
    "id": "454773e1-09f7-945b-aa2c-590af1c3f8e0",
    "detail-type": "CloudWatch Alarm Configuration Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2022-07-14T13:59:46Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServiceAggregatedAlarm"
    ],
    "detail": {
        "alarmName": "ServiceAggregatedAlarm",
        "operation": "create",
        "state": {
            "value": "INSUFFICIENT_DATA",
            "timestamp": "2022-07-14T13:59:46.425+0000"
        },
        "configuration": {
            "alarmRule": "ALARM(ServerCpuTooHigh) OR
 ALARM(TotalNetworkTrafficTooHigh)",
            "actionsSuppressor": "ServiceMaintenanceAlarm",
            "actionsSuppressorWaitPeriod": 120,
            "actionsSuppressorExtensionPeriod": 180,
            "alarmName": "ServiceAggregatedAlarm",
            "actionsEnabled": true,
            "timestamp": "2022-07-14T13:59:46.425+0000",
            "okActions": [],
            "alarmActions": [],
            "insufficientDataActions": []
        }
    }
}
```

Pembaruan alarm metrik

```
{
```

```
    "version": "0",
    "id": "bc7d3391-47f8-ae47-f457-1b4d06118d50",
    "detail-type": "CloudWatch Alarm Configuration Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2022-03-03T17:06:34Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServerCpuTooHigh"
    ],
    "detail": {
        "alarmName": "ServerCpuTooHigh",
        "operation": "update",
        "state": {
            "value": "INSUFFICIENT_DATA",
            "timestamp": "2022-03-03T17:06:13.757+0000"
        },
        "configuration": {
            "evaluationPeriods": 1,
            "threshold": 80,
            "comparisonOperator": "GreaterThanThreshold",
            "treatMissingData": "ignore",
            "metrics": [
                {
                    "id": "86bfa85f-b14c-ebf7-8916-7da014ce23c0",
                    "metricStat": {
                        "metric": {
                            "namespace": "AWS/EC2",
                            "name": "CPUUtilization",
                            "dimensions": {
                                "InstanceId": "i-12345678901234567"
                            }
                        },
                        "period": 300,
                        "stat": "Average"
                    },
                    "returnData": true
                }
            ],
            "alarmName": "ServerCpuTooHigh",
            "description": "Goes into alarm when server CPU utilization is too high!",
            "actionsEnabled": true,
            "timestamp": "2022-03-03T17:06:34.267+0000",
```

```
            "okActions": [],
            "alarmActions": [],
            "insufficientDataActions": []
        },
        "previousConfiguration": {
            "evaluationPeriods": 1,
            "threshold": 70,
            "comparisonOperator": "GreaterThanThreshold",
            "treatMissingData": "ignore",
            "metrics": [
                {
                    "id": "d6bfa85f-893e-b052-a58b-4f9295c9111a",
                    "metricStat": {
                        "metric": {
                            "namespace": "AWS/EC2",
                            "name": "CPUUtilization",
                            "dimensions": {
                                "InstanceId": "i-12345678901234567"
                            }
                        },
                        "period": 300,
                        "stat": "Average"
                    },
                    "returnData": true
                }
            ],
            "alarmName": "ServerCpuTooHigh",
            "description": "Goes into alarm when server CPU utilization is too high!",
            "actionsEnabled": true,
            "timestamp": "2022-03-03T17:06:13.757+0000",
            "okActions": [],
            "alarmActions": [],
            "insufficientDataActions": []
        }
    }
}
```

Pembaruan alarm gabungan dengan alarm penekan

```
{
    "version": "0",
    "id": "4c6f4177-6bd5-c0ca-9f05-b4151c54568b",
    "detail-type": "CloudWatch Alarm Configuration Change",
```

```
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2022-07-14T13:59:56Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServiceAggregatedAlarm"
    ],
    "detail": {
        "alarmName": "ServiceAggregatedAlarm",
        "operation": "update",
        "state": {
            "actionsSuppressedBy": "WaitPeriod",
            "value": "ALARM",
            "timestamp": "2022-07-14T13:59:46.425+0000"
        },
        "configuration": {
            "alarmRule": "ALARM(ServerCpuTooHigh) OR
 ALARM(TotalNetworkTrafficTooHigh)",
            "actionsSuppressor": "ServiceMaintenanceAlarm",
            "actionsSuppressorWaitPeriod": 120,
            "actionsSuppressorExtensionPeriod": 360,
            "alarmName": "ServiceAggregatedAlarm",
            "actionsEnabled": true,
            "timestamp": "2022-07-14T13:59:56.290+0000",
            "okActions": [],
            "alarmActions": [],
            "insufficientDataActions": []
        },
        "previousConfiguration": {
            "alarmRule": "ALARM(ServerCpuTooHigh) OR
 ALARM(TotalNetworkTrafficTooHigh)",
            "actionsSuppressor": "ServiceMaintenanceAlarm",
            "actionsSuppressorWaitPeriod": 120,
            "actionsSuppressorExtensionPeriod": 180,
            "alarmName": "ServiceAggregatedAlarm",
            "actionsEnabled": true,
            "timestamp": "2022-07-14T13:59:46.425+0000",
            "okActions": [],
            "alarmActions": [],
            "insufficientDataActions": []
        }
    }
}
```

Penghapusan alarm matematika metrik


```
{

    "version": "0",
    "id": "f171d220-9e1c-c252-5042-2677347a83ed",
    "detail-type": "CloudWatch Alarm Configuration Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2022-03-03T17:07:13Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:TotalNetworkTrafficTooHigh"
    ],
    "detail": {
        "alarmName": "TotalNetworkTrafficTooHigh",
        "operation": "delete",
        "state": {
            "value": "INSUFFICIENT_DATA",
            "timestamp": "2022-03-03T17:06:17.672+0000"
        },
        "configuration": {
            "evaluationPeriods": 1,
            "threshold": 10000,
            "comparisonOperator": "GreaterThanThreshold",
            "treatMissingData": "ignore",
            "metrics": [{
                    "id": "m1",
                    "metricStat": {
                        "metric": {
                            "namespace": "AWS/EC2",
                            "name": "NetworkIn",
                            "dimensions": {
                                "InstanceId": "i-12345678901234567"
                            }
                        },
                        "period": 300,
                        "stat": "Maximum"
                    },
                    "returnData": false
                },
                {
                    "id": "m2",
                    "metricStat": {
```

```
                        "metric": {
                            "namespace": "AWS/EC2",
                            "name": "NetworkOut",
                            "dimensions": {
                                "InstanceId": "i-12345678901234567"
                            }
                        },
                        "period": 300,
                        "stat": "Maximum"
                    },
                    "returnData": false
                },
                {
                    "id": "e1",
                    "expression": "SUM(METRICS())",
                    "label": "Total Network Traffic",
                    "returnData": true
                }
            ],
            "alarmName": "TotalNetworkTrafficTooHigh",
            "description": "Goes into alarm if total network traffic exceeds 10Kb",
            "actionsEnabled": true,
            "timestamp": "2022-03-03T17:06:17.672+0000",
            "okActions": [],
            "alarmActions": [],
            "insufficientDataActions": []

        }
    }
}
```

Penghapusan alarm gabungan dengan alarm penekan

```
{
    "version": "0",
    "id": "e34592a1-46c0-b316-f614-1b17a87be9dc",
    "detail-type": "CloudWatch Alarm Configuration Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2022-07-14T14:00:01Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServiceAggregatedAlarm"
```

```
    ],
    "detail": {
        "alarmName": "ServiceAggregatedAlarm",
        "operation": "delete",
        "state": {
            "actionsSuppressedBy": "WaitPeriod",
            "value": "ALARM",
            "timestamp": "2022-07-14T13:59:46.425+0000"
        },
        "configuration": {
            "alarmRule": "ALARM(ServerCpuTooHigh) OR
 ALARM(TotalNetworkTrafficTooHigh)",
            "actionsSuppressor": "ServiceMaintenanceAlarm",
            "actionsSuppressorWaitPeriod": 120,
            "actionsSuppressorExtensionPeriod": 360,
            "alarmName": "ServiceAggregatedAlarm",
            "actionsEnabled": true,
            "timestamp": "2022-07-14T13:59:56.290+0000",
            "okActions": [],
            "alarmActions": [],
            "insufficientDataActions": []
        }
    }
}
```

Mengelola alarm-alarm

Mengedit atau menghapus CloudWatch alarm

Anda dapat menyunting atau menghapus sebuah alarm yang sudah ada.

Anda tidak dapat mengubah nama dari sebuah alarm yang sudah ada. Anda dapat menyalin sebuah alarm dan memberikan nama baru yang berbeda kepada alarm tersebut. Untuk menyalin sebuah alarm, Anda harus memilih kotak centang yang ada di samping nama alarm pada daftar alarm dan pilih Tindakan, Salin.

Cara menyunting sebuah alarm

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Alarm, Semua Alarm.
3. Pilih nama alarm.

4. Untuk menambah atau menghapus tag, pilih tab Tanda dan kemudian pilih Kelola tag.
5. Untuk menyunting bagian lain dari alarm tersebut, silakan pilih Tindakan, Sunting.

   Halaman Tentukan metrik dan ketentuan ditampilkan, di sana ditampilkan grafik dan informasi lain tentang metrik dan statistik yang Anda pilih.
6. Untuk mengubah metrik, pilih Sunting, pilih tab Semua metrik, dan kemudian Anda lakukan salah satu hal berikut:
   - Anda harus memilih namespace layanan yang memuat metrik yang Anda inginkan. Kemudian Anda harus memilih pilihan yang tampaknya akan mempersempit pilihan. Ketika ada daftar metrik yang muncul, pilihlah kotak centang yang ada di samping metrik yang Anda inginkan.
   - Pada kotak pencarian, Anda harus memasukkan nama metrik, dimensi, atau ID sumber daya dan tekan Enter. Dan kemudian Anda harus memilih salah satu hasil pencarian dan melanjutkan hingga daftar metrik muncul. Pilihlah kotak centang yang ada di samping metrik yang Anda inginkan.

   Pilih Pilih Metrik.
7. Untuk mengubah aspek alarm lainnya, silakan pilih pilihan yang sesuai. Untuk mengubah seberapa banyak titik data yang harus dilanggar sehingga alarm dapat beralih statusnya menjadi `ALARM` atau mengubah cara memperlakukan data yang hilang, silakan pilih Konfigurasi tambahan.
8. Pilih Berikutnya.
9. Pada Notifikasi, Tindakan penskalaan otomatis (Auto Scaling), dan Tindakan EC2, secara opsional, Anda bisa menyunting tindakan yang diambil ketika alarm dipicu. Lalu pilih Berikutnya.
10. Secara opsional, Anda bisa mengubah deskripsi alarm.

    Anda tidak dapat mengubah nama dari sebuah alarm yang sudah ada. Anda dapat menyalin sebuah alarm dan memberikan nama baru yang berbeda kepada alarm tersebut. Untuk menyalin sebuah alarm, Anda harus memilih kotak centang yang ada di samping nama alarm pada daftar alarm dan pilih Tindakan, Salin.
11. Pilih Berikutnya.
12. Pada Pratinjau dan buat, konfirmasikan bahwa informasi dan ketentuannya sudah sesuai keinginan Anda, kemudian pilih Perbarui alarm.

Untuk memperbarui sebuah daftar notifikasi email yang dibuat dengan menggunakan konsol Amazon SNS

1. Buka konsol Amazon SNS di https://console.aws.amazon.com/sns/v3/home.
2. Pada panel navigasi, silakan pilih Topik dan kemudian pilih ARN untuk daftar notifikasi Anda (topik).
3. Lakukan salah satu langkah berikut:
   - Untuk menambahkan sebuah alamat email, pilih Buat berlangganan. Untuk Protokol, pilih Email. Untuk Titik Akhir, masukkan alamat email penerima baru. Pilih Buat langganan.
   - Untuk menghapus sebuah alamat email, pilih ID Berlangganan. Pilih Tindakan berlangganan lainnya, Hapus berlangganan.
4. Pilih Terbitkan ke topik.

Cara menghapus sebuah alarm

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Alarm.
3. Pilih kotak centang yang ada di sebelah kiri nama alarm, dan pilih Tindakan, Hapus.
4. Pilih Hapus.

Sembunyikan alarm Auto Scaling

Saat Anda melihat alarm di AWS Management Console, Anda dapat menyembunyikan alarm yang terkait dengan Auto Scaling Amazon EC2 dan Application Auto Scaling. Fitur ini hanya tersedia di AWS Management Console.

Untuk menyembunyikan alarm penskalaan otomatis (Auto Scaling) sementara waktu

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Alarm, Semua alarm, dan pilih Sembunyikan alarm penskalaan otomatis (Auto Scaling).

Kasus dan contoh-contoh penggunaan alarm

Bagian berikut akan memberikan contoh dan tutorial untuk alarm untuk kasus penggunaan umum.

Buat alarm penagihan untuk memantau perkiraan AWS biaya Anda

Anda dapat memantau perkiraan AWS biaya Anda dengan menggunakan Amazon CloudWatch. Saat Anda mengaktifkan pemantauan estimasi biaya untuk AWS akun Anda, perkiraan biaya dihitung dan dikirim beberapa kali sehari CloudWatch sebagai data metrik.

Data metrik penagihan kemudian akan disimpan di Wilayah AS Timur (Virginia Utara) dan mewakili biaya di seluruh dunia. Data ini mencakup perkiraan biaya untuk setiap layanan AWS yang Anda gunakan, selain perkiraan total keseluruhan AWS biaya Anda.

Alarm ini akan memicu ketika penagihan akun Anda melebihi ambang batas yang Anda tentukan. Alarm ini hanya akan memicu ketika penagihan berjalan melampaui ambang batas yang ditentukan. Alarm ini tidak menggunakan proyeksi berdasarkan pada penggunaan Anda sejauh ini di bulan tersebut.

Jika Anda membuat sebuah alarm penagihan pada waktu biaya Anda telah melebihi ambang batas yang ditentukan, maka alarm akan beralih statusnya menjadi `ALARM` pada saat itu juga.

> **Note**
>
> Untuk informasi tentang menganalisis CloudWatch biaya yang telah ditagih untuk Anda, lihatCloudWatch penagihan dan biaya.

Tugas

- Mengaktifkan peringatan penagihan
- Membuat sebuah alarm penagihan
- Menghapus sebuah alarm penagihan

Mengaktifkan peringatan penagihan

Sebelum Anda dapat membuat alarm untuk perkiraan tagihan, Anda harus mengaktifkan peringatan penagihan, sehingga Anda dapat memantau perkiraan AWS biaya dan membuat alarm menggunakan data metrik penagihan. Setelah Anda mengaktifkan peringatan penagihan, Anda tidak akan dapat menonaktifkan pengumpulan data, tetapi Anda akan dapat menghapus setiap alarm penagihan yang telah dibuat.

Setelah Anda mengaktifkan peringatan penagihan untuk pertama kalinya, diperlukan waktu sekitar 15 menit sebelum Anda dapat melihat data penagihan dan mengatur alarm penagihan tersebut.

Persyaratan

- Anda harus masuk menggunakan kredensial pengguna root akun atau sebagai pengguna IAM yang telah mendapatkan izin untuk melihat informasi penagihan.
- Untuk akun penagihan terkonsolidasi, data penagihan untuk masing-masing akun yang dikaitkan dapat Anda temukan dengan cara masuk sebagai akun pembayaran. Anda dapat melihat data penagihan untuk total estimasi biaya dan estimasi biaya berdasarkan layanan untuk setiap akun yang dikaitkan, selain akun konsolidasian tersebut.
- Dalam sebuah akun penagihan konsolidasian, metrik akun yang dikaitkan anggota hanya akan diambil datanya jika akun pembayar mengaktifkan pilihan Terima Pemberitahuan Penagihan. Jika Anda mengubah akun yang merupakan akun manajemen/pembayar, maka Anda harus mengaktifkan peringatan penagihan pada akun manajemen/pembayar yang baru.
- Akun tidak boleh menjadi bagian dari Jaringan Mitra Amazon (APN) karena metrik penagihan tidak dipublikasikan CloudWatch untuk akun APN. Untuk informasi selengkapnya mengenai hal ini, silakan lihat Jaringan Partner AWS.

Cara mengaktifkan pemantauan estimasi biaya

1. Buka AWS Billing konsol di https://console.aws.amazon.com/billing/.
2. Pada panel navigasi, silakan pilih Preferensi Penagihan.
3. Pada Preferensi Peringatan pilih Sunting.
4. Pilih Terima Pemberitahuan CloudWatch Penagihan.
5. Pilih Simpan preferensi.

Membuat sebuah alarm penagihan

> **Important**
>
> Sebelum Anda membuat sebuah alarm penagihan, Anda harus mengatur Wilayah Anda ke AS Timur (Virginia Utara). Data metrik penagihan akan disimpan di Wilayah ini dan mewakili biaya di seluruh dunia. Anda juga harus mengaktifkan peringatan penagihan

untuk akun Anda atau di akun manajemen/pembayar (jika Anda menggunakan penagihan konsolidasian). Untuk informasi selengkapnya, lihat Mengaktifkan peringatan penagihan.

Dalam prosedur ini, Anda membuat alarm yang mengirimkan pemberitahuan ketika perkiraan biaya Anda AWS melebihi ambang batas yang ditentukan.

Untuk membuat alarm penagihan menggunakan konsol CloudWatch

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Alarm, dan kemudian pilih Semua alarm.
3. Pilih Buat alarm.
4. Pilih Pilih Metrik. Pada Jelajah, pilih Penagihan, kemudian pilih Total Estimasi Biaya.

   > **Note**
   >
   > Jika Anda tidak melihat metrik Penagihan/Total Estimasi Biaya, aktifkan peringatan penagihan, dan ubah Wilayah Anda ke AS Timur (Virginia Utara). Untuk informasi selengkapnya, lihat Mengaktifkan peringatan penagihan.

5. Pilih kotak untuk EstimatedChargesmetrik, lalu pilih Pilih metrik.
6. Untuk Statistik, pilih Maksimum.
7. Untuk Periode, pilih 6 jam.
8. Untuk Jenis ambang batas, pilih Statis.
9. Untuk Kapanpun EstimatedCharges . , pilih Lebih Besar.
10. Untuk daripada . . ., Anda harus menentukan nilai yang ingin Anda pilih untuk memicu alarm. Sebagai contoh, **`200`** USD.

    Nilai EstimatedChargesmetrik hanya dalam dolar AS (USD), dan konversi mata uang disediakan oleh Amazon Services LLC. Untuk informasi lebih lanjut, lihat Apa itu AWS Billing? .

    > **Note**
    >
    > Setelah Anda menentukan nilai ambang batas, grafik pratinjau akan menampilkan estimasi biaya untuk bulan berjalan.

11. Pilih Konfigurasi Tambahan dan lakukan hal-hal berikut:

- Untuk Titik data untuk alarm, tentukan 1 dari 1.
- Untuk Perlakuan data yang hilang, pilih Perlakukan data yang hilang sebagai hilang.

12. Pilih Berikutnya.
13. Pada Notifikasi, pastikan bahwa Dalam alarm yang dipilih. Kemudian Anda harus menentukan topik Amazon SNS yang akan dikirimkan notifikasinya saat alarm Anda berada dalam status `ALARM`. Topik Amazon SNS tersebut bisa menyertakan alamat email Anda sehingga Anda akan menerima email saat jumlah penagihan melewati ambang batas yang Anda tentukan.

    Anda dapat memilih topik Amazon SNS yang sudah ada, membuat topik Amazon SNS yang baru, atau menggunakan topik ARN untuk memberikan notifikasi ke akun lain. Jika Anda ingin alarm mengirimkan beberapa notifikasi untuk status alarm yang sama atau untuk status alarm yang berbeda, silakan pilih Tambahkan notifikasi.
14. Pilih Berikutnya.
15. Pada Nama dan deskripsi, masukkan nama untuk alarm Anda. Nama tersebut harus menggunakan karakter UTF-8, dan tidak dapat berisi karakter kontrol ASCII.

    - (Opsional) Masukkan deskripsi untuk alarm Anda. Deskripsi dapat mencakup pemformatan penurunan harga, yang hanya ditampilkan di tab Detail alarm di CloudWatch konsol. Penurunan harga dapat Anda gunakan untuk menambahkan tautan ke runbook atau sumber daya internal lainnya.
16. Pilih Berikutnya.
17. Pada Pratinjau dan buat, periksa apakah konfigurasi Anda sudah benar, kemudian pilih Buat alarm.

Menghapus sebuah alarm penagihan

Anda dapat menghapus sebuah alarm penagihan ketika Anda tidak lagi membutuhkannya.

Untuk menghapus alarm penagihan

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Jika perlu, ubah Wilayah ke AS Timur (Virginia Utara). Data metrik penagihan akan disimpan di Wilayah ini dan ini akan mencerminkan biaya di seluruh dunia.
3. Pada panel navigasi, silakan pilih Alarm, Semua alarm.
4. Pilih kotak centang yang ada di samping alarm dan pilih Tindakan, Hapus.

5. Ketika diminta untuk mengonfirmasi, pilih Ya, Hapus.

Membuat sebuah alarm penggunaan CPU

Anda dapat membuat CloudWatch alarm yang mengirimkan pemberitahuan menggunakan Amazon SNS saat alarm berubah status dari `OK` ke. `ALARM`

Alarm tersebut akan berubah statusnya menjadi `ALARM` ketika penggunaan CPU rata-rata dari instans EC2 melebihi ambang batas yang telah ditentukan untuk periode tertentu secara berturut-turut.

Menyiapkan alarm penggunaan CPU menggunakan AWS Management Console

Gunakan langkah-langkah ini untuk menggunakan AWS Management Console untuk membuat alarm penggunaan CPU.

Cara membuat sebuah alarm yang didasarkan pada penggunaan CPU

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Alarm, Semua Alarm.
3. Pilih Buat alarm.
4. Pilih Pilih Metrik.
5. Pada tab Semua metrik, pilih metrik EC2.
6. Pilih kategori metrik (misalnya, Metrik Per-Instans).
7. Temukan baris dengan contoh yang ingin Anda cantumkan di InstanceIdkolom dan CPUUutilization di kolom Nama Metrik. Pilih kotak centang yang ada di samping baris ini, dan pilih Pilih metrik.
8. Pada Tetapkan metrik dan kondisi, untuk Statistik pilih Rata-rata, kemudian pilih salah satu persentil yang sudah ditentukan sebelumnya, atau Anda bisa menetapkan persentil kustom (misalnya, **`p95.45`**).
9. Pilih sebuah periode (misalnya, **`5 minutes`**).
10. Pada Ketentuan, tentukan hal-hal berikut:

    a. Untuk Jenis ambang batas, pilih Statis.

b. Untuk Setiap kali CPUUtilization adalah, tentukan Lebih besar. Pada dari..., tetapkan ambang batas untuk memicu alarm beralih statusnya menjadi ALARM ketika penggunaan CPU melebihi persentase ini. Sebagai contoh, 70.

c. Pilih Konfigurasi tambahan. Untuk Titik data alarm, tentukan berapa banyak periode evaluasi (titik data) yang harus ada dalam status `ALARM` untuk memicu alarm. Jika kedua nilai di sini cocok, Anda membuat alarm yang beralih ke status `ALARM` jika terjadi pelanggaran selama sebanyak itu dalam periode berturut-turut.

   Untuk membuat alarm M dari N, tentukan angka yang lebih rendah untuk nilai pertama dibandingkan dengan nilai yang Anda tentukan untuk nilai kedua. Untuk informasi selengkapnya, lihat Melakukan evaluasi alarm.

d. Untuk Perlakuan data yang hilang, pilih cara alarm berperilaku ketika beberapa titik data hilang. Untuk informasi selengkapnya, lihat Mengkonfigurasi bagaimana CloudWatch alarm memperlakukan data yang hilang.

e. Jika alarm menggunakan sebuah persentil sebagai statistik yang dipantau, maka kotak Persentil dengan sampel rendah akan ditampilkan. Pilih apakah Anda akan melakukan evaluasi atau mengabaikan kasus yang memiliki tingkat sampel yang kecil. Jika Anda memilih abaikan (status alarm tidak berubah), maka status alarm saat ini akan tetap dipertahankan ketika ukuran sampel terlalu kecil. Untuk informasi selengkapnya, lihat CloudWatch Alarm berbasis persentil dan sampel data rendah.

11. Pilih Berikutnya.
12. Pada Notifikasi, pilih Dalam alarm dan kemudian pilih sebuah topik SNS yang akan diberitahukan sebagai notifikasi ketika alarm berada dalam status `ALARM`

    Agar alarm mengirimkan beberapa notifikasi untuk status alarm yang sama atau untuk status alarm yang berbeda, silakan pilih Tambahkan notifikasi.

    Agar alarm tidak mengirim notifikasi, silakan pilih Hapus.
13. Setelah selesai, silakan pilih Berikutnya.
14. Masukkan sebuah nama dan deskripsi untuk alarm tersebut. Lalu pilih Berikutnya.

    Nama tersebut harus menggunakan karakter UTF-8, dan tidak dapat berisi karakter kontrol ASCII. Deskripsi dapat mencakup pemformatan penurunan harga, yang hanya ditampilkan di tab Detail alarm di CloudWatch konsol. Penurunan harga dapat Anda gunakan untuk menambahkan tautan ke runbook atau sumber daya internal lainnya.

15. Pada Pratinjau dan buat, konfirmasikan bahwa informasi dan ketentuannya sudah sesuai keinginan Anda, kemudian pilih Buat alarm.

Menyiapkan alarm penggunaan CPU menggunakan AWS CLI

Gunakan langkah-langkah ini untuk menggunakan AWS CLI untuk membuat alarm penggunaan CPU.

Cara membuat sebuah alarm yang didasarkan pada penggunaan CPU

1. Menyiapkan sebuah topik SNS. Untuk informasi selengkapnya, lihat Menyiapkan notifikasi Amazon SNS.
2. Buat alarm menggunakan put-metric-alarmperintah sebagai berikut.

```
aws cloudwatch put-metric-alarm --alarm-name cpu-mon --alarm-description "Alarm
 when CPU exceeds 70%" --metric-name CPUUtilization --namespace AWS/EC2 --statistic
 Average --period 300 --threshold 70 --comparison-operator GreaterThanThreshold --
dimensions  Name=InstanceId,Value=i-12345678 --evaluation-periods 2 --alarm-actions
 arn:aws:sns:us-east-1:111122223333:my-topic --unit Percent
```

3. Uji alarm dengan memaksa perubahan status alarm menggunakan set-alarm-stateperintah.

   a. Ubah status alarm dari `INSUFFICIENT_DATA` menjadi `OK`.

   ```
   aws cloudwatch set-alarm-state --alarm-name cpu-mon --state-reason
    "initializing" --state-value OK
   ```

   b. Ubah status alarm dari `OK` menjadi `ALARM`.

   ```
   aws cloudwatch set-alarm-state --alarm-name cpu-mon --state-reason
    "initializing" --state-value ALARM
   ```

   c. Pastikan bahwa Anda telah menerima notifikasi tentang alarm tersebut.

Buat sebuah alarm latensi penyeimbang beban yang mengirim email

Anda dapat menyiapkan sebuah notifikasi Amazon SNS dan mengonfigurasi sebuah alarm yang akan memantau latensi yang melebihi 100 ms untuk Penyeimbang Beban Klasik Anda.

Menyiapkan alarm latensi menggunakan AWS Management Console

Gunakan langkah-langkah ini untuk menggunakan AWS Management Console alarm latensi penyeimbang beban.

Cara membuat sebuah alarm latensi penyeimbang beban

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Alarm, Semua Alarm.
3. Pilih Buat alarm.
4. Di bawah CloudWatch Metrik berdasarkan Kategori, pilih kategori Metrik ELB.
5. Pilih baris dengan Penyeimbang Beban Klasik dan metrik Latensi.
6. Untuk statistik pilih Rata-rata, kemudian pilih salah satu persentil yang sudah ditentukan sebelumnya, atau Anda bisa menetapkan persentil kustom (misalnya, **`p95.45`**).
7. Untuk periodenya, silakan pilih 1 Menit.
8. Pilih Berikutnya.
9. Pada Ambang Batas Alarm, masukkan nama unik untuk alarm (misalnya, **`myHighCpuAlarm`**) dan deskripsi alarm (misalnya, **`Alarm when Latency exceeds 100s`**). Nama alarm harus menggunakan karakter UTF-8, dan tidak dapat berisi karakter kontrol ASCII

   Nama tersebut harus menggunakan karakter UTF-8, dan tidak dapat berisi karakter kontrol ASCII. Deskripsi dapat mencakup pemformatan penurunan harga, yang hanya ditampilkan di tab Detail alarm di CloudWatch konsol. Penurunan harga dapat Anda gunakan untuk menambahkan tautan ke runbook atau sumber daya internal lainnya.
10. Pada Kapan pun, untuk adalah, pilih **>** dan masukkan **`0.1`**. Untuk for, masukkan **`3`**.
11. Pada Pengaturan tambahan, untuk Perlakukan data yang hilang sebagai, pilih abaikan (status alarm dipertahankan) sehingga tidak ada titik data yang hilang tidak akan memicu perubahan status alarm.

    Untuk Persentil dengan sampel kecil, pilih abaikan (pertahankan status alarm) sehingga alarm hanya akan mengevaluasi situasi dengan jumlah sampel data yang memadai.
12. Pada Tindakan, untuk Setiap kali alarm ini, pilih Berada dalam status ALARM. Untuk Kirim notifikasi ke, pilih topik SNS yang sudah ada atau buat topik baru.

    Cara membuat sebuah topik SNS, pilih Daftar baru. Untuk Kirim notifikasi ke, masukkan nama untuk topik SNS (misalnya, **`myHighCpuAlarm`**), dan untuk Daftar email, masukkan daftar alamat

email yang dipisahkan dengan tanda koma yang akan mendapatkan notifikasi ketika alarm statusnya beralih menjadi ALARM. Masing-masing alamat email akan mendapatkan sebuah email konfirmasi untuk berlangganan topik. Anda harus mengonfirmasi berlangganan tersebut sebelum notifikasi dapat dikirim.

13. Pilih Buat alarm.

Menyiapkan alarm latensi menggunakan AWS CLI

Gunakan langkah-langkah ini untuk menggunakan AWS CLI alarm latensi penyeimbang beban.

Cara membuat sebuah alarm latensi penyeimbang beban

1. Menyiapkan sebuah topik SNS. Untuk informasi selengkapnya, lihat Menyiapkan notifikasi Amazon SNS.
2. Buat alarm menggunakan put-metric-alarmperintah sebagai berikut:

```
aws cloudwatch put-metric-alarm --alarm-name lb-mon --alarm-description "Alarm
 when Latency exceeds 100s" --metric-name Latency --namespace AWS/ELB --statistic
 Average --period 60 --threshold 100 --comparison-operator GreaterThanThreshold --
dimensions Name=LoadBalancerName,Value=my-server --evaluation-periods 3 --alarm-
actions arn:aws:sns:us-east-1:111122223333:my-topic --unit Seconds
```

3. Uji alarm dengan memaksa perubahan status alarm menggunakan set-alarm-stateperintah.

   a. Ubah status alarm dari INSUFFICIENT_DATA menjadi OK.

   ```
   aws cloudwatch set-alarm-state --alarm-name lb-mon --state-reason
    "initializing" --state-value OK
   ```

   b. Ubah status alarm dari OK menjadi ALARM.

   ```
   aws cloudwatch set-alarm-state --alarm-name lb-mon --state-reason
    "initializing" --state-value ALARM
   ```

   c. Periksa apakah Anda telah menerima sebuah email notifikasi tentang alarm tersebut.

Membuat sebuah alarm throughput penyimpanan yang mengirim email

Anda dapat menyetel sebuah notifikasi SNS dan mengonfigurasikan sebuah alarm yang akan memicu ketika Amazon EBS melebihi throughput 100 MB.

Menyiapkan sebuah alarm throughput penyimpanan dengan menggunakan AWS Management Console

Gunakan langkah-langkah ini untuk menggunakan AWS Management Console untuk membuat alarm berdasarkan throughput Amazon EBS.

Cara membuat sebuah alarm throughput penyimpanan

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Alarm, Semua Alarm.
3. Pilih Buat alarm.
4. Pada Metrik EBS, pilih kategori metrik.
5. Pilih baris dengan volume dan VolumeWriteBytesmetrik.
6. Untuk statistik, silakan pilih Rata-rata. Untuk periodenya, silakan pilih 5 Menit. Pilih Berikutnya.
7. Pada Ambang Batas Alarm, masukkan nama unik untuk alarm (misalnya, **`myHighWriteAlarm`**) dan deskripsi alarm (misalnya, **`VolumeWriteBytes exceeds 100,000 KiB/s`**). Nama tersebut harus menggunakan karakter UTF-8, dan tidak dapat berisi karakter kontrol ASCII. Deskripsi dapat mencakup pemformatan penurunan harga, yang hanya ditampilkan di tab Detail alarm di CloudWatch konsol. Penurunan harga dapat Anda gunakan untuk menambahkan tautan ke runbook atau sumber daya internal lainnya.
8. Pada Kapan pun, untuk adalah, pilih **>** dan masukkan **`100000`**. Untuk for, masukkan **`15`** titik-titik berturut-turut.

   Pernyataan grafis dari ambang batas tersebut akan ditampilkan pada Pratinjau Alarm.
9. Pada Pengaturan tambahan, untuk Perlakukan data yang hilang sebagai, pilih abaikan (status alarm dipertahankan) sehingga tidak ada titik data yang hilang tidak akan memicu perubahan status alarm.
10. Pada Tindakan, untuk Setiap kali alarm ini, pilih Berada dalam status ALARM. Untuk Kirim notifikasi ke, pilih topik SNS yang sudah ada atau buat topik yang baru.

    Cara membuat sebuah topik SNS, pilih Daftar baru. Untuk Kirim notifikasi ke, masukkan nama untuk topik SNS (misalnya, **`myHighCpuAlarm`**), dan untuk Daftar email, masukkan daftar alamat

email yang dipisahkan dengan tanda koma yang akan mendapatkan notifikasi ketika alarm statusnya beralih menjadi ALARM. Masing-masing alamat email akan mendapatkan sebuah email konfirmasi untuk berlangganan topik. Anda harus mengonfirmasi berlangganan tersebut sebelum notifikasi dapat dikirimkan ke sebuah alamat email.

11. Pilih Buat alarm.

Menyiapkan alarm throughput penyimpanan menggunakan AWS CLI

Gunakan langkah-langkah ini untuk menggunakan AWS CLI untuk membuat alarm berdasarkan throughput Amazon EBS.

Cara membuat sebuah alarm throughput penyimpanan

1. Membuat sebuah topik SNS. Untuk informasi selengkapnya, lihat Menyiapkan notifikasi Amazon SNS.
2. Membuat alarm.

```
aws cloudwatch put-metric-alarm --alarm-name ebs-mon --alarm-description "Alarm
 when EBS volume exceeds 100MB throughput" --metric-name VolumeReadBytes --
namespace AWS/EBS --statistic Average --period 300 --threshold 100000000 --
comparison-operator GreaterThanThreshold --dimensions Name=VolumeId,Value=my-
volume-id --evaluation-periods 3 --alarm-actions arn:aws:sns:us-
east-1:111122223333:my-alarm-topic --insufficient-data-actions arn:aws:sns:us-
east-1:111122223333:my-insufficient-data-topic
```

3. Uji alarm dengan memaksa perubahan status alarm menggunakan set-alarm-stateperintah.

   a. Ubah status alarm dari INSUFFICIENT_DATA menjadi OK.

   ```
   aws cloudwatch set-alarm-state --alarm-name ebs-mon --state-reason
    "initializing" --state-value OK
   ```

   b. Ubah status alarm dari OK menjadi ALARM.

   ```
   aws cloudwatch set-alarm-state --alarm-name ebs-mon --state-reason
    "initializing" --state-value ALARM
   ```

   c. Ubah status alarm dari ALARM menjadi INSUFFICIENT_DATA.

```
aws cloudwatch set-alarm-state --alarm-name ebs-mon --state-reason
 "initializing" --state-value INSUFFICIENT_DATA
```

d. Periksa apakah Anda telah menerima sebuah email notifikasi tentang alarm tersebut.

Membuat alarm pada metrik penghitung Performance Insights dari database AWS

CloudWatch menyertakan fungsi matematika metrik DB_PERF_INSIGHTS yang dapat Anda gunakan untuk menghadirkan metrik penghitung Performance Insights dari Amazon Relational Database Service dan Amazon DocumentDB (dengan CloudWatch kompatibilitas MongoDB). DB_PERF_INSIGHTS juga akan membawa metrik `DBLoad` pada interval sub-menit. Anda dapat mengatur CloudWatch alarm pada metrik ini.

Untuk informasi selengkapnya tentang Wawasan Performa Amazon RDS, silakan lihat Memantau pemuatan DB dengan Wawasan Performa di Amazon RDS.

Untuk informasi selengkapnya tentang Amazon DocumentDB Wawasan Performa, silakan lihat Memantau dengan Wawasan Performa.

Deteksi anomali tidak didukung untuk alarm-alarm yang dibuat berdasarkan fungsi DB_PERF_INSIGHTS.

> **Note**
>
> Metrik-metrik resolusi tinggi dengan pedetail sub-menit yang diambil oleh DB_PERF_INSIGHTS hanya berlaku untuk metrik DBLoad, atau untuk metrik sistem operasi jika Anda telah mengaktifkan Pemantauan yang Ditingkatkan dengan resolusi yang lebih tinggi. Untuk informasi selengkapnya tentang pemantauan yang disempurnakan Amazon RDS, silakan lihat Memantau metrik OS dengan Pemantauan yang Ditingkatkan. .
> Anda dapat membuat alarm resolusi tinggi menggunakan fungsi DB_PERF_INSIGHTS. Rentang evaluasi maksimum untuk alarm resolusi tinggi adalah tiga jam. Anda dapat menggunakan CloudWatch konsol untuk membuat grafik metrik yang diambil dengan fungsi DB_PERF_INSIGHTS untuk rentang waktu apa pun.

Cara membuat sebuah alarm yang didasarkan pada metrik Wawasan Performa

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Alarm, dan kemudian pilih Semua alarm.
3. Pilih Buat alarm.
4. Pilih Pilih Metrik.
5. Pilih menu geser-turun Tambahkan matematika, lalu pilih Metrik Performa Basis Data, dari daftar DB_PERF_INSIGHTS.

   Setelah Anda memilih DB_PERF_INSIGHTS, kotak ekspresi matematika akan ditampilkan, Anda dapat menerapkan atau menyunting ekspresi matematika di sana.
6. Pada kotak ekspresi matematika tersebut, masukkan ekspresi matematika DB_PERF_INSIGHTS Anda, dan kemudian pilih Terapkan.

   Sebagai contoh, **`DB_PERF_INSIGHTS('RDS', 'db-ABCDEFGHIJKLMNOPQRSTUVWXY1', 'os.cpuUtilization.user.avg')`**.

   > **Important**
   >
   > Bila Anda menggunakan ekspresi matematika DB_PERF_INSIGHTS, Anda harus menentukan ID Sumber Daya Basis Data Unik dari basis data. Hal ini berbeda dengan pengidentifikasi basis data. Untuk menemukan ID sumber daya basis data di konsol Amazon RDS, pilih instans DB untuk melihat detailnya. Kemudian, pilih tab Konfigurasi. ID Sumber Daya ditampilkan di bagian Konfigurasi.

   Untuk informasi selengkapnya tentang fungsi DB_PERF_INSIGHTS dan fungsi-fungsi lainnya yang tersedia untuk matematika metrik, silakan lihat Sintaks dan fungsi matematika metrik.
7. Pilih Pilih Metrik.

   Halaman Tentukan metrik dan ketentuan ditampilkan, di sana akan ditampilkan grafik dan informasi lain tentang ekspresi matematika yang Anda pilih.
8. Untuk Setiap kali ***`ekspresi`*** adalah, Anda harus menentukan apakah metrik harus lebih besar dari, kurang dari, atau sama dengan ambang batas. Di bawah dari..., tentukan nilai ambang batas.
9. Pilih Konfigurasi tambahan. Untuk Titik data alarm, tentukan berapa banyak periode evaluasi (titik data) yang harus ada dalam status ALARM untuk memicu alarm. Jika kedua nilai di sini

cocok, Anda membuat alarm yang beralih ke status ALARM jika terjadi pelanggaran selama sebanyak itu dalam periode berturut-turut.

Untuk membuat alarm M dari N, tentukan angka yang lebih rendah untuk nilai pertama dibandingkan dengan nilai yang Anda tentukan untuk nilai kedua. Untuk informasi selengkapnya, lihat Melakukan evaluasi alarm.

10. Untuk Perlakuan data yang hilang, pilih cara alarm berperilaku ketika beberapa titik data hilang. Untuk informasi selengkapnya, lihat Mengkonfigurasi bagaimana CloudWatch alarm memperlakukan data yang hilang.
11. Pilih Berikutnya.
12. Pada Notifikasi, silakan pilih topik SNS yang akan diberitahukan sebagai notifikasi ketika alarm berada dalam status ALARM, status OK, atau status INSUFFICIENT_DATA.

    Agar alarm mengirimkan beberapa notifikasi untuk status alarm yang sama atau untuk status alarm yang berbeda, silakan pilih Tambahkan notifikasi.

    Agar alarm tidak mengirim notifikasi, silakan pilih Hapus.
13. Agar alarm dapat melakukan tindakan penskalaan otomatis, EC2, Lambda atau Systems Manager, pilih tombol yang sesuai dan pilih status alarm serta tindakan yang harus dilakukan. Jika Anda memilih sebuah fungsi Lambda sebagai tindakan alarm, maka Anda menentukan nama fungsi atau ARN, dan Anda dapat memilih versi tertentu dari fungsi tersebut secara opsional.

    Alarm dapat melakukan tindakan Systems Manager hanya ketika masuk ke status ALARM. Untuk informasi selengkapnya tentang tindakan Systems Manager, lihat Mengkonfigurasi CloudWatch untuk membuat OpsItems dari alarm dan pembuatan Insiden.

    > **Note**
    >
    > Untuk membuat alarm yang melakukan tindakan SSM Incident Manager, Anda harus memiliki izin tertentu. Untuk informasi selengkapnya, lihat contoh kebijakan berbasis identitas untuk Manajer Insiden AWS Systems Manager.
14. Setelah selesai, silakan pilih Berikutnya.
15. Masukkan sebuah nama dan deskripsi untuk alarm tersebut. Lalu pilih Berikutnya.

    Nama tersebut harus menggunakan karakter UTF-8, dan tidak dapat berisi karakter kontrol ASCII. Deskripsi dapat mencakup pemformatan penurunan harga, yang hanya ditampilkan di tab

Detail alarm di CloudWatch konsol. Penurunan harga dapat Anda gunakan untuk menambahkan tautan ke runbook atau sumber daya internal lainnya.

16. Pada Pratinjau dan buat, konfirmasikan bahwa informasi dan ketentuannya sudah sesuai keinginan Anda, kemudian pilih Buat alarm.

Membuat alarm untuk menghentikan, mengakhiri, menyalakan ulang, atau memulihkan instans EC2

Menggunakan tindakan alarm Amazon CloudWatch, Anda dapat membuat alarm yang secara otomatis menghentikan, mengakhiri, menyalakan ulang, atau memulihkan instans EC2 Anda. Anda dapat menggunakan tindakan menghentikan atau mengakhiri untuk membantu menghemat uang ketika Anda tidak lagi memerlukan suatu instans untuk berjalan. Anda dapat menyalakan ulang dan memulihkan tindakan-tindakan untuk menyalakan ulang instans secara otomatis atau memulihkannya ke perangkat keras yang baru jika terjadi gangguan pada sistem.

Ada sejumlah skenario di mana Anda mungkin ingin menghentikan atau mengakhiri instans Anda secara otomatis. Misalnya, Anda mungkin memiliki instans yang didedikasikan untuk pekerjaan pemrosesan penggajian batch atau tugas komputasi ilmiah yang berjalan selama jangka waktu tertentu dan kemudian menyelesaikan pekerjaannya. Alih-alih membiarkan instans tersebut tidak berfungsi (dan mengeluarkan biaya), Anda dapat menghentikan atau mengakhirinya, sehingga membantu Anda menghemat uang. Perbedaan utama antara penggunaan tindakan menghentikan dan mengakhiri alarm adalah bahwa Anda dapat dengan mudah memulai ulang instans yang terhenti jika Anda perlu mengoperasikannya lagi nanti. Anda juga dapat menyimpan ID instans dan volume akar yang sama. Namun demikian, Anda tidak dapat memulai ulang instans pengakhiran. Sebaliknya, Anda harus meluncurkan instans baru.

Anda dapat menambahkan tindakan penghentian, pengakhiran, penyalaan ulang, atau pemulihan ke alarm apa pun yang diatur pada metrik per instans Amazon EC2, termasuk metrik pemantauan dasar dan terperinci yang disediakan oleh Amazon CloudWatch (dalam ruang nama AWS/EC2), di samping metrik kustom yang menyertakan dimensi "InstanceId=", selama nilai InstanceId mengacu pada instans Amazon EC2 yang beroperasi dengan benar. Anda juga dapat menambahkan tindakan pemulihan ke alarm yang disetel pada metrik per instans Amazon EC2 apa pun kecuali untuk `StatusCheckFailed_Instance`.

Untuk mengatur tindakan alarm CloudWatch yang dapat menyalakan ulang, menghentikan, atau mengakhiri suatu instans, Anda harus menggunakan peran IAM tertaut layanan,

AWSServiceRoleForCloudWatchEvents. Peran IAM AWSServiceRoleForCloudWatchEvents memungkinkan AWS untuk melakukan tindakan alarm atas nama Anda.

Untuk membuat peran tertaut layanan untuk CloudWatch Events, gunakan perintah berikut:

```
aws iam create-service-linked-role --aws-service-name events.amazonaws.com
```

Dukungan konsol

Anda dapat membuat alarm menggunakan konsol CloudWatch atau konsol Amazon EC2. Prosedur dalam dokumentasi ini menggunakan konsol CloudWatch. Untuk prosedur yang menggunakan konsol Amazon EC2, silakan lihat Buat Alarm yang Menghentikan, Mengakhiri, Menyalakan Ulang, atau Memulihkan Instans dalam Panduan Pengguna Amazon EC2 untuk Instans Linux.

Izin

Jika Anda menggunakan akun AWS Identity and Access Management (IAM) untuk membuat atau memodifikasi alarm yang melakukan tindakan EC2 atau tindakan OpsItem Systems Manager, Anda harus memiliki izin `iam:CreateServiceLinkedRole`.

Daftar Isi

Menambahkan tindakan penghentian ke alarm Amazon CloudWatch

Anda dapat membuat alarm yang menghentikan instans Amazon EC2 ketika ambang batas tertentu telah terpenuhi. Misalnya, Anda dapat mengoperasikan pengembangan atau instans pengujian dan terkadang lupa untuk mematikannya. Anda dapat membuat alarm yang dipicu ketika persentase penggunaan CPU rata-rata lebih rendah dari 10 persen selama 24 jam, yang menandakan bahwa alarm tidak berfungsi dan tidak digunakan lagi. Anda dapat menyesuaikan ambang batas, durasi, dan periode agar sesuai dengan kebutuhan, ditambah lagi Anda dapat menambahkan notifikasi SNS, sehingga Anda akan menerima email ketika alarm dipicu.

Instans Amazon EC2 yang menggunakan volume Amazon Elastic Block Store karena perangkat akar dapat dihentikan atau diakhiri, sedangkan instans yang menggunakan penyimpanan instans sebagai perangkat akar hanya dapat diakhiri.

Untuk membuat sebuah alarm yang akan menghentikan instans yang sedang diam dengan menggunakan konsol Amazon CloudWatch

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Alarm, Semua alarm.
3. Pilih Buat alarm.
4. Pilih Pilih Metrik.
5. Untuk ruang nama AWS, pilih EC2.
6. Lakukan hal berikut:
   a. Pilih Metrik Per-Instans.
   b. Pilih kotak centang di baris dengan instans yang benar dan metrik CPUUutilization.
   c. Pilih Metrik bergrafik.
   d. Untuk statistik, silakan pilih Rata-rata.
   e. Pilih sebuah periode (misalnya, **`1 Hour`**).
   f. Pilih Pilih metrik.
7. Untuk langkah Tentukan Alarm, lakukan hal berikut:
   a. Di bawah Kondisi, pilih Statis.
   b. Di bawah Kapan pun CPUUutilization, pilih Lower.
   c. Untuk dari, ketik **`10`**.
   d. Pilih Berikutnya.
   e. Di bawah Notifikasi, untuk Kirim notifikasi ke, pilih topik SNS yang ada atau buat topik yang baru.

      Cara membuat sebuah topik SNS, pilih Daftar baru. Untuk Kirim notifikasi ke, ketik nama untuk topik SNS (misalnya, Stop_EC2_Instance). Untuk Daftar email, ketik daftar alamat email yang dipisahkan dengan tanda koma untuk mendapatkan notifikasi ketika alarm mengubah statusnya menjadi `ALARM`. Masing-masing alamat email akan mendapatkan sebuah email konfirmasi untuk berlangganan topik. Anda harus mengonfirmasi berlangganan tersebut sebelum notifikasi dapat dikirimkan ke sebuah alamat email.

f. Pilih Tambahkan Tindakan EC2.

g. Untuk Pemicu status alarm, pilih Dalam alarm. Untuk Ambil tindakan berikut, pilih Hentikan instans ini.

h. Pilih Berikutnya.

i. Masukkan sebuah nama dan deskripsi untuk alarm tersebut. Nama harus menggunakan karakter ASCII saja. Lalu pilih Berikutnya.

j. Pada Pratinjau dan buat, konfirmasikan bahwa informasi dan ketentuannya sudah sesuai keinginan Anda, kemudian pilih Buat alarm.

Menambahkan tindakan pengakhiran ke alarm Amazon CloudWatch

Anda dapat membuat alarm yang mengakhiri instans EC2 secara otomatis ketika ambang batas tertentu telah terpenuhi (selama proteksi pengakhiran tidak diaktifkan untuk instans tersebut). Misalnya, Anda mungkin ingin mengakhiri sebuah instans ketika telah menyelesaikan pekerjaannya, dan Anda tidak memerlukan instans itu lagi. Jika Anda mungkin ingin menggunakan instans tersebut nanti, Anda harus menghentikan instans tersebut dan tidak mengakhirinya. Untuk informasi tentang mengaktifkan dan menonaktifkan proteksi pengakhiran pada instans tersebut, silakan lihat Mengaktifkan Proteksi Pengakhiran untuk Instans dalam Panduan Pengguna Amazon EC2 untuk Instans Linux.

Untuk membuat alarm guna mengakhiri instans yang sedang menganggur menggunakan konsol Amazon CloudWatch

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Alarm, Buat Alarm.
3. Untuk langkah Pilih Metrik, lakukan hal berikut:

   a. Di bawah Metrik EC2, pilih Metrik Per-Instans.

   b. Pilih baris dengan instans dan metrik CPUUtilization.

   c. Untuk statistik, silakan pilih Rata-rata.

   d. Pilih sebuah periode (misalnya, **1 Hour**).

   e. Pilih Berikutnya.
4. Untuk langkah Tentukan Alarm, lakukan hal berikut:

a. Di bawah Ambang Batas Alarm, ketik nama unik untuk alarm tersebut (misalnya, Akhiri instans EC2) dan penjelasan alarm tersebut (misalnya, Akhiri instans EC2 ketika CPU menganggur terlalu lama). Nama alarm harus memiliki hanya karakter ASCII.

b. Di bawah Kapan pun, untuk adalah, pilih **<** dan ketik **10**. Untuk for, ketik **24** titik-titik berturut-turut.

   Pernyataan grafis dari ambang batas tersebut akan ditampilkan pada Pratinjau Alarm.

c. Di bawah Notifikasi, untuk Kirim notifikasi ke, pilih topik SNS yang ada atau buat topik yang baru.

   Cara membuat sebuah topik SNS, pilih Daftar baru. Untuk Kirim notifikasi ke, ketik nama untuk topik SNS (misalnya, Terminate_EC2_Instance). Untuk Daftar email, ketik daftar alamat email yang dipisahkan dengan tanda koma untuk mendapatkan notifikasi ketika alarm mengubah statusnya menjadi `ALARM`. Masing-masing alamat email akan mendapatkan sebuah email konfirmasi untuk berlangganan topik. Anda harus mengonfirmasi berlangganan tersebut sebelum notifikasi dapat dikirimkan ke sebuah alamat email.

d. Pilih Tindakan EC2.

e. Untuk Kapan pun alarm ini, pilih Status adalah ALARM. Untuk Lakukan tindakan ini, pilih Akhiri instans ini.

f. Pilih Buat Alarm.

Menambahkan tindakan boot ulang ke alarm Amazon CloudWatch

Anda dapat membuat alarm Amazon CloudWatch yang memantau instans Amazon EC2 dan secara otomatis melakukan boot ulang instans. Tindakan alarm boot ulang direkomendasikan untuk kegagalan Pemeriksaan Kondisi instans (sebagai lawan dari tindakan alarm pemulihan, yang sesuai untuk kegagalan Pemeriksaan Kondisi Sistem). Sebuah instans yang melakukan boot ulang setara dengan penyalaan ulang sistem operasi. Dalam kebanyakan kasus, hanya diperlukan beberapa menit untuk menyalakan ulang instans Anda. Saat Anda melakukan boot ulang sebuah instans, ia tetap berada di host fisik yang sama, jadi instans Anda tetap menggunakan nama DNS publik, alamat IP pribadi, dan data apa pun pada volume penyimpanan instansnya.

Melakukan boot ulang instans tidak memulai jam penagihan instans yang baru, tidak seperti menghentikan dan memulai ulang instans Anda. Untuk informasi selengkapnya tentang menyalakan

ulang instans, silakan lihat Menyalakan Ulang Instans Anda dalam Panduan Pengguna Amazon EC2 untuk Instans Linux.

> **Important**
>
> Untuk menghindari kondisi balapan antara tindakan penyalaan ulang dan pemulihan, Anda tidak boleh mengatur periode evaluasi yang sama untuk alarm penyalaan ulang dan alarm pemulihan. Kami menyarankan agar Anda mengatur alarm boot ulang ke tiga periode evaluasi masing-masing selama satu menit.

Untuk membuat alarm guna melakukan boot ulang instans menggunakan konsol Amazon CloudWatch

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Alarm, Buat Alarm.
3. Untuk langkah Pilih Metrik, lakukan hal berikut:
   a. Di bawah Metrik EC2, pilih Metrik Per-Instans.
   b. Pilih baris dengan instans dan metrik StatusCheckFailed_Instance.
   c. Untuk statistik, pilih Minimum.
   d. Pilih sebuah periode (misalnya, **`1 Minute`**) dan pilih Berikutnya.
4. Untuk langkah Tentukan Alarm, lakukan hal berikut:
   a. Di bawah Ambang Batas Alarm, ketik nama unik untuk alarm tersebut (misalnya, instans Boot Ulang EC2) dan penjelasan alarm (misalnya, instans Penyalaan Ulang EC2 ketika pemeriksaan kondisi gagal). Nama alarm harus memiliki hanya karakter ASCII.
   b. Di bawah Kapan pun, untuk adalah, pilih **>** dan ketik **`0`**. Untuk for, ketik **`3`** titik-titik berturut-turut.

      Pernyataan grafis dari ambang batas tersebut akan ditampilkan pada Pratinjau Alarm.
   c. Di bawah Notifikasi, untuk Kirim notifikasi ke, pilih topik SNS yang ada atau buat topik yang baru.

      Cara membuat sebuah topik SNS, pilih Daftar baru. Untuk Kirim notifikasi ke, ketik nama untuk topik SNS (misalnya, Reboot_EC2_Instance). Untuk Daftar email, ketik daftar alamat email yang dipisahkan dengan tanda koma untuk mendapatkan notifikasi

ketika alarm mengubah statusnya menjadi `ALARM`. Masing-masing alamat email akan mendapatkan sebuah email konfirmasi untuk berlangganan topik. Anda harus mengonfirmasi berlangganan tersebut sebelum notifikasi dapat dikirimkan ke sebuah alamat email.

d. Pilih Tindakan EC2.

e. Untuk Kapan pun alarm ini, pilih Status adalah ALARM. Untuk Lakukan tindakan ini, pilih Lakukan boot ulang instans ini.

f. Pilih Buat Alarm.

Menambahkan tindakan pemulihan ke alarm Amazon CloudWatch

Anda dapat membuat alarm Amazon CloudWatch yang memantau instans Amazon EC2 dan secara otomatis memulihkan instans jika rusak akibat kegagalan perangkat keras yang mendasari atau masalah yang memerlukan keterlibatan AWS untuk perbaikan. Instans yang diakhiri tidak dapat dipulihkan. Instans yang dipulihkan identik dengan instans awal, termasuk ID instans , alamat IP pribadi, alamat IP Elastis, dan semua metadata instans.

Ketika alarm `StatusCheckFailed_System` dipicu, dan tindakan pemulihan dimulai, Anda akan diberi tahu oleh topik Amazon SNS yang Anda pilih ketika membuat alarm dan mengaitkan tindakan pemulihan. Selama pemulihan instans, instans tersebut dimigrasikan selama boot ulang instans, dan setiap data yang berada dalam memori hilang. Saat proses selesai, informasi diterbitkan ke topik SNS yang telah Anda konfigurasikan untuk alarm. Siapa pun yang berlangganan topik SNS ini akan menerima notifikasi email yang menyertakan status upaya pemulihan dan instruksi lebih lanjut. Anda akan melihat instans penyalaan ulang pada instans yang dipulihkan.

Tindakan pemulihan hanya dapat digunakan dengan `StatusCheckFailed_System`, tidak dengan `StatusCheckFailed_Instance`.

Contoh masalah yang menyebabkan pemeriksaan status sistem gagal meliputi:

- Kehilangan konektivitas jaringan
- Kehilangan daya sistem
- Masalah perangkat lunak pada host fisik
- Masalah perangkat keras pada host fisik yang memengaruhi jangkauan jaringan

Tindakan pemulihan hanya didukung pada beberapa instans. Untuk informasi selengkapnya tentang jenis instans yang didukung dan persyaratan lainnya, silakan lihat Memulihkan instans dan Persyaratan Anda.

> **Important**
>
> Untuk menghindari kondisi balapan antara tindakan penyalaan ulang dan pemulihan, Anda tidak boleh mengatur periode evaluasi yang sama untuk alarm penyalaan ulang dan alarm pemulihan. Kami menyarankan Anda untuk mengatur alarm pemulihan ke dua periode evaluasi masing-masing satu menit dan boot ulang alarm ke tiga periode evaluasi masing-masing satu menit.

Untuk membuat alarm guna memulihkan instans menggunakan konsol Amazon CloudWatch

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Alarm, Buat Alarm.
3. Untuk langkah Pilih Metrik, lakukan hal berikut:
   a. Di bawah Metrik EC2, pilih Metrik Per-Instans.
   b. Pilih baris dengan instans dan metrik StatusCheckFailed_System.
   c. Untuk statistik, pilih Minimum.
   d. Pilih sebuah periode (misalnya, **`1 Minute`**).

      > **Important**
      >
      > Untuk menghindari kondisi balapan antara tindakan penyalaan ulang dan pemulihan, Anda tidak boleh mengatur periode evaluasi yang sama untuk alarm penyalaan ulang dan alarm pemulihan. Kami menyarankan Anda untuk mengatur alarm pemulihan ke dua periode evaluasi masing-masing selama satu menit.

   e. Pilih Berikutnya.
4. Untuk langkah Tentukan Alarm, lakukan hal berikut:
   a. Di bawah Ambang Batas Alarm, ketik nama unik untuk alarm (misalnya, Pulihkan instans EC2) dan penjelasan alarm (misalnya, Pulihkan instans EC2 jika pemeriksaan kondisi gagal). Nama alarm harus memiliki hanya karakter ASCII.

b. Di bawah Kapan pun, untuk adalah, pilih **>** dan ketik **0**. Untuk for, ketik **2** titik-titik berturut-turut.

c. Di bawah Notifikasi, untuk Kirim notifikasi ke, pilih topik SNS yang ada atau buat topik yang baru.

   Cara membuat sebuah topik SNS, pilih Daftar baru. Untuk Kirim notifikasi ke, ketik nama untuk topik SNS (misalnya, Recover_EC2_Instance). Untuk Daftar email, ketik daftar alamat email yang dipisahkan dengan tanda koma untuk mendapatkan notifikasi ketika alarm mengubah statusnya menjadi `ALARM`. Masing-masing alamat email akan mendapatkan sebuah email konfirmasi untuk berlangganan topik. Anda harus mengonfirmasi berlangganan tersebut sebelum notifikasi dapat dikirimkan ke sebuah alamat email.

d. Pilih Tindakan EC2.

e. Untuk Kapan pun alarm ini, pilih Status adalah ALARM. Untuk Lakukan tindakan ini, pilih Pulihkan instans ini.

f. Pilih Buat Alarm.

Melihat riwayat alarm yang terpicu dan tindakan

Anda dapat melihat riwayat alarm dan tindakan di konsol Amazon CloudWatch. Amazon CloudWatch mempertahankan riwayat alarm dan tindakan selama dua minggu terakhir.

Untuk melihat riwayat alarm dan tindakan yang dipicu

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Alarm dan pilih alarm.
3. Untuk melihat peralihan status terbaru beserta nilai waktu dan metrik, pilih Rincian.
4. Untuk melihat catatan riwayat terbaru, pilih Riwayat.

Sinyal Aplikasi

⚠ Sinyal Aplikasi ada di rilis Pratinjau. Jika Anda memiliki umpan balik tentang fitur ini, Anda dapat menghubungi kami di app-signals-feedback@amazon .com.

Gunakan Sinyal CloudWatch Aplikasi untuk menginstruksikan aplikasi Anda secara otomatis AWS sehingga Anda dapat 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.

- Aktifkan Sinyal Aplikasi untuk secara otomatis mengumpulkan metrik dan jejak dari aplikasi Anda, dan menampilkan metrik kunci seperti volume call, ketersediaan, latensi, gangguan, dan kesalahan. Dengan cepat lihat dan tentukan prioritas kesehatan operasional saat ini dengan cepat, dan apakah aplikasi Anda memenuhi sasaran tujuan jangka panjangnya, tanpa menulis kode ubah suai atau membuat dasbor.
- Buat dan pantau tujuan tingkat layanan (SLO) dengan Sinyal Aplikasi. Buat dan lacak status SLI yang terkait dengan metrik dengan mudah, termasuk CloudWatch metrik aplikasi standar baru yang dikumpulkan oleh Sinyal Aplikasi. Melihat dan melacak status indikator tingkat layanan (SLI) layanan aplikasi Anda dalam sebuah daftar layanan dan peta topologi. Buat alarm untuk melacak SLO Anda, dan lacak metrik aplikasi standar baru yang dikumpulkan Sinyal Aplikasi tersebut.
- Meihat peta topologi aplikasi Anda yang secara otomatis ditemukan oleh Sinyal Aplikasi, yang akan memberi Anda representasi visual dari aplikasi, dependensi, dan konektivitasnya.
- Application Signals bekerja dengan CloudWatch Real user monitoring (RUM), kenari CloudWatch Synthetics, dan AWS Service Catalog AppRegistryuntuk menampilkan halaman klien Anda, kenari Synthetics, dan nama aplikasi dalam dasbor dan peta.

Gunakan Sinyal Aplikasi untuk pemantauan aplikasi harian

Gunakan Sinyal Aplikasi dalam CloudWatch konsol, sebagai bagian dari pemantauan aplikasi harian:

1. Jika Anda telah membuat tujuan tingkat layanan (SLO) untuk layanan Anda, maka mulailah dengan halaman Tujuan Tingkat Layanan (SLO). Layanan ini memberi Anda tampilan langsung kesehatan layanan dan operasi Anda yang paling penting. Pilih nama layanan atau operasi untuk

SLO untuk membuka halaman Detail Layanan dan melihat informasi layanan yang detail saat Anda menerbitkan pemecahan masalah.

2. Buka halaman Layanan untuk melihat ringkasan semua layanan Anda, dan dengan cepat melihat layanan dengan kecepatan gangguan atau latensi tertinggi. Jika Anda telah membuat SLO, silakan lihat tabel Layanan untuk melihat layanan manakah yang memiliki indikator tingkat layanan (SLI) yang tidak sehat. Jika layanan tertentu dalam keadaan tidak sehat, pilih layanan untuk membuka halaman Detail Layanan dan lihat operasi layanan, dependensi, canary Synthetics, dan permintaan klien. Pilih titik dalam grafik untuk melihat jejak yang berkorelasi sehingga Anda dapat memecahkan masalah dan mengidentifikasi akar penyebab masalah-masalah operasional.
3. Jika layanan baru telah disebarkan atau dependensi telah berubah, buka Peta Layanan untuk menginspeksi topologi aplikasi Anda. Lihat peta aplikasi Anda yang menunjukkan hubungan antara klien, canary Synthetics, layanan, dan dependensi. Lihat kesehatan SLI dengan cepat, silakan lihat metrik kunci seperti volume call, kecepatan gangguan, dan latensi, dan telusuri untuk melihat informasi lebih detail di halaman Detail layanan.

Penggunaan Sinyal Aplikasi menimbulkan pengeluaran biaya. Untuk informasi tentang CloudWatch harga, lihat CloudWatch Harga Amazon.

> **Note**
>
> Tidak perlu mengaktifkan Sinyal Aplikasi untuk menggunakan CloudWatch Synthetics, CloudWatch RUM, atau CloudWatch Evidently. Namun, Synthetics dan CloudWatch RUM bekerja dengan Application Signals untuk memberikan manfaat ketika Anda menggunakan fitur ini bersama-sama.

Bahasa dan arsitektur yang didukung

Saat ini, Sinyal Aplikasi mendukung aplikasi Java. Dukungan aplikasi berupa bahasa lain direncanakan untuk waktu yang akan datang. Aplikasi Python mendukung akses awal tersedia. Untuk petunjuk penyiapan, hubungi kami di app-signals-feedback@amazon .com.

Sinyal Aplikasi didukung dan diuji di Amazon EKS, Amazon ECS, dan Amazon EC2. Di klaster Amazon EKS, secara otomatis menemukan nama layanan dan cluster Anda. Di arsitektur lain, Anda harus memberikan nama layanan dan lingkungan ketika Anda mengaktifkan layanan tersebut untuk Sinyal Aplikasi.

Petunjuk untuk mengaktifkan Sinyal Aplikasi di Amazon EC2 harus bekerja pada arsitektur apa pun yang mendukung agen AWS dan CloudWatch Distro. OpenTelemetry Namun demikian, instruksinya belum diuji pada arsitektur selain Amazon ECS dan Amazon EC2.

Wilayah yang didukung

Untuk rilis Pratinjau ini, Sinyal Aplikasi didukung di Wilayah-Wilayah berikut.

- AS Timur (Virginia Utara)
- AS Timur (Ohio)
- AS Barat (Oregon)
- Asia Pasifik (Sydney)
- Asia Pasifik (Tokyo)
- Eropa (Irlandia)

Pratinjau SDK

Versi Pratinjau SDK sudah tersedia untuk diunduh.

> **Warning**
>
> Operasi dan parameter API dapat berubah sebelum Sinyal Aplikasi tersedia secara umum. Perubahan-perubahan tersebut mungkin melanggar perubahan. Jangan gunakan versi Pratinjau SDK untuk tujuan produksi.

Untuk menginstal SDK Pratinjau, instal atau perbarui versi terbaru AWS CLI versi 2. Untuk informasi selengkapnya, silakan lihat Menginstal atau memperbarui versi terbaru dari AWS CLI.

Kemudian gunakan perintah berikut untuk mengunduh file zip SDK dari bucket Amazon S3, kemudian ekstrak isinya. Setiap file zip SDK berisi instruksi-instruksi SDK dan dokumentasi API.

> **Note**
>
> SDK disediakan dalam beberapa bahasa pemrograman sehingga Anda dapat menggunakan Application Signals API dengan salah satu bahasa pemrograman ini. Namun, instrumentasi otomatis aplikasi Anda untuk mengirim data ke Sinyal Aplikasi hanya didukung untuk aplikasi Java.

- SDK Java V2: `aws s3 cp s3://application-signals-preview-sdk/awsJavaSdkV2.zip ./`
- JavaScript SDK V3: `aws s3 cp s3://application-signals-preview-sdk/jsSdkV3.zip ./`
- JavaScript V2 SDK: `aws s3 cp s3://application-signals-preview-sdk/jsSdkV2.zip ./`
- SDK Python: `aws s3 cp s3://application-signals-preview-sdk/pythonSdk.zip ./`
- SDK Kotlin: `aws s3 cp s3://application-signals-preview-sdk/kotlin.zip ./`
- SDK Android: `aws s3 cp s3://application-signals-preview-sdk/andriod.zip ./`
- C++ SDK: `aws s3 cp s3://application-signals-preview-sdk/awsCppSdk.zip ./`
- SDK PHP: `aws s3 cp s3://application-signals-preview-sdk/awsSdkPhp.zip ./`
- SDK Ruby `aws s3 cp s3://application-signals-preview-sdk/awsSdkRuby.zip ./`
- SDK Go V2: `aws s3 cp s3://application-signals-preview-sdk/awsSdkGoV2.zip ./`
- SDK Go V1: `aws s3 cp s3://application-signals-preview-sdk/go.zip ./`
- SDK iOS: `aws s3 cp s3://application-signals-preview-sdk/iOS.zip ./`

Topik

Izin diperlukan untuk Sinyal Aplikasi

⚠ Sinyal Aplikasi berada dalam rilis pratinjau untuk Amazon CloudWatch dan dapat berubah sewaktu-waktu.

Bagian ini menjelaskan izin yang diperlukan bagi Anda untuk mengaktifkan, mengelola, dan mengoperasikan Sinyal Aplikasi.

Izin untuk mengaktifkan dan mengelola Sinyal Aplikasi

Untuk mengelola Sinyal Aplikasi, dan untuk mengaktifkan Sinyal Aplikasi dengan pengaturan khusus pada arsitektur selain Amazon EKS, Anda harus masuk dengan izin berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CloudWatchApplicationSignalsFullAccessPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:BatchGetServiceLevelIndicatorReport",
                "cloudwatch:BatchGetServiceLevelObjectiveBudgetReport",
                "cloudwatch:CreateServiceLevelObjective",
                "cloudwatch:DeleteServiceLevelObjective",
                "cloudwatch:EnableTopologyDiscovery",
                "cloudwatch:GetService",
                "cloudwatch:GetServiceLevelObjective",
                "cloudwatch:GetTopologyMap",
                "cloudwatch:ListServices",
                "cloudwatch:ListServiceLevelObjectives",
                "cloudwatch:UpdateServiceLevelObjective",
                "iam:GetRole"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsAlarmsPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:DescribeAlarms"
```

```
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsMetricsPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsSyntheticsPermissions",
            "Effect": "Allow",
            "Action": [
                "synthetics:DescribeCanariesLastRun",
                "synthetics:GetCanaryRuns"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsRumPermissions",
            "Effect": "Allow",
            "Action": [
                "rum:BatchCreateRumMetricDefinitions",
                "rum:BatchDeleteRumMetricDefinitions",
                "rum:BatchGetRumMetricDefinitions",
                "rum:GetAppMonitor",
                "rum:GetAppMonitorData",
                "rum:ListAppMonitors",
                "rum:PutRumMetricsDestination",
                "rum:UpdateRumMetricDefinition"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsXrayPermissions",
            "Effect": "Allow",
            "Action": [
                "xray:GetTraceSummaries"
            ],
            "Resource": "*"
        },
        {
```

```
            "Sid": "CloudWatchApplicationSignalsPutMetricAlarmPermissions",
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricAlarm",
            "Resource": [
                "arn:aws:cloudwatch:*:*:alarm:SLO-AttainmentGoalAlarm-*",
                "arn:aws:cloudwatch:*:*:alarm:SLO-WarningAlarm-*",
                "arn:aws:cloudwatch:*:*:alarm:SLI-HealthAlarm-*"
            ]
        },
        {
            "Sid": "CloudWatchApplicationSignalsEksPermissions",
            "Effect": "Allow",
            "Action": [
                "eks:ListAddons",
                "eks:ListClusters"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsEksDescribeAddonPermissions",
            "Effect": "Allow",
            "Action": [
                "eks:DescribeAddon"
            ],
            "Resource": "arn:aws:eks:*:*:addon/*/amazon-cloudwatch-observability/*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsCreateServiceLinkedRolePermissions",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/application-
signals.cloudwatch.amazonaws.com/AWSServiceRoleForCloudWatchApplicationSignals",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "application-
signals.cloudwatch.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CloudWatchApplicationSignalsTaggingPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:TagResource",
```

```
                "cloudwatch:UntagResource",
                "cloudwatch:ListTagsForResource"
            ],
            "Resource": "arn:aws:cloudwatch:*:*:slo/*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsSnsWritePermissions",
            "Effect": "Allow",
            "Action": [
                "sns:CreateTopic",
                "sns:Subscribe"
            ],
            "Resource": "arn:aws:sns:*:*:cloudwatch-application-signals-*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsSnsReadPermissions",
            "Effect": "Allow",
            "Action": "sns:ListTopics",
            "Resource": "*"
        }
    ]
}
```

Untuk menggunakan konsol untuk mengaktifkan Sinyal Aplikasi pada aplikasi di klaster Amazon EKS, Anda juga memerlukan izin berikut. Izin ini diperlukan untuk melakukan instalasi dan mengelola add-on EKS Observabilitas Amazon CloudWatch.

> **Important**
>
> Izin ini termasuk `iam:PassRole` dengan `Resource "*"` dan `eks:CreateAddon` dengan`Resource “*”`. Ini adalah izin yang kuat dan Anda harus berhati-hati dalam memberikannya.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CloudWatchApplicationSignalsEksAddonManagementPermissions",
            "Effect": "Allow",
            "Action": [
                "eks:CreateAddon",
```

```
                "eks:DescribeAddon",
                "eks:DescribeAddonConfiguration",
                "eks:DescribeAddonVersions",
                "eks:DescribeCluster",
                "eks:DescribeUpdate",
                "eks:ListAddons",
                "eks:ListClusters",
                "eks:ListUpdates",
                "iam:ListRoles",
                "iam:PassRole"
            ],
            "Resource": "*"
        },
        {
            "Sid":
 "CloudWatchApplicationSignalsEksCloudWatchObservabilityAddonManagementPermissions",
            "Effect": "Allow",
            "Action": [
                "eks:DeleteAddon",
                "eks:UpdateAddon"
            ],
            "Resource": "arn:aws:eks:*:*:addon/*/amazon-cloudwatch-observability/*"
        }
    ]
}
```

Untuk melihat Aplikasi AppRegistry AWS Service Catalog mana yang terkait dengan SLO Anda di halaman SLO di dasbor Sinyal Aplikasi, Anda juga memerlukan izin berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CloudWatchApplicationSignalsTaggingReadPermissions",
            "Effect": "Allow",
            "Action": "tag:GetResources",
            "Resource": "*"
        }
    ]
}
```

Sinyal Aplikasi Operasi

Operator layanan yang menggunakan Sinyal Aplikasi untuk memantau layanan dan SLO harus masuk ke akun dengan izin berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CloudWatchApplicationSignalsReadOnlyAccessPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:BatchGetServiceLevelIndicatorReport",
                "cloudwatch:BatchGetServiceLevelObjectiveBudgetReport",
                "cloudwatch:GetService",
                "cloudwatch:GetServiceLevelObjective",
                "cloudwatch:GetTopologyMap",
                "cloudwatch:ListServices",
                "cloudwatch:ListServiceLevelObjectives"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsAlarmsReadPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:DescribeAlarms"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsMetricsReadPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsSyntheticsReadPermissions",
            "Effect": "Allow",
            "Action": [
                "synthetics:DescribeCanariesLastRun",
```

```
                "synthetics:GetCanaryRuns"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsRumReadPermissions",
            "Effect": "Allow",
            "Action": [
                "rum:BatchGetRumMetricDefinitions",
                "rum:GetAppMonitor",
                "rum:GetAppMonitorData",
                "rum:ListAppMonitors"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsXrayReadPermissions",
            "Effect": "Allow",
            "Action": [
                "xray:GetTraceSummaries"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsTaggingReadPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:ListTagsForResource"
            ],
            "Resource": "arn:aws:cloudwatch:*:*:slo/*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsEksReadPermissions",
            "Effect": "Allow",
            "Action": [
                "eks:ListAddons",
                "eks:ListClusters"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsEksDescribeAddonReadPermissions",
            "Effect": "Allow",
            "Action": [
```

```
                "eks:DescribeAddon"
            ],
            "Resource": "arn:aws:eks:*:*:addon/*/amazon-cloudwatch-observability/*"
        }
    ]
}
```

Agar operator dapat melihat Aplikasi AppRegistry AWS Service Catalog mana yang terkait dengan SLO Anda di halaman SLO di dasbor Sinyal Aplikasi, mereka juga memerlukan izin berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CloudWatchApplicationSignalsTaggingReadPermissions",
            "Effect": "Allow",
            "Action": "tag:GetResources",
            "Resource": "*"
        }
    ]
}
```

Mengaktifkan Sinyal Aplikasi

⚠ Sinyal Aplikasi ada di rilis Pratinjau. Jika Anda memiliki umpan balik tentang fitur ini, Anda dapat menghubungi kami di app-signals-feedback@amazon .com.

Topik di bagian ini menjelaskan cara mengaktifkan Sinyal CloudWatch Aplikasi di lingkungan Anda. Sinyal Aplikasi didukung pada klaster Amazon EKS dengan alur kerja pengaturan menggunakan konsol. Ini juga didukung pada platform lain, termasuk Amazon EC2, dengan proses pengaturan khusus.

Topik

- Melakukan pemecahan masalah instalasi yang terjadi pada Sinyal Aplikasi Anda
- Mengonfigurasi Sinyal Aplikasi

Sistem yang didukung oleh Sinyal Aplikasi

⚠ Sinyal Aplikasi ada di rilis Pratinjau. Jika Anda memiliki umpan balik tentang fitur ini, Anda dapat menghubungi kami di app-signals-feedback@amazon .com.

Sinyal Aplikasi didukung dan diuji di Amazon EKS, Amazon ECS, dan Amazon EC2. Petunjuk untuk mengaktifkan Sinyal Aplikasi di Amazon EC2 harus berfungsi pada platform apa pun yang mendukung CloudWatch agen AWS dan Distro OpenTelemetry, tetapi instruksinya belum diuji pada platform lain.

Kompatibilitas Java

Application Signals mendukung aplikasi Java, dan mendukung pustaka dan kerangka kerja Java yang sama seperti yang dilakukan AWS Distro. OpenTelemetry Untuk informasi selengkapnya, silakan lihat Pustaka, kerangka kerja, server aplikasi, dan JVM yang didukung.

JVM versi 8, 11, dan 17 didukung.

Kompatibilitas Python (Beta)

Aplikasi Python mendukung akses awal tersedia. Untuk petunjuk penyiapan, hubungi kami di app-signals-feedback@amazon .com.

OpenTelemetry pertimbangan kompatibilitas

⚠ Sinyal Aplikasi ada di rilis Pratinjau. Jika Anda memiliki umpan balik tentang fitur ini, Anda dapat menghubungi kami di app-signals-feedback@amazon .com.

Untuk onboard aplikasi Anda dengan CloudWatch Application Signals, kami sarankan Anda menghapus sepenuhnya solusi pemantauan kinerja aplikasi yang ada dari aplikasi Anda sebelumnya. Ini termasuk menghapus kode instrumentasi dan konfigurasi apa pun.

Meskipun Sinyal Aplikasi menggunakan OpenTelemetry instrumentasi, itu tidak dijamin kompatibel dengan OpenTelemetry instrumentasi atau konfigurasi Anda yang ada. Dalam skenario kasus terbaik, Anda mungkin dapat menyimpan beberapa OpenTelemetry fungsionalitas Anda, seperti metrik khusus. Namun demikian, pastikan untuk membaca bagian berikut untuk detailnya.

Pertimbangan jika Anda sudah menggunakan OpenTelemetry

Jika Anda sudah menggunakan OpenTelemetry dengan aplikasi Java Anda, sisa bagian ini berisi informasi penting untuk mencapai kompatibilitas dengan Sinyal Aplikasi.

- Sebelum Anda mengaktifkan aplikasi Anda untuk Sinyal Aplikasi, Anda harus menghapus injeksi agen OpenTelemetry Java dari aplikasi Anda.
- Jika Anda menggunakan instrumentasi manual untuk menghasilkan rentang atau metrik kustom dari aplikasi Anda, maka tergantung pada kompleksitas instrumentasi yang digunakan, mengaktifkan Sinyal Aplikasi akan dapat menyebabkan mereka berhenti menghasilkan data atau memiliki perilaku berbeda yang tidak diinginkan. Anda mungkin dapat menggunakan beberapa konfigurasi yang tersedia di OpenTelemetry (kecuali yang disebutkan dalam tabel nanti di bagian ini) untuk mempertahankan perilaku yang diinginkan dari metrik atau rentang yang ada. Untuk informasi selengkapnya tentang konfigurasi ini, lihat Konfigurasi Otomatis OpenTelemetry SDK aktif. GitHub

  Misalnya, dengan menggunakan `OTEL_EXPORTER_OTLP_METRICS_ENDPOINT` konfigurasi dan instance OpenTelemetry Kolektor yang dikelola sendiri, Anda mungkin dapat terus mengirim metrik kustom ke tujuan yang Anda inginkan.
- Beberapa variabel lingkungan atau properti sistem tidak boleh digunakan dengan Sinyal Aplikasi, sementara Anda dapat menggunakan variabel lingkungan atau properti sistem yang lain selama Anda mengikuti panduan dalam tabel. Lihat tabel berikut untuk detailnya.

Variabel Lingkungan	Rekomendasi dengan Sinyal Aplikasi
`OTEL_SDK_DISABLED`	Tidak boleh diatur ke `true`.
`OTEL_TRACES_EXPORTER`	Harus diatur ke `otlp`.
`OTEL_EXPORTER_OTLP_ENDPOINT`	Tidak boleh digunakan.
`OTEL_EXPORTER_OTLP_TRACES_ENDPOINT`	Tidak boleh digunakan.

Variabel Lingkungan	Rekomendasi dengan Sinyal Aplikasi
`OTEL_JAVA_ENABLED_RESOURCE_PROVIDERS`	Jika diatur, harus menyertakan detektor AWS sumber daya.
`OTEL_ATTRIBUTE_COUNT_LIMIT`	Jika diatur, harus diatur cukup tinggi untuk menyertakan sekitar 10 atribut rentang lebih yang ditambahkan oleh Sinyal CloudWatch Aplikasi.
`OTEL_PROPAGATORS`	Jika diatur, maka ia harus menyertakan `xray` untuk penelusuran akhir.
`OTEL_TRACES_SAMPLER`	Jika diatur, harus `xray` untuk menggunakan sampling terpusat X-Ray. Untuk menggunakan sampling lokal, atur ini ke `parentbased_traceidratio` dan tentukan laju sampling di `OTEL_TRACES_SAMPLER_ARG` .
`OTEL_TRACES_SAMPLER_ARG`	Jika Anda menggunakan default sampel pelacakan terpusat X-Ray, variabel ini tidak boleh digunakan. Jika Anda menggunakan sampling lokal sebagai gantinya, atur laju sampling dalam variabel ini. Misalnya, `0.05` untuk laju sampling 5%.

Mengaktifkan Sinyal Aplikasi di klaster Amazon EKS

⚠ Sinyal Aplikasi ada di rilis Pratinjau. Jika Anda memiliki umpan balik tentang fitur ini, Anda dapat menghubungi kami di app-signals-feedback@amazon .com.

CloudWatch Sinyal Aplikasi didukung untuk aplikasi Java yang berjalan di kluster Amazon EKS. Untuk mengaktifkan Sinyal Aplikasi untuk aplikasi di klaster Amazon EKS, Anda memiliki dua opsi:

- Untuk mengaktifkan Sinyal Aplikasi untuk aplikasi Anda di klaster Amazon EKS yang ada, gunakan langkah-langkah yang dijelaskan di Aktifkan Sinyal Aplikasi di klaster Amazon EKS dengan layanan Anda.
- Untuk mencoba Sinyal Aplikasi di lingkungan non-produksi dengan aplikasi sampel, gunakan instruksi yang dijelaskan di Aktifkan Sinyal Aplikasi di klaster Amazon EKS baru dengan aplikasi sampel. Alur kerja ini menggunakan skrip yang disediakan oleh AWS untuk membuat klaster Amazon EKS baru dan menginstal aplikasi sampel yang diaktifkan untuk Sinyal Aplikasi. Ini memungkinkan Anda melihat dan menguji end-to-end fungsionalitas Sinyal Aplikasi.

Topik

Aktifkan Sinyal Aplikasi di klaster Amazon EKS dengan layanan Anda

⚠ Sinyal Aplikasi ada di rilis Pratinjau. Jika Anda memiliki umpan balik tentang fitur ini, Anda dapat menghubungi kami di app-signals-feedback@amazon .com.

Untuk mengaktifkan Sinyal CloudWatch Aplikasi pada aplikasi Java Anda pada kluster Amazon EKS yang ada, gunakan instruksi di bagian ini.

⚠ **Important**

Jika Anda sudah menggunakan aplikasi OpenTelemetry yang ingin Anda aktifkan untuk Sinyal Aplikasi, lihat OpenTelemetry pertimbangan kompatibilitas sebelum Anda mengaktifkan Sinyal Aplikasi.

Aplikasi Python mendukung akses awal tersedia. Untuk petunjuk penyiapan, hubungi kami di app-signals-feedback@amazon .com.

Untuk mengaktifkan Sinyal Aplikasi untuk aplikasi Anda di klaster Amazon EKS yang ada

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Layanan.
3. Jika Anda belum mengaktifkan Sinyal Aplikasi di akun ini, Anda harus memberikan izin yang diperlukan Sinyal Aplikasi untuk menemukan layanan Anda. Untuk melakukan hal itu, lakukan hal berikut. Anda perlu melakukan ini sekali untuk akun Anda.

   a. Pilih Mulai menemukan Layanan Anda.

   b. Pilih kotak centang kemudian pilih Mulai menemukan Layanan.

      Menyelesaikan langkah ini untuk pertama kalinya di akun Anda akan membuat peran AWSServiceRoleForCloudWatchApplicationSignalsterkait layanan. Peran ini akan memberi Sinyal Aplikasi izin-izin berikut:

      - `xray:GetServiceGraph`
      - `logs:StartQuery`
      - `logs:GetQueryResults`
      - `cloudwatch:GetMetricData`
      - `cloudwatch:ListMetrics`
      - `tag:GetResources`

      Untuk informasi selengkapnya tentang peran ini, silakan lihat Izin peran terkait layanan untuk Sinyal Aplikasi CloudWatch .
4. Pilih Aktifkan Sinyal Aplikasi.
5. Untuk Tentukan platform, pilih EKS.
6. Untuk Pilih klaster EKS, pilih klaster tempat Anda ingin mengaktifkan Sinyal Aplikasi.
7. Jika klaster ini belum mengaktifkan add-on Amazon CloudWatch Observability EKS, Anda akan diminta untuk mengaktifkannya. Jika ini terjadi, lakukan hal berikut:

   a. Pilih Add CloudWatch Observability EKS add-on. Konsol Amazon EKS akan dimunculkan.

   b. Pilih kotak centang untuk Amazon CloudWatch Observability dan pilih Berikutnya.

Add-on CloudWatch Observability EKS memungkinkan Sinyal Aplikasi dan CloudWatch Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS. Untuk informasi selengkapnya tentang Wawasan Kontainer ini, silakan lihat Wawasan Kontainer.

c. Pilih versi terbaru add-on untuk diinstal.

d. Pilih peran IAM yang akan digunakan untuk add-on. Jika Anda memilih Diwariskan dari node, lampirkan izin yang benar untuk peran IAM yang digunakan oleh simpul pekerja Anda. Ganti *my-worker-node-role*dengan peran IAM yang digunakan oleh node pekerja Kubernetes Anda.

```
aws iam attach-role-policy \
--role-name my-worker-node-role \
--policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy \
--policy-arn arn:aws:iam::aws:policy/AWSXRayWriteOnlyAccess
```

e. Jika Anda ingin membuat peran layanan untuk menggunakan add-on, silakan lihat Instal CloudWatch agen dengan menggunakan add-on Amazon CloudWatch Observability EKS.

f. Pilih Berikutnya, konfirmasikan informasi di layar, dan pilih Buat.

g. Di layar berikutnya, pilih Aktifkan Sinyal CloudWatch Aplikasi untuk kembali ke CloudWatch konsol dan menyelesaikan prosesnya.

8. Di CloudWatch konsol, bagian Configure application metrics dan trace sampling menjelaskan bahwa Anda harus menambahkan anotasi ke manifes YAMAL di cluster. Dengan menambahkan keterangan ini maka secara otomatis melengkapi aplikasi untuk mengirim metrik, jejak, dan log ke Sinyal Aplikasi.

   Anda memiliki dua pilihan untuk keterangan:

   - Beri Keterangan untuk Beban Kerja secara otomatis melengkapi sebuah beban kerja tunggal di klaster.
   - Beri Keterangan Namespace secara otomatis melengkapi semua beban kerja yang di-deploy di namespace yang dipilih.

   Pilih salah satu pilihan tersebut, dan ikuti langkah-langkah yang sesuai:

   - a. Pilih Beri Keterangan Beban Kerja.
     b. Tempelkan baris `annotations: instrumentation.opentelemetry.io/inject-java: "true"` ke bagian `PodTemplate` manifes beban kerja.

c. Di terminal Anda, masukkan `kubectl apply -f your_deployment_yaml` untuk menerapkan perubahan.

- a. Pilih Berikan Keterangan Namespace.
  b. Tempelkan baris `annotations: instrumentation.opentelemetry.io/inject-java: "true"` ke dalam bagian metadata manifes namespace.
  c. Di terminal Anda, masukkan `kubectl apply -f your_namespace_yaml` untuk menerapkan perubahan.
  d. Di terminal Anda, masukkan satu perintah untuk memulai ulang semua pod di namespace tersebut. Contoh perintah untuk me-restart deployment beban kerja adalah `kubectl rollout restart deployment -n namespace_name`

9. Pilih Tampilkan Layanan setelah selesai. Hal ini akan membawa Anda ke tampilan Layanan Sinyal Aplikasi, di mana Anda dapat melihat data yang dikumpulkan oleh Sinyal Aplikasi. Mungkin diperlukan waktu beberapa menit hingga akhirnya data muncul.

   Untuk mengaktifkan Sinyal Aplikasi di klaster Amazon EKS lainnya, pilih Aktifkan Sinyal Aplikasi dari layar Layanan.

   Untuk informasi selengkapnya tentang tampilan Layanan, silakan lihat Memantau kondisi kesehatan operasional aplikasi Anda dengan Sinyal Aplikasi.

Aktifkan Sinyal Aplikasi di klaster Amazon EKS baru dengan aplikasi sampel

⚠ Sinyal Aplikasi ada di rilis Pratinjau. Jika Anda memiliki umpan balik tentang fitur ini, Anda dapat menghubungi kami di app-signals-feedback@amazon .com.

Untuk mencoba Sinyal CloudWatch Aplikasi pada aplikasi sampel sebelum Anda menginstruksikan aplikasi Java Anda sendiri dengannya, ikuti petunjuk di bagian ini. Instruksi ini menggunakan skrip untuk membantu Anda membuat klaster Amazon EKS, menginstal aplikasi sampel, dan melengkapi aplikasi sampel untuk bekerja dengan Sinyal Aplikasi.

Aplikasi sampel adalah sebuah aplikasi Spring “Pet Clinic” yang terdiri dari empat layanan mikro. Layanan ini berjalan di Amazon EKS di Amazon EC2 dan memanfaatkan skrip pemberdayaan Sinyal Aplikasi untuk mengaktifkan klaster dengan agen instrumentasi otomatis Java.

Persyaratan

- Saat ini, Sinyal Aplikasi hanya memantau aplikasi Java.
- Anda harus AWS CLI menginstal pada instance. Kami merekomendasikan AWS CLI versi 2, tetapi versi 1 juga harus berfungsi. Untuk informasi selengkapnya tentang menginstal AWS CLI, lihat Menginstal atau memperbarui versi terbaru AWS CLI.
- Skrip-skrip yang ada di bagian ini dimaksudkan untuk dijalankan di lingkungan Linux dan macOS. Untuk instance Windows, kami menyarankan Anda menggunakan AWS Cloud9 lingkungan untuk menjalankan skrip ini. Untuk informasi lebih lanjut tentang AWS Cloud9, lihat Apa itu AWS Cloud9? .
- Instal versi yang didukung dari `kubectl`. Anda harus menggunakan versi dari `kubectl` dalam satu perbedaan versi kecil dari bidang kontrol klaster Amazon EKS. Sebagai contoh, klien `kubectl` 1,26 bekerja dengan klaster Kubernetes 1,25, 1,26, dan 1,27. Jika Anda sudah memiliki kluster Amazon EKS, Anda mungkin perlu mengonfigurasi AWS kredensialnya. `kubectl` Untuk informasi selengkapnya, silakan lihat Membuat atau memperbarui file `kubeconfig` untuk klaster Amazon EKS.
- Instal`eksctl`. `eksctl`menggunakan AWS CLI untuk berinteraksi dengan AWS, yang berarti ia menggunakan AWS kredensil yang sama dengan. AWS CLI Untuk informasi selengkapnya, silakan lihat Menginstal atau memperbarui `eksctl`.
- Instal `jq`. `jq` diperlukan untuk menjalankan skrip pemberdayaan Sinyal Aplikasi. Untuk informasi selengkapnya, silakan lihat Unduh jq.

Langkah 1: Mengunduh skrip

Untuk mengunduh skrip untuk mengatur Sinyal CloudWatch Aplikasi dengan aplikasi sampel, Anda dapat mengunduh dan membuka kompres file GitHub proyek zip ke drive lokal, atau Anda dapat mengkloning proyek. GitHub

Untuk mengkloning proyek, buka sebuah jendela terminal dan masukkan perintah Git berikut di direktori kerja yang diberikan.

```
git clone https://github.com/aws-observability/application-signals-demo.git
```

Langkah 2: Membangun dan menerapkan aplikasi sampel

Untuk membuat dan mendorong gambar aplikasi sampel, ikuti petunjuk ini.

Langkah 3: Mendeploy dan mengaktifkan Sinyal Aplikasi dan aplikasi sampel

Pastikan Anda telah melengkapi persyaratan yang tercantum di Aktifkan Sinyal Aplikasi di klaster Amazon EKS baru dengan aplikasi sampel sebelum Anda menyelesaikan langkah-langkah berikut.

Untuk mendeploy dan mengaktifkan Sinyal Aplikasi dan aplikasi sampel

1. Masukkan perintah berikut di terminal lokal tempat Anda membuka paket skrip orientasi. Ganti *new-cluster-name*dengan nama yang ingin Anda gunakan untuk cluster baru. *Ganti nama wilayah* dengan nama AWS Wilayah, seperti. `us-west-1`

   Perintah ini mengatur aplikasi sampel yang berjalan di klaster Amazon EKS baru dengan Sinyal Aplikasi yang diaktifkan.

   ```
   # assuming the current working directory is 'onboarding'
   # this script sets up a new cluster, enables Application Signals, and deploys the
   # sample application
   cd application-signals-demo/scripts/eks/appsignals/one-step && ./setup.sh new-
   cluster-name region-name
   ```

   Skrip pengaturan membutuhkan waktu sekitar 30 menit untuk dijalankan, dan melakukan hal berikut:

   - Untuk membuat sebuah klaster Amazon EKS yang baru di Wilayah yang ditentukan.
   - Membuat izin IAM yang diperlukan untuk Sinyal Aplikasi (`arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess` dan `arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy`).
   - Mengaktifkan Sinyal Aplikasi dengan menginstal CloudWatch agen dan menginstrumentasi aplikasi sampel secara otomatis untuk CloudWatch metrik dan jejak X-Ray.
   - Menyebarkan aplikasi sampel PetClinic Spring di cluster Amazon EKS yang sama.
   - Menciptakan lima kenari CloudWatch Synthetics, bernama,,`pc-add-vist`,`pc-create-owners`,`pc-visit-pet`. `pc-visit-vet` `pc-clinic-traffic` Canary ini akan berjalan pada frekuensi satu menit untuk menghasilkan lalu lintas sintetis untuk aplikasi sampel dan menunjukkan bagaimana canary Synthetics muncul di Sinyal Aplikasi.
   - Menciptakan empat tujuan tingkat layanan (SLOs) untuk PetClinic aplikasi dengan nama-nama berikut:
     - Ketersediaan untuk Mencari Pemilik
     - Latensi untuk Mencari Pemilik

    - Ketersediaan untuk Mendaftarkan Pemilik
    - Latensi untuk Mendaftarkan Pemilik
  - Membuat peran IAM yang diperlukan dengan sebuah kebijakan kepercayaan kustom yang memberikan Sinyal Aplikasi izin-izin berikut ini:
    - `cloudwatch:PutMetricData`
    - `cloudwatch:GetMetricData`
    - `xray:GetServiceGraph`
    - `logs:StartQuery`
    - `logs:GetQueryResults`
2. (Opsional) Jika Anda ingin meninjau kode sumber untuk aplikasi PetClinic sampel, Anda dapat menemukannya di bawah folder root.

```
- application-signals-demo
  - spring-petclinic-admin-server
  - spring-petclinic-api-gateway
  - spring-petclinic-config-server
  - spring-petclinic-customers-service
  - spring-petclinic-discovery-server
  - spring-petclinic-vets-service
  - spring-petclinic-visits-service
```

3. Untuk melihat aplikasi PetClinic sampel yang digunakan, jalankan perintah berikut untuk menemukan URL:

```
kubectl get ingress
```

Langkah 4: Memantau aplikasi sampel

Setelah menyelesaikan langkah-langkah di bagian sebelumnya untuk membuat klaster Amazon EKS dan mendeploy aplikasi sampel, Anda dapat menggunakan Sinyal Aplikasi untuk memantau aplikasi.

> **Note**
>
> Agar konsol Sinyal Aplikasi mulai mengisi, beberapa lalu lintas harus mencapai aplikasi sampel tersebut. Bagian dari langkah sebelumnya dibuat kenari CloudWatch Synthetics yang menghasilkan lalu lintas ke aplikasi sampel.

Pemantauan kondisi layanan

Setelah diaktifkan, Sinyal CloudWatch Aplikasi secara otomatis menemukan dan mengisi daftar layanan tanpa memerlukan pengaturan tambahan.

Untuk melihat daftar layanan yang ditemukan dan memantau kondisi kesehatan mereka

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Sinyal Aplikasi, Layanan.
3. Untuk melihat layanan, operasinya, dan dependensinya, pilih nama dari salah satu layanan yang ada dalam daftar.

   Tampilan terpadu dan terpusat pada aplikasi ini membantu memberikan perspektif penuh tentang bagaimana pengguna berinteraksi dengan layanan Anda. Hal ini dapat membantu Anda dalam melakukan triase masalah jika terjadi anomali performa. Untuk detail selengkapnya tentang tampilan Layanan, silakan lihat Memantau kondisi kesehatan operasional aplikasi Anda dengan Sinyal Aplikasi.
4. Pilih tab Operasi Layanan untuk melihat metrik aplikasi standar untuk operasi-operasi layanan tersebut. Operasi-operasi tersebut adalah operasi API yang dipanggil layanan, misalnya.

   Kemudian, untuk dapat melihat grafik untuk satu operasi layanan itu, pilih nama operasi itu.
5. Pilih tab Dependensi untuk melihat dependensi yang dimiliki aplikasi Anda, bersama dengan metrik aplikasi penting untuk setiap dependensi. Dependensi mencakup AWS layanan dan layanan pihak ketiga yang dipanggil aplikasi Anda.
6. Untuk dapat melihat jejak berkorelasi dari halaman detail layanan, pilih sebuah titik data di salah satu dari tiga grafik di atas tabel. Hal ini akan mengisi panel baru dengan jejak yang difilter dari periode waktu. Jejak ini diurutkan dan difilter berdasarkan grafik yang Anda pilih. Sebagai contoh, jika Anda memilih grafik Latensi, maka jejak akan diurutkan berdasarkan waktu respons layanan.
7. Di panel navigasi CloudWatch konsol, pilih SLOs. Anda melihat SLO yang dibuat skrip untuk aplikasi sampel. Untuk informasi selengkapnya tentang SLO, silakan lihat Tujuan tingkat layanan (SLO).

(Opsional) Langkah 5: Pembersihan

Setelah selesai menguji sinyal Aplikasi, Anda dapat menggunakan sebuah skrip yang disediakan oleh Amazon untuk membersihkan dan menghapus artefak yang dibuat di akun Anda untuk aplikasi

sampel. Untuk melakukan pembersihan, Anda bisa memasukkan perintah berikut. Ganti *new-cluster-name*dengan nama cluster yang Anda buat untuk aplikasi sampel, dan ganti *region* -name dengan nama AWS Region, seperti`us-west-1`.

```
cd application-signals-demo/scripts/eks/appsignals/one-step && ./cleanup.sh new-
cluster-name region-name
```

Mengaktifkan Sinyal Aplikasi di platform lain dengan pengaturan khusus

⚠ Sinyal Aplikasi ada di rilis Pratinjau. Jika Anda memiliki umpan balik tentang fitur ini, Anda dapat menghubungi kami di app-signals-feedback@amazon .com.

Aktifkan Sinyal CloudWatch Aplikasi pada platform selain Amazon EKS dengan menggunakan langkah-langkah pengaturan khusus di bagian ini. Pada arsitektur ini, Anda menginstal dan mengkonfigurasi CloudWatch agen dan AWS Distro untuk OpenTelemetry diri Anda sendiri.

Pada arsitektur ini, Sinyal Aplikasi tidak secara otomatis menemukan nama layanan Anda atau klaster atau hostnya. Anda harus menentukan nama-nama ini selama melakukan pengaturan kustom, dan nama yang Anda tentukan adalah apa yang akan ditampilkan pada dasbor Sinyal Aplikasi.

Topik

- Gunakan pengaturan khusus untuk mengaktifkan Sinyal Aplikasi di Amazon ECS
- Gunakan pengaturan khusus untuk mengaktifkan Sinyal Aplikasi di Amazon EC2 dan platform lainnya

Gunakan pengaturan khusus untuk mengaktifkan Sinyal Aplikasi di Amazon ECS

⚠ Sinyal Aplikasi ada di rilis Pratinjau. Jika Anda memiliki umpan balik tentang fitur ini, Anda dapat menghubungi kami di app-signals-feedback@amazon .com.

Gunakan petunjuk penyiapan khusus ini untuk memasukkan aplikasi Anda di Amazon ECS ke Sinyal CloudWatch Aplikasi. Anda menginstal dan mengkonfigurasi CloudWatch agen dan AWS Distro untuk OpenTelemetry diri Anda sendiri.

Pada klaster Amazon ECS, Sinyal Aplikasi tidak secara otomatis menemukan nama layanan Anda atau klaster tempat layanan berjalan. Anda harus menentukan nama-nama ini selama melakukan pengaturan kustom, dan nama yang Anda tentukan adalah apa yang akan ditampilkan pada dasbor Sinyal Aplikasi.

> **Important**
>
> Hanya mode jaringan `awsvpc` yang didukung.

Langkah 1: Mengaktifkan Sinyal Aplikasi di akun Anda

Jika Anda belum mengaktifkan Sinyal Aplikasi di akun ini, Anda harus memberikan izin yang diperlukan Sinyal Aplikasi untuk menemukan layanan Anda. Untuk melakukan hal itu, lakukan hal berikut. Anda perlu melakukan ini sekali untuk akun Anda.

Untuk mengaktifkan Sinyal Aplikasi untuk aplikasi-aplikasi Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Layanan.
3. Pilih Mulai menemukan Layanan Anda.
4. Pilih kotak centang kemudian pilih Mulai menemukan Layanan.

   Menyelesaikan langkah ini untuk pertama kalinya di akun Anda akan membuat peran AWSServiceRoleForCloudWatchApplicationSignalsterkait layanan. Peran ini akan memberi Sinyal Aplikasi izin-izin berikut:

   - `xray:GetServiceGraph`
   - `logs:StartQuery`
   - `logs:GetQueryResults`
   - `cloudwatch:GetMetricData`
   - `cloudwatch:ListMetrics`
   - `tag:GetResources`

   Untuk informasi selengkapnya tentang peran ini, silakan lihat Izin peran terkait layanan untuk Sinyal Aplikasi CloudWatch .

Langkah 2: Membuat peran IAM

Anda harus membuat dua peran IAM. Jika Anda sudah membuat peran-peran ini, Anda mungkin perlu menambahkan izin untuk peran-peran tersebut.

- Peran tugas ECS— Kontainer menggunakan peran ini untuk beroperasi. Izin harus apa pun yang dibutuhkan aplikasi Anda, plus CloudWatchAgentServerPolicydan AWSXRayWriteOnlyAccess.
- Peran eksekusi tugas ECS— Amazon ECS menggunakan peran ini untuk meluncurkan dan mengeksekusi kontainer Anda. Jika Anda telah membuat peran ini, lampirkan AmazonSSM, ReadOnlyAccess AmazonECS, dan kebijakan ke dalamnya. TaskExecutionRolePolicy CloudWatchAgentServerPolicy

  Jika Anda ingin menyimpan data yang lebih sensitif untuk digunakan oleh Amazon ECS, silakan lihat Menentukan Data Sensitif untuk informasi selengkapnya.

Untuk informasi selengkapnya tentang membuat peran IAM, silakan lihat Membuat Peran IAM.

Langkah 3: Siapkan konfigurasi CloudWatch agen

Pertama, siapkan konfigurasi agen dengan Sinyal Aplikasi yang diaktifkan. Untuk melakukan hal ini, buat file lokal bernama `/tmp/ecs-cwagent.json`.

```
{
  "traces": {
    "traces_collected": {
      "app_signals": {}
    }
  },
  "logs": {
    "metrics_collected": {
      "app_signals": {}
    }
  }
}
```

Kemudian unggah konfigurasi ini ke SSM Parameter Store. Untuk melakukan hal ini, masukkan perintah berikut. Dalam file tersebut, ganti *$WILAYAH* dengan nama WIlayah Anda yang sebenarnya.

```
aws ssm put-parameter \
--name "ecs-cwagent" \
--type "String" \
```

```
--value "`cat /tmp/ecs-cwagent.json`" \
--region "$REGION"
```

Langkah 4: Instrumentasikan aplikasi Anda dengan CloudWatch agen

Untuk instrumen aplikasi Anda di Amazon ECS dengan agen CloudWatch

1. Pertama, tentukan bind mount. Volume akan digunakan untuk berbagi file di seluruh kontainer pada langkah-langkah selanjutnya. Anda akan menggunakan bind mount ini nanti dalam prosedur ini.

   ```
   "volumes": [
     {
       "name": "opentelemetry-auto-instrumentation"
     }
   ]
   ```

2. Tambahkan definisi sespan CloudWatch agen. Untuk melakukan hal ini, tambahkan sebuah kontainer baru yang bernama `ecs-cwagent` ke definisi tugas aplikasi Anda. Ganti *$WILAYAH* dengan nama WIlayah Anda yang sebenarnya. Ganti dengan jalur ke gambar CloudWatch kontainer terbaru di Amazon Elastic Container Registry. Untuk informasi selengkapnya, silakan lihat cloudwatch-agent di Amazon ECR.

   ```
   {
     "name": "ecs-cwagent",
     "image": "$IMAGE",
     "essential": true,
     "secrets": [
       {
         "name": "CW_CONFIG_CONTENT",
         "valueFrom": "ecs-cwagent"
       }
     ],
     "logConfiguration": {
       "logDriver": "awslogs",
       "options": {
         "awslogs-create-group": "true",
         "awslogs-group": "/ecs/ecs-cwagent",
         "awslogs-region": "$REGION",
         "awslogs-stream-prefix": "ecs"
       }
     }
   ```

```
}
```

3. Tambahkan sebuah kontainer baru yang bernama `init` ke penetapan tugas aplikasi Anda. Ganti *$IMAGE* dengan gambar terbaru dari repositori gambar AWS Distro untuk OpenTelemetry Amazon ECR.

```
{
  "name": "init",
  "image": "$IMAGE",
  "essential": false,
  "command": [
    "cp",
    "/javaagent.jar",
    "/otel-auto-instrumentation/javaagent.jar"
  ],
  "mountPoints": [
    {
      "sourceVolume": "opentelemetry-auto-instrumentation",
      "containerPath": "/otel-auto-instrumentation",
      "readOnly": false
    }
  ]
}
```

4. Tambahkan variabel lingkungan berikut ke kontainer aplikasi Anda. Untuk informasi selengkapnya, silakan lihat

Variabel Lingkungan	Pengaturan untuk mengaktifkan Sinyal Aplikasi
`OTEL_RESOURCE_ATTRIBUTES`	Ganti `$SVC_NAME` dengan nama aplikasi Anda. Ini akan ditampilkan sebagai nama aplikasi di dasbor Sinyal Aplikasi. Ganti `$HOST_ENV` dengan lingkungan host tempat aplikasi Anda sedang berjalan. Ini akan diputar sebagai Ditempati Di lingkungan aplikasi Anda di dasbor Sinyal Aplikasi.

Variabel Lingkungan	Pengaturan untuk mengaktifkan Sinyal Aplikasi
OTEL_SMP_ENABLED	Setel `true` untuk mengaktifkan Sinyal Aplikasi SpanMetricsProcessor.
OTEL_METRICS_EXPORTER	Atur none untuk menonaktifkan pengekspor metrik lainnya.
OTEL_AWS_SMP_EXPORTER_ENDPOINT	Setel `http://127.0.0.1:4315` untuk mengirim metrik ke CloudWatch sespan.
OTEL_EXPORTER_OTLP_TRACES_ENDPOINT	Setel `http://127.0.0.1:4315` untuk mengirim jejak ke CloudWatch sespan.
OTEL_TRACES_SAMPLER	Tentukan X-Ray sebagai sampler jejak.
OTEL_PROPAGATORS	Tambahkan X-Ray sebagai salah satu propagator.
JAVA_TOOL_OPTIONS	Suntikkan AWS Distro untuk agen OpenTelemetry Java.

5. Pasang volume `opentelemetry-auto-instrumentation` yang sudah Anda tentukan pada langkah 1 prosedur ini.

```
{
  "name": "app",
   ...
  "environment": [
    {
      "name": "OTEL_RESOURCE_ATTRIBUTES",
      "value": "aws.hostedin.environment=$HOST_ENV,service.name=$SVC_NAME"
    },
    {
      "name": "OTEL_SMP_ENABLED",
      "value": "true"
    },
    {
      "name": "OTEL_METRICS_EXPORTER",
      "value": "none"
```

```
    },
    {
      "name": "JAVA_TOOL_OPTIONS",
      "value": " -javaagent:/otel-auto-instrumentation/javaagent.jar"
    },
    {
      "name": "OTEL_AWS_SMP_EXPORTER_ENDPOINT",
      "value": "http://127.0.0.1:4315"
    },
    {
      "name": "OTEL_TRACES_SAMPLER",
      "value": "xray"
    },
    {
      "name": "OTEL_EXPORTER_OTLP_TRACES_ENDPOINT",
      "value": "http://127.0.0.1:4315"
    },
    {
      "name": "OTEL_PROPAGATORS",
      "value": "tracecontext,baggage,b3,xray"
    }
  ],
  "mountPoints": [
    {
      "sourceVolume": "opentelemetry-auto-instrumentation",
      "containerPath": "/otel-auto-instrumentation",
      "readOnly": false
    }
  ]
}
```

Langkah 5: Menerapkan aplikasi Anda

Buat revisi baru penetapan tugas Anda dan terapkan ke klaster aplikasi Anda. Anda akan melihat tiga kontainer dalam tugas yang baru saja Anda buat:

- `init`
- `ecs-cwagent`
- `app`

Gunakan pengaturan khusus untuk mengaktifkan Sinyal Aplikasi di Amazon EC2 dan platform lainnya

⚠ Sinyal Aplikasi ada di rilis Pratinjau. Jika Anda memiliki umpan balik tentang fitur ini, Anda dapat menghubungi kami di app-signals-feedback@amazon .com.

Untuk aplikasi yang berjalan di Amazon EC2 dan arsitektur lain yang bukan Amazon EKS, Anda menginstal dan mengonfigurasi CloudWatch agen dan AWS Distro untuk Anda sendiri. OpenTelemetry Pada arsitektur ini yang diaktifkan dengan pengaturan Sinyal Aplikasi, Sinyal Aplikasi tidak secara otomatis menemukan nama layanan Anda atau klaster atau host tempat berjalannya. Anda harus menentukan nama-nama ini selama melakukan pengaturan kustom, dan nama yang Anda tentukan adalah apa yang akan ditampilkan pada dasbor Sinyal Aplikasi.

Langkah-langkah berikut telah diuji pada instans Amazon EC2, tetapi juga diharapkan dapat bekerja pada arsitektur lain yang mendukung Distro. AWS OpenTelemetry

Persyaratan

- Saat ini, Sinyal Aplikasi hanya memantau aplikasi Java.
- Untuk mendapatkan dukungan untuk Sinyal Aplikasi, Anda harus menggunakan versi terbaru dari CloudWatch agen dan AWS Distro untuk OpenTelemetry agen.
- Anda harus AWS CLI menginstal pada instance. Kami merekomendasikan AWS CLI versi 2, tetapi versi 1 juga harus berfungsi. Untuk informasi selengkapnya tentang menginstal AWS CLI, lihat Menginstal atau memperbarui versi terbaru AWS CLI.

⚠ Important

Jika Anda sudah menggunakan aplikasi OpenTelemetry yang ingin Anda aktifkan untuk Sinyal Aplikasi, lihat OpenTelemetry pertimbangan kompatibilitas sebelum Anda mengaktifkan Sinyal Aplikasi.

Aplikasi Python mendukung akses awal tersedia. Untuk petunjuk penyiapan, hubungi kami di app-signals-feedback@amazon .com.

Langkah 1: Mengaktifkan Sinyal Aplikasi di akun Anda

Jika Anda belum mengaktifkan Sinyal Aplikasi di akun ini, Anda harus memberikan izin yang diperlukan Sinyal Aplikasi untuk menemukan layanan Anda. Untuk melakukan hal itu, lakukan hal berikut. Anda perlu melakukan ini sekali untuk akun Anda.

Untuk mengaktifkan Sinyal Aplikasi untuk aplikasi-aplikasi Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Layanan.
3. Pilih Mulai menemukan Layanan Anda.
4. Pilih kotak centang kemudian pilih Mulai menemukan Layanan.

   Menyelesaikan langkah ini untuk pertama kalinya di akun Anda akan membuat peran AWSServiceRoleForCloudWatchApplicationSignalsterkait layanan. Peran ini akan memberi Sinyal Aplikasi izin-izin berikut:

   - `xray:GetServiceGraph`
   - `logs:StartQuery`
   - `logs:GetQueryResults`
   - `cloudwatch:GetMetricData`
   - `cloudwatch:ListMetrics`
   - `tag:GetResources`

   Untuk informasi selengkapnya tentang peran ini, silakan lihat Izin peran terkait layanan untuk Sinyal Aplikasi CloudWatch .

Langkah 2: Unduh dan mulai CloudWatch agen

Untuk menginstal CloudWatch agen sebagai bagian dari mengaktifkan Sinyal Aplikasi pada instans Amazon EC2

1. Unduh versi terbaru CloudWatch agen ke instans. Jika instans sudah menginstal CloudWatch agen, Anda mungkin perlu memperbaruinya. Hanya versi agen yang dirilis pada 30 November 2023 atau yang lebih baru yang mendukung Sinyal CloudWatch Aplikasi.

   Untuk informasi tentang mengunduh CloudWatch agen, lihatUnduh paket CloudWatch agen.

2. Sebelum Anda memulai CloudWatch agen, konfigurasikan untuk mengaktifkan Sinyal Aplikasi. Contoh berikut adalah konfigurasi CloudWatch agen yang memungkinkan Sinyal Aplikasi untuk metrik dan jejak pada host EC2.

   Anda dapat membuat file ini dengan memasukkan perintah berikut:

   ```
   vim amazon-cloudwatch-agent.json
   ```

   Tambahkan yang berikut ini sebagai isi file ini.

   ```
   {
     "traces": {
       "traces_collected": {
         "app_signals": {}
       }
     },
     "logs": {
       "metrics_collected": {
         "app_signals": {}
       }
     }
   }
   ```

3. Lampirkan kebijakan CloudWatchAgentServerPolicydan AWSXrayWriteOnlyAccessIAM ke peran IAM instans Amazon EC2 Anda.

   a. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.

   b. Pilih Peran dan temukan peran yang digunakan oleh instans Amazon EC2 Anda. Kemudian pilih nama dari peran itu.

   c. Pada tab Izin, pilih Tambahkan izin, Lampirkan kebijakan.

   d. Menemukan CloudWatchAgentServerPolicy. Gunakan kotak pencarian, jika diperlukan. Kemudian pilih kotak centang untuk kebijakan tersebut dan kemudian pilih Tambahkan izin.

   e. Menemukan AWSXrayWriteOnlyAccess. Gunakan kotak pencarian, jika diperlukan. Kemudian pilih kotak centang untuk kebijakan tersebut dan kemudian pilih Tambahkan izin.

4. Mulai CloudWatch agen dengan memasukkan perintah berikut. Ganti *agent-config-file-path*dengan path ke file konfigurasi CloudWatch agen, seperti `./amazon-cloudwatch-agent.json`. Anda harus menyertakan awalan `file:` seperti yang ditunjukkan.

```
export CONFIG_FILE_PATH=./amazon-cloudwatch-agent.json
```

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl \
-a fetch-config \
-m ec2 -s -c file:$CONFIG_FILE_PATH
```

Langkah 3: Menginstrumentasikan aplikasi Anda dan memulainya

Untuk melengkapi aplikasi Anda sebagai bagian dari mengaktifkan Sinyal Aplikasi di instans Amazon EC2

1. Unduh versi terbaru dari agen instrumentasi otomatis AWS Distro untuk OpenTelemetry Java. Anda dapat mengunduh versi terbaru dengan menggunakan tautan ini. Anda dapat melihat informasi tentang semua versi yang dirilis di aws-otel-java-instrumentation Rilis.
2. Untuk mengoptimalkan manfaat Sinyal Aplikasi, gunakan variabel lingkungan untuk memberikan informasi tambahan sebelum memulai aplikasi Anda. Informasi ini akan ditampilkan di dasbor Sinyal Aplikasi.

   a. Untuk variabel `OTEL_RESOURCE_ATTRIBUTES`, tentukan informasi berikut sebagai pasangan nilai kunci:

      - `aws.hostedin.environment` menetapkan lingkungan tempat aplikasi berjalan. Ini akan diputar sebagai Ditempati Di lingkungan aplikasi Anda di dasbor Sinyal Aplikasi. Kunci atribut ini hanya digunakan oleh Sinyal Aplikasi, dan diubah menjadi anotasi jejak X-Ray dan dimensi CloudWatch metrik. Jika Anda tidak memberikan nilai untuk kunci ini, maka `Generic` digunakan sebagai nilai default.
      - `service.name` menetapkan nama layanan. Ini akan diputar sebagai nama layanan di dasbor Sinyal Aplikasi Anda. Jika Anda tidak memberikan nilai untuk kunci ini, maka `unknown_service` digunakan sebagai nilai default.

   b. Untuk variabel `OTEL_EXPORTER_OTLP_TRACES_ENDPOINT`, tentukan URL titik akhir dasar tempat jejak akan diekspor. CloudWatch Agen mengekspos 4315 sebagai port OLTP. Di Amazon EC2, karena aplikasi berkomunikasi dengan CloudWatch agen lokal, Anda harus menetapkan nilai ini `OTEL_EXPORTER_OTLP_TRACES_ENDPOINT=http://localhost:4315`

c. Untuk variabel `OTEL_AWS_SMP_EXPORTER_ENDPOINT`, tentukan URL titik akhir dasar tempat metrik akan diekspor. CloudWatch Agen mengekspos 4315 sebagai port OLTP. Di Amazon EC2, karena aplikasi berkomunikasi dengan CloudWatch agen lokal, Anda harus menetapkan nilai ini `OTEL_AWS_SMP_EXPORTER_ENDPOINT=http://localhost:4315`

d. Untuk `JAVA_TOOL_OPTIONS` variabel, tentukan jalur tempat agen instrumentasi otomatis AWS Distro untuk OpenTelemetry Java disimpan.

```
export JAVA_TOOL_OPTIONS=' -javaagent:$ADOT_AGENT_PATH'
```

Sebagai contoh:

```
export ADOT_AGENT_PATH=./aws-opentelemetry-agent.jar
```

e. Untuk variabel `OTEL_METRICS_EXPORTER`, kami sarankan Anda menetapkan nilainya menjadi `none`. Hal ini akan menonaktifkan pengekspor metrik lain sehingga hanya pengekspor Sinyal Aplikasi yang digunakan.

f. Untuk `OTEL_SMP_ENABLED` variabel, aktifkan SpanMetricProcessor (SMP) dengan menyetel `OTEL_SMP_ENABLED` ke`true`. Ini akan menghasilkan metrik Sinyal Aplikasi dari jejak.

3. Mulai aplikasi Anda dengan variabel-variabel lingkungan yang dibahas pada langkah sebelumnya. Berikut ini adalah sebuah contoh skrip awal.

```
JAVA_TOOL_OPTIONS=' -javaagent:$ADOT_AGENT_PATH' \
OTEL_METRICS_EXPORTER=none \
OTEL_SMP_ENABLED=true \
OTEL_AWS_SMP_EXPORTER_ENDPOINT=http://localhost:4315 \
OTEL_EXPORTER_OTLP_TRACES_ENDPOINT=http://localhost:4315 \
OTEL_RESOURCE_ATTRIBUTES=aws.hostedin.environment=$YOUR_HOST_ENV,service.name=
$YOUR_SVC_NAME \
java -jar $MY_JAVA_APP.jar
```

Melakukan pemecahan masalah instalasi yang terjadi pada Sinyal Aplikasi Anda

⚠ Sinyal Aplikasi ada di rilis Pratinjau. Jika Anda memiliki umpan balik tentang fitur ini, Anda dapat menghubungi kami di app-signals-feedback@amazon .com.

Bagian ini berisi tips pemecahan masalah untuk Sinyal CloudWatch Aplikasi.

Topik

Aplikasi tidak memulai setelah Sinyal Aplikasi diaktifkan

Jika aplikasi Anda di klaster Amazon EKS tidak memulai setelah Anda mengaktifkan Sinyal Aplikasi di klaster, Anda perlu memeriksa hal-hal berikut:

- Periksa apakah aplikasi telah diinstrumentasi oleh solusi pemantauan yang lain. Sinyal Aplikasi tidak mendukung aktif bersamaan dengan solusi instrumentasi lainnya.
- Konfirmasikan bahwa aplikasi Anda sudah memenuhi persyaratan kompatibilitas untuk menggunakan Sinyal Aplikasi. Untuk informasi selengkapnya, silakan lihat Sistem yang didukung oleh Sinyal Aplikasi .
- Jika aplikasi Anda gagal menarik artefak Sinyal Aplikasi seperti AWS Distro untuk agen OpenTelemetery Java dan gambar CloudWatch agen, itu bisa menjadi masalah jaringan.

Untuk mengurangi masalah, hapus keterangan `instrumentation.opentelemetry.io/inject-java: "true"` dari manifes deployment aplikasi Anda, dan terapkan ulang aplikasi Anda. Kemudian periksa apakah aplikasi sudah berfungsi.

Data telemetri hilang dan X-Ray CloudWatch

Jika ada metrik atau jejak yang hilang di dasbor Sinyal Aplikasi, hal-hal berikut ini mungkin menjadi penyebabnya. Selidiki penyebab-penyebab tersebut hanya jika Anda telah menunggu 15 menit untuk Sinyal Aplikasi mengumpulkan dan menampilkan data sejak pembaruan terakhir Anda.

- Pastikan pustaka dan kerangka kerja yang Anda gunakan didukung oleh agen ADOT Java. Untuk informasi selengkapnya, silakan lihat Pustaka / Kerangka Kerja.
- Pastikan CloudWatch agen sedang berjalan. Pertama periksa status pod CloudWatch agen dan pastikan semuanya dalam `Running` status.

  ```
  kubectl -n amazon-cloudwatch get pods.
  ```

  Tambahkan berikut ini ke file konfigurasi CloudWatch agen untuk mengaktifkan log debugging, dan kemudian restart agen.

  ```
  "agent": {
    "region": "${REGION}",
    "debug": true
  },
  ```

  Kemudian periksa kesalahan pada pod CloudWatch agen.
- Periksa masalah konfigurasi dengan CloudWatch agen. Konfirmasikan bahwa berikut ini masih dalam file konfigurasi CloudWatch agen dan agen telah dimulai ulang sejak ditambahkan.

  ```
  "agent": {
    "region": "${REGION}",
    "debug": true
  },
  ```

  Kemudian periksa log OpenTelemetry debugging untuk pesan kesalahan sepertiERROR `io.opentelemetry.exporter.internal.grpc.OkHttpGrpcExporter - Failed to export ...`. Pesan-pesan ini mungkin menunjukkan masalah.

  Jika itu tidak menyelesaikan masalah, buang dan periksa variabel lingkungan dengan nama yang dimulai `OTEL_` dengan menjelaskan pod dengan perintah `kubectl describe pod`.
- Periksa izin yang salah atau tidak memadai untuk mengekspor data dari agen. CloudWatch Jika Anda melihat `Access Denied` pesan di log CloudWatch agen, ini mungkin masalahnya. Ada

kemungkinan bahwa izin yang diterapkan saat Anda menginstal CloudWatch agen kemudian diubah atau dicabut.

- Periksa masalah AWS Distro untuk OpenTelemetry (ADOT) saat membuat data telemetri.

  Pastikan bahwa keterangan instrumentasi `instrumentation.opentelemetry.io/inject-java` dan `sidecar.opentelemetry.io/inject-java` diterapkan pada penerapan aplikasi deployment dan nilainya adalah `true`. Tanpa ini, pod aplikasi tidak akan diinstrumentasi bahkan jika add-on ADOT sudah diinstal dengan benar.

  Berikutnya, periksa apakah kontainer `Init` diterapkan pada aplikasi dan status `Ready` adalah `True`. Jika kontainer `init` belum siap, silakan lihat statusnya untuk alasannya.

  Jika masalah berlanjut, lakukan hal berikut untuk mengaktifkan logging debug di OpenTelemetry Java SDK. Kemudian cari pesan yang dimulai dengan `ERROR io.telemetry`.

  Untuk mengaktifkan logging debug, atur variabel lingkungan `OTEL_JAVAAGENT_DEBUG` ke benar dan terapkan ulang aplikasi.
- Pengekspor metrik/rentang mungkin membuang data. Untuk mengetahuinya, periksa log aplikasi untuk pesan yang menyertakan `Failed to export...`
- CloudWatch Agen mungkin terhambat saat mengirim metrik atau bentang ke Sinyal Aplikasi. Periksa pesan yang menunjukkan pelambatan di log CloudWatch agen.

Metrik-metrik dependensi memiliki nilai Tidak Dikenal

Jika Anda melihat UnknownOperation, UnknownRemoteService, atau UnknownRemoteOperationuntuk nama ketergantungan atau operasi di dasbor Sinyal Aplikasi, periksa apakah kemunculan titik data untuk layanan jarak jauh yang tidak diketahui dan operasi jarak jauh yang tidak diketahui bertepatan dengan penerapannya. Ini adalah sebuah masalah yang diketahui pada Sinyal Aplikasi dan direncanakan untuk diperbaiki dalam rilis yang akan datang.

Menangani ConfigurationConflict saat mengelola add-on Amazon CloudWatch Observability EKS

Saat Anda menginstal atau memperbarui add-on Amazon CloudWatch Observability EKS, jika Anda melihat kegagalan yang `Health Issue` disebabkan oleh tipe `ConfigurationConflict` dengan deskripsi yang dimulai`Conflicts found when trying to apply. Will not continue due to resolve conflicts mode`, kemungkinan karena Anda sudah memiliki CloudWatch agen dan komponen terkait seperti ServiceAccount, ClusterRole dan yang ClusterRoleBinding diinstal pada

cluster. Ketika add-on mencoba menginstal CloudWatch agen dan komponen terkait, jika mendeteksi perubahan apa pun dalam konten, secara default gagal instalasi atau pembaruan untuk menghindari penimpaan status sumber daya pada cluster.

Jika Anda mencoba untuk onboard ke add-on Amazon CloudWatch Observability EKS dan Anda melihat kegagalan ini, kami sarankan untuk menghapus penyiapan CloudWatch agen yang ada yang sebelumnya Anda instal di cluster dan kemudian menginstal add-on EKS. Pastikan untuk mencadangkan penyesuaian apa pun yang mungkin telah Anda buat ke penyiapan CloudWatch agen asli seperti konfigurasi agen khusus, dan berikan ini ke add-on Amazon CloudWatch Observability EKS saat Anda menginstal atau memperbaruinya berikutnya. Jika sebelumnya Anda telah menginstal CloudWatch agen untuk orientasi ke Container Insights, lihat Menghapus CloudWatch agen dan Fluentd untuk Wawasan Kontainer untuk informasi selengkapnya.

Atau, add-on mendukung opsi konfigurasi resolusi konflik yang memiliki kemampuan untuk menentukan `OVERWRITE`. Anda dapat menggunakan opsi ini untuk melanjutkan dengan melakukan instalasi atau memperbarui add-on dengan menimpa konflik di klaster. Jika Anda menggunakan konsol Amazon EKS, Anda akan menemukan Metode penyelesaian konflik saat Anda memilih Pengaturan konfigurasi opsional ketika Anda membuat atau memperbarui add-on. Jika Anda menggunakan AWS CLI, Anda dapat memberikan perintah Anda `--resolve-conflicts OVERWRITE` untuk membuat atau memperbarui add-on.

Mengonfigurasi Sinyal Aplikasi

> ⚠ Sinyal Aplikasi ada di rilis Pratinjau. Jika Anda memiliki umpan balik tentang fitur ini, Anda dapat menghubungi kami di app-signals-feedback@amazon .com.

Bagian ini berisi informasi tentang mengkonfigurasi Sinyal CloudWatch Aplikasi.

Lacak laju sampling

Secara default, saat Anda mengaktifkan sampling terpusat X-Ray Sinyal Aplikasi diaktifkan menggunakan pengaturan laju sampling default `reservoir=1/s` dan `fixed_rate=5%`. Variabel lingkungan untuk agen SDK AWS Distro for OpenTelemetry (ADOT) ditetapkan sebagai berikut.

Variabel lingkungan	Nilai	Catatan
`OTEL_TRACES_SAMPLER`	`xray`	

Variabel lingkungan	Nilai	Catatan
`OTEL_TRACES_SAMPLER_ARG`	`endpoint=http://cloudwatch-agent.amazon-cloudwatch:2000`	Titik akhir agen CloudWatch

Untuk informasi tentang cara mengubah konfigurasi pengambilan sampel, silakan lihat yang berikut ini:

- Untuk mengubah sampling X-Ray, silakan lihat Menyesuaikan aturan sampling
- Untuk mengubah pengambilan sampel ADOT, lihat Mengonfigurasi Kolektor OpenTelemetry untuk pengambilan sampel jarak jauh X-Ray

Jika Anda ingin menonaktifkan sampling terpusat X-Ray dan menggunakan sampling lokal sebagai gantinya, tetapkan nilai berikut untuk agen ADOT SDK Java seperti di bawah ini. Contoh berikut menetapkan laju pengambilan sampel pada 5%.

Variabel lingkungan	Nilai
`OTEL_TRACES_SAMPLER`	`parentbased_traceidratio`
`OTEL_TRACES_SAMPLER_ARG`	`0.05`

Untuk informasi tentang pengaturan pengambilan sampel lanjutan lainnya, silakan lihat OTEL_TRACES_SAMPLER.

Mengurangi kardinalitas tinggi

File konfigurasi CloudWatch agen dapat menyertakan bidang opsional yang dapat mengurangi kardinalitas tinggi dalam metrik dan jejak yang dikumpulkannya. Untuk informasi selengkapnya, lihat Aktifkan Sinyal CloudWatch Aplikasi.

Tujuan tingkat layanan (SLO)

⚠ Sinyal Aplikasi ada di rilis Pratinjau. Jika Anda memiliki umpan balik mengenai fitur ini, Anda dapat menghubungi kami di app-signals-feedback@amazon.com.

Anda dapat menggunakan Sinyal Aplikasi untuk membuat tujuan tingkat layanan bagi layanan-layanan untuk operasi bisnis penting Anda. Dengan membuat SLO pada layanan ini, Anda akan dapat melacaknya di dasbor SLO, sehingga memberi pandangan sekilas tentang operasi yang paling penting bagi Anda.

Selain membuat tampilan cepat yang dapat digunakan operator Anda untuk melihat status operasi kritis saat ini, Anda dapat menggunakan SLO untuk melacak performa jangka panjang layanan Anda, untuk memastikan bahwa layanan tersebut memenuhi harapan Anda. Jika Anda memiliki perjanjian tingkat layanan dengan pelanggan, SLO merupakan alat yang hebat untuk memastikan bahwa perjanjian tersebut terpenuhi.

Menilai kondisi kesehatan layanan Anda dengan SLO dimulai dengan menetapkan tujuan yang jelas dan terukur berdasarkan metrik-metrik performa utama— indikator tingkat layanan (SLI). SLO melacak performa SLI terhadap ambang batas dan sasaran yang Anda tetapkan, dan melaporkan seberapa jauh atau seberapa dekat performa aplikasi Anda dengan ambang batas.

Sinyal Aplikasi membantu Anda mengatur SLO pada metrik performa utama Anda. Sinyal Aplikasi secara otomatis mengumpulkan metrik `Latency` dan `Availability` untuk setiap layanan dan operasi yang ditemukannya, dan metrik-metrik ini seringkali ideal untuk digunakan sebagai SLI. Dengan pemandu pembuatan SLO, Anda dapat menggunakan metrik ini untuk SLO Anda. Anda kemudian dapat melacak status semua SLO Anda dengan menggunakan dasbor Sinyal Aplikasi.

Anda dapat mengatur SLO pada operasi tertentu yang dipanggil atau digunakan layanan Anda. Anda dapat menggunakan metrik CloudWatch atau ekspresi metrik sebagai SLI, selain menggunakan metrik `Latency` dan `Availability`.

Membuat SLO adalah langkah yang sangat penting untuk mendapatkan manfaat maksimal dari Sinyal Aplikasi CloudWatch. Setelah membuat SLO, Anda dapat melihat statusnya di konsol Sinyal Aplikasi untuk melihat layanan dan operasi penting Anda yang berperforma baik dan mana yang tidak sehat dengan cepat. Memiliki SLO untuk dilacak akan memberikan manfaat-manfaat utama berikut:

- Lebih mudah bagi para operator layanan Anda untuk melihat kondisi kesehatan operasional saat ini dari layanan kritis yang diukur berdasarkan SLI. Kemudian mereka dapat dengan cepat melakukan penilaian awal dan mengidentifikasi layanan dan operasi yang sedang dalam kondisi tidak sehat.
- Anda dapat melacak performa layanan Anda terhadap sasaran bisnis yang terukur dalam jangka waktu yang lebih lama.

Dengan memilih apa yang akan diatur SLO, Anda dapat memprioritaskan hal apa saja yang penting bagi Anda. Dasbor Sinyal Aplikasi secara otomatis akan menyajikan informasi mengenai apa yang telah Anda prioritaskan.

Saat membuat sebuah SLO, Anda juga dapat memilih untuk membuat alarm-alarm CloudWatch secara bersamaan untuk memantau SLO tersebut. Anda dapat mengatur alarm yang memantau terjadinya pelanggaran ambang batas, dan juga alarm untuk tingkat-tingkat peringatan. Alarm-alarm ini dapat secara otomatis memberikan notifikasi kepada Anda jika metrik-metrik SLO melanggar ambang batas yang Anda tetapkan, atau jika mendekati ambang peringatan. Misalnya, SLO yang mendekati ambang peringatannya dapat memberi tahu Anda bahwa tim Anda mungkin perlu memperlambat churn dalam aplikasi untuk memastikan bahwa tujuan performa jangka panjang terpenuhi.

Topik

Konsep-konsep SLO

Suatu SLO mencakup komponen-komponen berikut:

- Indikator tingkat layanan (SLI), yang merupakan sebuah metrik performa utama yang Anda tentukan. Ini mewakili tingkat performa yang diinginkan untuk aplikasi Anda. Sinyal Aplikasi secara otomatis mengumpulkan metrik kunci `Latency` dan `Availability` untuk setiap layanan dan operasi yang ditemukannya, dan metrik-metrik kunci ini seringkali menjadi metrik yang ideal untuk mengatur SLO.

Anda memilih ambang batas yang akan Anda gunakan untuk SLI Anda. Seperti, 200 ms untuk latensi.

- Tujuan atau tujuan pencapaian, yang merupakan persentase waktu yang diharapkan SLI untuk memenuhi ambang batas selama setiap interval waktu. Interval waktu tersebut bisa dalam hitungan jam atau selama setahun.

  Interval dapat berupa interval kalender atau interval bergulir.

  - Interval kalender diselaraskan dengan kalender, seperti suatu SLO yang dilacak per bulan. CloudWatch secara otomatis akan menyesuaikan kondisi kesehatan, anggaran, dan angka pencapaian berdasarkan jumlah hari dalam sebulan. Interval kalender lebih cocok untuk tujuan-tujuan bisnis yang diukur berdasarkan kalender yang sudah diselaraskan.
  - Interval bergulir dihitung secara bergulir. Interval bergulir lebih cocok untuk melakukan pelacakan terhadap pengalaman pengguna terbaru dari aplikasi Anda.
- Periode adalah jangka waktu yang lebih pendek, dan banyak periode membentuk interval. Performa aplikasi dibandingkan dengan SLI selama masing-masing periode dalam interval. Untuk setiap periode, aplikasi ditentukan telah mencapai atau tidak mencapai performa yang diperlukan.

Sebagai contoh, tujuan 99% dengan interval kalender satu hari dan periode 1 menit berarti bahwa aplikasi harus memenuhi atau mencapai ambang keberhasilan selama 99% dari periode 1 menit di siang hari. Jika ya, artinya SLO terpenuhi untuk hari itu. Hari berikutnya adalah interval evaluasi baru, dan aplikasi tersebut harus memenuhi atau mencapai ambang keberhasilan selama 99% dari periode 1 menit selama hari kedua untuk memenuhi SLO untuk hari kedua itu.

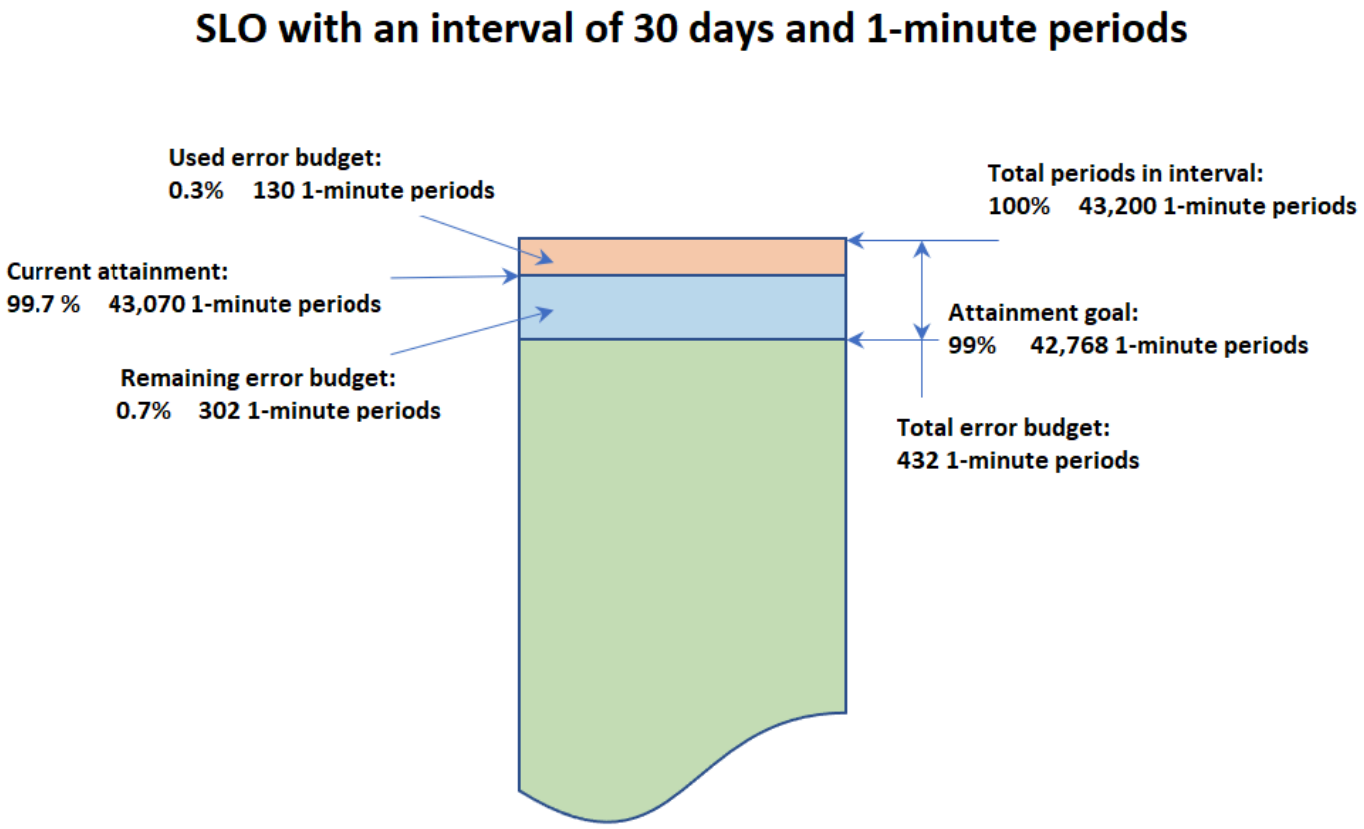
SLI dapat didasarkan pada salah satu metrik aplikasi standar baru yang dikumpulkan Sinyal Aplikasi. Atau, ini bisa berupa metrik CloudWatch atau ekspresi metrik apa pun. Metrik aplikasi standar yang dapat Anda gunakan untuk SLI adalah `Latency` dan `Availability`. `Availability` mewakili respons yang berhasil dibagi dengan total permintaan. Ini dihitung sebagai (1 - Tingkat Kegagalan)*100, di mana respons Kegagalan adalah kesalahan `5xx`. Respons keberhasilan adalah respons tanpa kesalahan `5XX`. Respons `4XX` dianggap berhasil.

> **Note**
>
> Saat ini hanya penghitungan berbasis periode yang didukung. Dukungan untuk penghitungan berbasis volume atau permintaan direncanakan untuk rilis masa depan.

Hitung anggaran kesalahan dan pencapaian

Ketika Anda melihat informasi tentang SLO, Anda melihat status kesehatan saat ini dan anggaran kesalahannya. Anggaran kesalahan adalah jumlah waktu dalam interval yang dapat menembus ambang batas tetapi tetap membiarkan SLO dipenuhi. Anggaran kesalahan total adalah jumlah total waktu pelanggaran yang dapat ditoleransi di seluruh interval. Sisa anggaran kesalahan adalah sisa jumlah waktu pelanggaran yang dapat ditoleransi selama interval saat ini. Ini setelah jumlah waktu pelanggaran yang telah terjadi telah dikurangi total anggaran kesalahan.

Gambar berikut menggambarkan konsep anggaran pencapaian dan kesalahan untuk suatu tujuan dengan interval 30 hari, periode 1 menit, dan tujuan pencapaian 99%. 30 hari mencakup 43.200 periode 1 menit. 99% dari 43.200 adalah 42.768, jadi 42.768 menit selama sebulan harus sehat agar SLO terpenuhi. Sejauh ini dalam interval saat ini, 130 dari periode 1 menit berada kondisi tidak sehat.



Menentukan keberhasilan dalam masing-masing periode

Dalam masing-masing periode, data SLI akan dikumpulkan menjadi satu titik data berdasarkan statistik yang digunakan untuk SLI. Titik data ini mewakili durasi periode seluruhnya. Titik data tunggal itu dibandingkan dengan ambang batas SLI untuk menentukan apakah periode tersebut

dalam kondisi sehat, atau tidak. Melihat periode yang tidak sehat selama rentang waktu saat ini di dasbor dapat mengingatkan para operator layanan Anda bahwa layanan perlu diprioritaskan.

Jika periode ditentukan tidak sehat, seluruh panjang periode dihitung sebagai gagal terhadap anggaran kesalahan. Melacak anggaran kesalahan memungkinkan Anda mengetahui apakah layanan mencapai performa yang Anda inginkan dalam jangka waktu yang lebih lama.

Membuat SLO

Kami menyarankan Anda untuk mengatur SLO latensi dan ketersediaan pada aplikasi penting Anda. Metrik yang dikumpulkan Sinyal Aplikasi ini selaras dengan tujuan bisnis bersama.

Anda juga dapat mengatur SLO pada metrik CloudWatch atau ekspresi matematika metrik apa pun yang menghasilkan satu deret waktu.

Cara membuat sebuah SLO

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Tujuan Tingkat Layanan (SLO).
3. Pilih Buat SLO.
4. Masukkan nama untuk SLO. Menyertakan nama layanan atau operasi, bersama kata kunci yang sesuai seperti latensi atau ketersediaan, akan membantu Anda mengidentifikasi apa yang ditunjukkan status SLO selama triase dengan cepat.
5. Untuk Mengatur Indikator Tingkat Layanan(SLI), lakukan salah satu hal berikut:
   - Untuk mengatur SLO pada salah satu metrik aplikasi standar `Latency` atau `Availability`:

     a. Pilih Operasi Layanan.

     b. Pilih layanan yang akan dipantau oleh SLO ini.

     c. Pilih operasi yang akan dipantau oleh SLO ini.

     Drop-down Pilih Layanan dan Pilih operasi diisi oleh layanan dan operasi yang telah aktif dalam 24 jam terakhir.

     d. Pilih Ketersediaan atau Latensi dan kemudian atur ambang batas.
   - Untuk mengatur SLO pada metrik CloudWatch atau ekspresi matematika metrik CloudWatch:

     a. Pilih Metrik CloudWatch.

b. Pilih Pilih metrik CloudWatch.

Layar Pilih metrik muncul. Gunakan tab Jelajahi atau Kueri untuk menemukan metrik yang Anda inginkan, atau membuat ekspresi matematika metrik.

Setelah Anda memilih metrik yang Anda inginkan, pilih tab Metrik bergrafik dan pilih Statistik dan Periode yang akan digunakan untuk SLO. Kemudian pilih Pilih metrik.

Untuk informasi selengkapnya tentang metrik ini, silakan lihat Membuat sebuah grafik metrik dan Tambahkan ekspresi matematika ke CloudWatch grafik.

c. Untuk Atur kondisi, pilih operator perbandingan dan ambang batas untuk SLO yang akan digunakan sebagai indikator keberhasilan.

6. Jika Anda memilih Operasi Layanan di langkah 5, Anda dapat memilih Pengaturan tambahan secara opsional dan kemudian melakukan penyesuaian pada panjang periode untuk SLO ini.
7. Atur interval dan tujuan pencapaian untuk SLO. Untuk informasi selengkapnya tentang interval dan pencapaian tujuan dan bagaimana keduanya bekerja sama, silakan lihat Konsep-konsep SLO.
8. (Opsional) Atur satu atau beberapa alarm CloudWatch atau ambang peringatan untuk SLO.

a. Alarm CloudWatch dapat menggunakan Amazon SNS untuk memberi tahu Anda secara proaktif jika aplikasi tidak sehat berdasarkan performa SLI-nya.

Untuk membuat alarm, pilih salah satu kotak centang alarm dan masukkan atau buat topik Amazon SNS yang akan digunakan untuk notifikasi saat alarm masuk ke status `ALARM`. Untuk informasi selengkapnya tentang alarm CloudWatch, silakan lihat Menggunakan CloudWatch alarm Amazon. Membuat alarm akan menimbulkan biaya. Untuk informasi selengkapnya tentang penetapan harga CloudWatch, silakan lihat Penetapan Harga Amazon CloudWatch.

b. Jika Anda mengatur ambang batas peringatan, peringatan ini muncul di layar Sinyal Aplikasi untuk membantu Anda mengidentifikasi SLO yang berisiko tidak terpenuhi, bahkan jika saat ini sehat.

Untuk mengatur ambang batas peringatan, masukkan nilai ambang batas di Ambang batas peringatan. Ketika anggaran kesalahan SLO lebih rendah dari ambang batas peringatan, SLO ditandai dengan Peringatan di beberapa layar Sinyal Aplikasi. Ambang batas peringatan juga muncul pada grafik anggaran kesalahan. Anda juga dapat membuat alarm peringatan SLO yang didasarkan pada ambang batas peringatan.

9. Untuk menambahkan tag ke SLO ini, silakan pilih tab Tag dan kemudian pilih Tambahkan tag baru. Tag dapat membantu Anda mengelola, mengidentifikasi, mengatur, dan memfilter sumber daya. Untuk informasi selengkapnya tentang penandaan, silakan lihat Menandai sumber daya AWS Anda.

   > **Note**
   >
   > Jika aplikasi yang terkait SLO ini terdaftar di AWS Service Catalog AppRegistry, Anda dapat menggunakan tag `awsApplication` untuk mengaitkan SLO ini dengan aplikasi tersebut di AppRegistry. Untuk informasi selengkapnya tentang ini, silakan lihat Apa itu AppRegistry?

10. Pilih Buat SLO. Jika Anda juga memilih untuk membuat satu atau beberapa alarm, nama tombol berubah sehingga mencerminkan hal ini.

Menampilkan dan melakukan penilaian awal pada status SLO

Anda dapat dengan cepat melihat kesehatan SLO Anda menggunakan Tujuan Tingkat Layanan atau opsi Layanan di konsol CloudWatch. Tampilan Layanan memberikan pandangan sekilas tentang rasio layanan yang tidak sehat, yang dihitung berdasarkan SLO yang telah Anda atur. Untuk informasi selengkapnya tentang penggunaan opsi Layanan, silakan lihat Memantau kondisi kesehatan operasional aplikasi Anda dengan Sinyal Aplikasi.

Tampilan Tujuan Tingkat Layanan memberikan sebuah tampilan makro organisasi Anda. Anda dapat melihat SLO yang terpenuhi dan tidak terpenuhi secara keseluruhan. Ini memberi Anda gambaran tentang berapa banyak layanan dan operasi Anda yang berperforma sesuai harapan Anda selama periode waktu yang lebih lama, sesuai dengan SLI yang Anda pilih.

Cara melihat semua SLO Anda dengan menggunakan tampilan Tujuan Tingkat Layanan

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Tujuan Tingkat Layanan (SLO).

   Daftar Tujuan Tingkat Layanan (SLO) ditampilkan.

   Anda dapat dengan cepat melihat status dari SLO Anda saat ini di kolom status SLI. Untuk mengurutkan SLO sehingga semua yang kondisinya tidak sehat berada di bagian atas daftar, pilih kolom status SLI sampai semua SLO yang sedang dalam kondisi tidak sehat berada di atas.

Tabel SLO memiliki kolom-kolom default berikut. Anda dapat menyesuaikan kolom-kolom mana saja yang ditampilkan dengan memilih ikon roda gigi yang ada di atas daftar. Untuk informasi selengkapnya tentang tujuan, SLI, pencapaian, dan interval, silakan lihat Konsep-konsep SLO.

- Nama SLO.
- Kolom Tujuan menampilkan persentase periode selama setiap interval yang harus berhasil memenuhi ambang batas SLI agar tujuan SLO terpenuhi. Ini juga menampilkan panjang interval untuk SLO tersebut.
- Status SLI akan menampilkan apakah status operasional aplikasi saat ini sedang dalam kondisi yang sehat atau tidak sehat. Jika ada periode selama rentang waktu yang dipilih saat ini tidak sehat untuk SLO, status SLI menampilkan Tidak Sehat.
- Pencapaian akhir adalah tingkat pencapaian yang dicapai pada akhir rentang waktu yang dipilih. Urutkan berdasarkan kolom ini untuk melihat SLO yang paling berisiko tidak terpenuhi.
- Delta pencapaian adalah perbedaan tingkat pencapaian antara awal dan akhir rentang waktu yang dipilih. Delta negatif berarti bahwa metrik kecenderungannya sedang ke arah bawah. Urutkan berdasarkan kolom ini untuk melihat kecenderungan-kecenderungan terbaru SLO.
- Anggaran kesalahan akhir (%) adalah persentase dari total waktu dalam periode yang dapat memiliki periode tidak sehat dan masih memiliki SLO yang berhasil dicapai. Jika Anda mengatur ini menjadi 5%, dan SLI sedang dalam kondisi tidak sehat dalam 5% atau kurang dari periode yang tersisa dalam interval, maka SLO masih berhasil dicapai.
- Delta anggaran kesalahan adalah perbedaan anggaran kesalahan antara awal dan akhir rentang waktu yang dipilih. Delta negatif berarti bahwa metrik sedang mengarah ke arah yang gagal.
- Anggaran kesalahan akhir (waktu) adalah jumlah waktu aktual dalam interval yang bisa tidak sehat dan masih memiliki SLO yang berhasil dicapai. Sebagai contoh, jika ini 14 menit, maka jika SLI tidak sehat selama kurang dari 14 menit selama interval yang tersisa, SLO akan tetap berhasil tercapai.
- Kolom Layanan, Operasi, dan Tipe menampilkan informasi tentang layanan dan operasi apa yang diatur SLO ini.

3. Untuk melihat grafik pencapaian dan kesalahan untuk SLO, pilih tombol radio di samping nama SLO.

   Grafik di bagian atas halaman menampilkan pencapaian SLO dan status Anggaran kesalahan. Sebuah grafik tentang metrik SLI yang dikaitkan dengan SLO ini juga ditampilkan.

4. Untuk melakukan penilaian awal lebih lanjut SLO yang tidak memenuhi tujuannya, pilih nama layanan atau nama operasi yang terkait dengan SLO tersebut. Anda dibawa ke halaman detail di mana Anda dapat melakukan penilaian awal lebih lanjut. Untuk informasi selengkapnya, lihat Menampilkan detail aktivitas layanan dan kesehatan operasional dengan halaman detail Layanan.
5. Untuk mengubah rentang waktu grafik dan tabel pada halaman tersebut, pilih rentang waktu baru di dekat bagian atas layar.

Sunting SLO yang ada

Ikuti langkah-langkah ini untuk menyunting SLO yang ada. Saat Anda menyunting SLO, Anda hanya dapat mengubah ambang batas, interval, tujuan pencapaian, dan tag. Untuk mengubah aspek lain seperti layanan, operasi, atau metrik, buat SLO baru alih-alih menyunting yang sudah ada.

Mengubah bagian dari konfigurasi inti SLO, seperti periode atau ambang batas, membatalkan semua titik data sebelumnya dan penilaian tentang pencapaian dan kondisi kesehatan. Ini secara efektif menghapus dan membuat kembali SLO.

> **Note**
>
> Jika Anda menyunting SLO, alarm yang terkait dengan SLO tersebut tidak diperbarui secara otomatis. Anda mungkin perlu memperbarui alarm-alarm tersebut agar tetap sinkron dengan SLO.

Cara menyunting SLO yang ada

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Tujuan Tingkat Layanan (SLO).
3. Pilih tombol radio yang ada di samping SLO yang ingin Anda sunting, dan pilih Tindakan, Sunting SLO.
4. Buat perubahan, lalu pilih Simpan perubahan.

Menghapus SLO

Ikuti langkah-langkah ini untuk menghapus SLO yang ada.

> **Note**
>
> Saat Anda menghapus sebuah SLO, alarm yang terkait dengan SLO tersebut tidak akan dihapus secara otomatis. Anda dapat menghapusnya sendiri.

Cara menghapus SLO

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Tujuan Tingkat Layanan (SLO).
3. Pilih tombol radio di samping SLO yang ingin Anda sunting, dan pilih Tindakan, Hapus SLO.
4. Pilih Konfirmasi.

Memantau kondisi kesehatan operasional aplikasi Anda dengan Sinyal Aplikasi

> **Sinyal Aplikasi ada di rilis Pratinjau. Jika Anda memiliki umpan balik mengenai fitur ini, Anda dapat menghubungi kami di app-signals-feedback@amazon.com.**

Gunakan Sinyal Aplikasi dalam konsol CloudWatch untuk memantau dan memecahkan masalah kondisi kesehatan operasional aplikasi Anda:

- Memantau layanan aplikasi Anda — Sebagai bagian dari pemantauan operasional harian, gunakan halaman Layanan untuk melihat ringkasan dari semua layanan Anda. Lihat layanan dengan tingkat kegagalan atau latensi tertinggi, dan lihat layanan mana yang memiliki indikator tingkat layanan (SLI) yang tidak sehat. Pilih layanan untuk membuka halaman Detail Layanan dan lihat metrik terperinci, operasi layanan, canary Synthetics, dan permintaan klien. Ini dapat membantu Anda memecahkan masalah dan mengidentifikasi akar penyebab masalah operasional.
- Periksa topologi aplikasi Anda — Gunakan Peta Layanan untuk memahami dan memantau topologi aplikasi Anda dari waktu ke waktu, termasuk hubungan antara klien, canary Synthetics, layanan, dan dependensi. Secara instan melihat kondisi kesehatan indikator tingkat layanan (SLI) dan melihat metrik-metrik utama seperti volume panggilan, tingkat kegagalan, dan latensi. Menelusuri untuk melihat informasi lebih detail di halaman Detail Layanan.

Telusuri contoh skenario yang menunjukkan bagaimana halaman ini dapat digunakan untuk menyelesaikan masalah-masalah kondisi kesehatan layanan operasional dengan cepat, mulai dari deteksi awal hingga mengidentifikasi akar penyebab.

Bagaimana Sinyal Aplikasi memungkinkan pemantauan kondisi kesehatan operasional

Setelah Anda mengaktifkan aplikasi untuk Sinyal Aplikasi, layanan aplikasi Anda, API, dan dependensinya secara otomatis ditemukan dan ditampilkan di halaman Layanan, detail Layanan, dan Peta Layanan. Sinyal Aplikasi mengumpulkan informasi dari berbagai sumber sehingga memungkinkan penemuan layanan dan pemantauan kondisi kesehatan operasional:

- AWS Distro for OpenTelemetry (ADOT) - Sebagai bagian dari mengaktifkan Sinyal Aplikasi, pustaka instrumentasi otomatis Java OpenTelemetry dikonfigurasi untuk memancarkan metrik dan jejak yang dikumpulkan oleh agen CloudWatch. Metrik-metrik dan jejak-jejak tersebut digunakan untuk memungkinkan penemuan layanan, operasi, dependensi, dan informasi layanan lainnya.
- Tujuan tingkat layanan (SLO) — Setelah Anda membuat sasaran tingkat layanan untuk layanan Anda, halaman Layanan, detail Layanan, dan Peta Layanan menampilkan kesehatan indikator tingkat layanan (SLI). SLI dapat memantau latensi, ketersediaan, dan metrik-metrik operasional lainnya.
- Canary CloudWatch Synthetics — Saat Anda mengonfigurasi pelacakan X-Ray pada canary, panggilan ke layanan Anda dari skrip canary dikaitkan dengan layanan Anda dan ditampilkan dalam halaman detail Layanan.
- Pemantauan pengguna nyata (RUM) CloudWatch - Ketika pelacakan X-Ray diaktifkan pada klien web RUM CloudWatch Anda, permintaan ke layanan Anda secara otomatis terkait dan ditampilkan dalam halaman detail layanan.
- AWS Service Catalog AppRegistry — Sinyal Aplikasi secara otomatis menemukan sumber daya AWS dalam akun Anda dan memungkinkan Anda untuk mengelompokkannya ke dalam aplikasi logis yang dibuat di AppRegistry. Nama aplikasi yang ditampilkan di halaman Layanan didasarkan pada sumber daya komputasi yang mendasari operasi layanan Anda.

> **Note**
>
> Sinyal Aplikasi akan menampilkan layanan dan operasi Anda berdasarkan metrik-metrik dan jejak-jejak yang dipancarkan dalam filter waktu saat ini yang Anda pilih. (Secara default, dalam tiga jam terakhir.) Jika tidak ada aktivitas dalam filter waktu saat ini untuk layanan, operasi, dependensi, canary Synthetics, atau halaman klien, itu tidak akan ditampilkan.

Saat ini, ia bisa menampilkan hingga 1.000 layanan. Penemuan layanan-layanan dan topologi layanan Anda mungkin tertunda hingga 10 menit. Evaluasi kondisi kesehatan indikator tingkat layanan (SLI) Anda mungkin tertunda hingga 15 menit.

Lihat keseluruhan aktivitas layanan dan kesehatan operasional dengan halaman Layanan

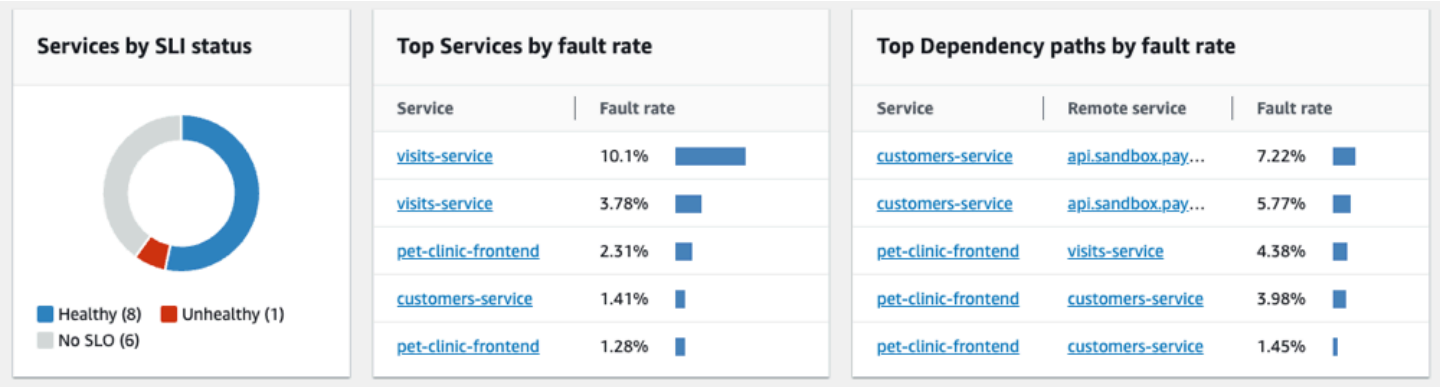
⚠ Sinyal Aplikasi berada dalam rilis pratinjau untuk Amazon CloudWatch dan dapat berubah sewaktu-waktu.

Gunakan halaman Layanan untuk melihat daftar layanan Anda yang diaktifkan untuk Sinyal Aplikasi. Anda juga dapat melihat metrik operasional dan dengan cepat melihat layanan mana yang memiliki indikator tingkat layanan (SLI) yang tidak sehat. Telusuri untuk mencari anomali performa saat Anda mengidentifikasi akar penyebab masalah operasional. Untuk melihat halaman ini, buka konsol CloudWatch dan pilih Layanan di bawah bagian Sinyal Aplikasi di panel navigasi kiri.

Jelajahi metrik kesehatan operasional untuk layanan Anda

Bagian atas halaman Layanan mencakup grafik kesehatan operasional layanan secara keseluruhan dan beberapa tabel yang menampilkan layanan teratas dan dependensi layanan berdasarkan tingkat kesalahan. Grafik Layanan di sebelah kiri menampilkan detail jumlah layanan yang memiliki indikator tingkat layanan (SLI) yang sehat atau tidak sehat selama filter waktu tingkat halaman saat ini. SLI dapat memantau latensi, ketersediaan, dan metrik-metrik operasional lainnya.

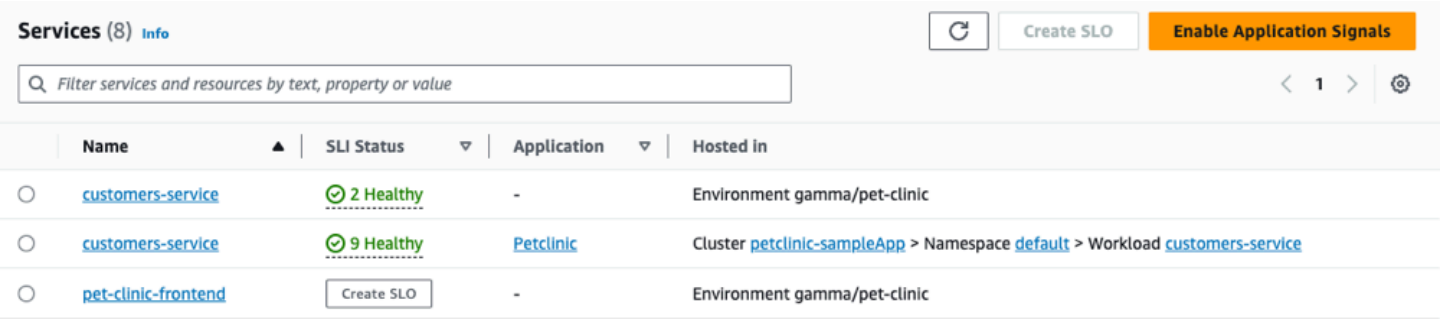
Dua tabel di sebelah grafik menampilkan daftar layanan teratas berdasarkan tingkat kesalahan. Pilih nama layanan apa pun di salah satu tabel untuk membuka halaman detail layanan dan melihat detail operasi layanan terperinci. Pilih jalur dependensi untuk membuka halaman detail dan melihat detail dependensi layanan. Kedua tabel menampilkan informasi hingga tiga jam terakhir, bahkan jika filter periode waktu yang lebih lama dipilih di kanan atas halaman.



Pantau kesehatan operasional dengan tabel Layanan

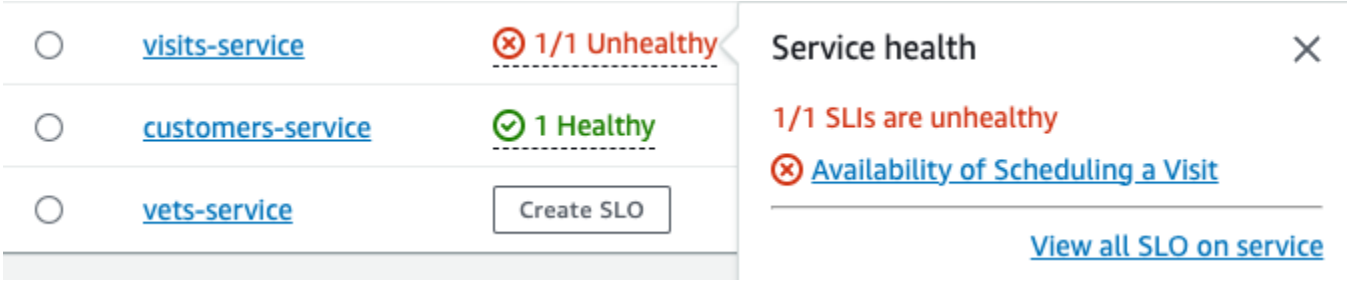
Tabel Layanan menampilkan daftar layanan Anda yang telah diaktifkan untuk Sinyal Aplikasi. Pilih Aktifkan Sinyal Aplikasi untuk membuka halaman penyiapan dan mulai mengkonfigurasi layanan Anda. Untuk informasi selengkapnya, silakan lihat Mengaktifkan Sinyal Aplikasi.

Filter tabel Layanan agar lebih mudah untuk menemukan apa yang Anda cari, dengan memilih satu atau beberapa properti dari kotak teks filter. Saat Anda memilih setiap properti, Anda dipandu melalui kriteria filter. Anda akan melihat filter lengkap di bawah kotak teks filter. Pilih Hapus filter kapan saja untuk menghapus filter tabel tersebut.



Pilih nama layanan apa pun dalam tabel untuk melihat halaman detail layanan yang berisi metrik tingkat layanan, operasi, dan detail tambahan. Jika Anda telah mengaitkan sumber daya komputasi dasar layanan dengan aplikasi di AppRegistry atau kartu Aplikasi di halaman beranda AWS Management Console, pilih nama aplikasi untuk menampilkan detail aplikasi di halaman konsol MyApplications. Untuk layanan yang di-host di Amazon EKS, pilih tautan apa pun di dalam kolom Di-host di untuk melihat Klaster, Namespace, atau Beban Kerja dalam Wawasan Kontainer CloudWatch. Untuk layanan yang berjalan di Amazon ECS atau Amazon EC2, nilai Lingkungan ditampilkan.

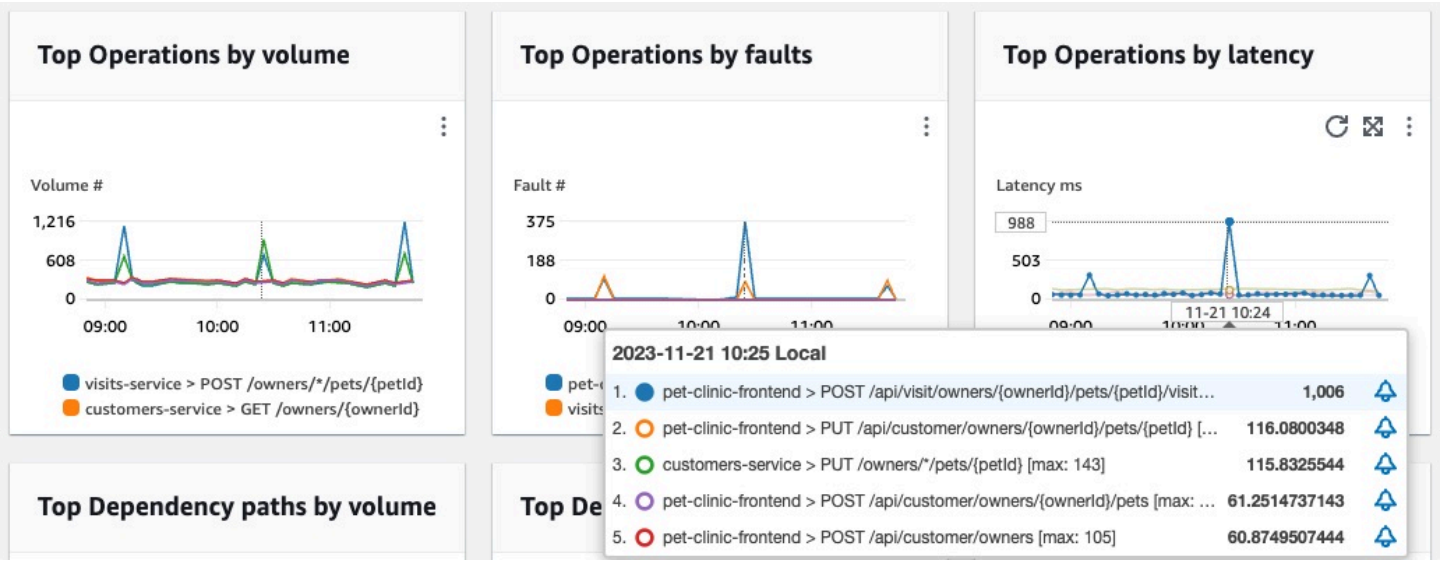
Status indikator tingkat layanan (SLI) ditampilkan untuk setiap layanan dalam tabel. Pilih status SLI agar layanan menampilkan pop-up yang berisi tautan ke SLI yang tidak sehat, dan tautan untuk melihat semua SLO untuk layanan tersebut.



Jika tidak ada SLO yang dibuat untuk layanan, pilih tombol Buat SLO dalam kolom Status SLI. Untuk membuat SLO tambahan untuk layanan apa pun, pilih tombol opsi di sebelah nama layanan, lalu pilih Buat SLO di kanan atas tabel. Saat Anda membuat SLO, Anda dapat melihat sekilas layanan dan operasi mana yang berperforma baik dan mana yang tidak sehat. Lihat tujuan tingkat layanan (SLO) untuk informasi lebih lanjut.

Lihat metrik operasi dan dependensi metrik teratas

Di bawah tabel Layanan, Anda dapat melihat operasi dan dependensi teratas di semua layanan berdasarkan volume panggilan, kesalahan, dan latensi. Kumpulan grafik ini memberi Anda informasi penting tentang operasi atau dependensi mana yang mungkin tidak sehat di semua layanan. Pilih titik mana pun dalam grafik untuk melihat pop-up yang berisi informasi seri yang lebih rinci. Arahkan kursor ke deskripsi seri di bagian bawah grafik untuk melihat pop-up yang berisi metrik terperinci untuk operasi atau jalur dependensi tertentu. Pilih tombol menu konteks di sudut kanan atas grafik untuk melihat opsi tambahan, termasuk melihat metrik CloudWatch atau halaman log.



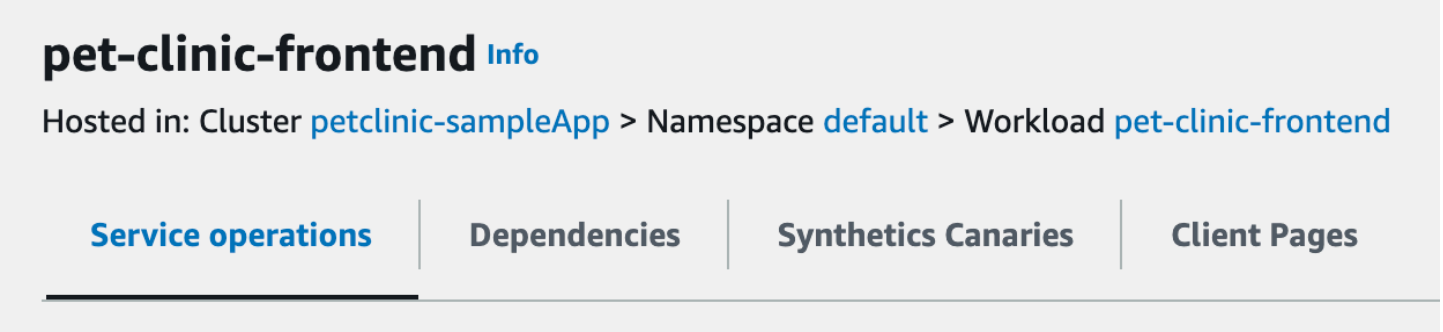
Menampilkan detail aktivitas layanan dan kesehatan operasional dengan halaman detail Layanan

⚠ Sinyal Aplikasi berada dalam rilis pratinjau untuk Amazon CloudWatch dan dapat berubah sewaktu-waktu.

Halaman detail Layanan menampilkan operasi, dependensi, canary, dan permintaan klien untuk satu layanan yang telah diaktifkan untuk Sinyal Aplikasi. Untuk melihat halaman ini, buka konsol CloudWatch, pilih Layanan di bawah bagian Sinyal Aplikasi di panel navigasi kiri, dan pilih nama layanan apa pun dari tabel Layanan atau Layanan teratas atau tabel dependensi.

Halaman detail Layanan disusun menjadi empat tab:

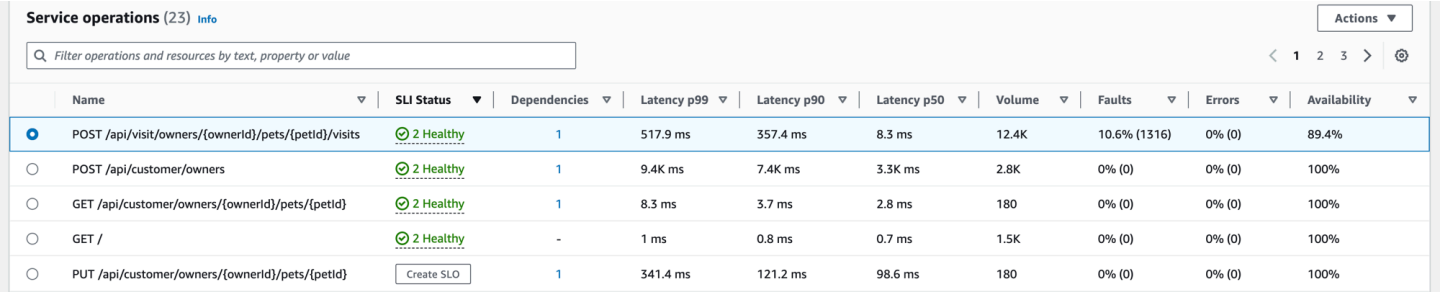
- Operasi layanan — Tab ini menampilkan daftar operasi yang ditampilkan layanan Anda, bersama dengan metrik utama untuk setiap operasi.
- Dependensi — Tab ini menampilkan daftar dependensi yang dipanggil oleh layanan Anda, dan daftar metrik dependensi.
- Canary Synthetics — Tab ini menampilkan daftar canary Synthetics yang memanggil layanan Anda, bersama dengan metrik utama untuk eksekusi canary.
- Halaman klien — Tab ini menampilkan daftar halaman klien yang memanggil layanan Anda, bersama dengan metrik halaman klien.



Untuk layanan yang di-host di Amazon EKS, pilih tautan apa pun di bawah nama layanan untuk melihat Klaster, Namespace, atau Beban Kerja dalam Wawasan Kontainer CloudWatch. Untuk layanan-layanan yang di-host di Amazon ECS atau Amazon EC2, nilai Lingkungan ditampilkan.

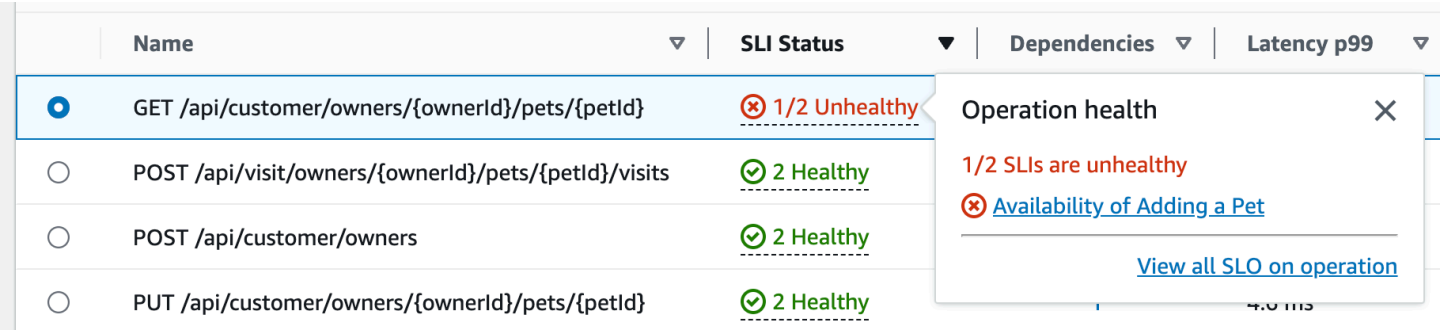
Menampilkan operasi layanan Anda

Pilih tab Operasi layanan untuk menampilkan tabel Operasi layanan, dan satu set metrik untuk operasi yang dipilih. Tabel tersebut berisi daftar operasi yang ditemukan oleh Sinyal Aplikasi, termasuk status indikator tingkat layanan (SLI), jumlah dependensi, dan metrik untuk latensi, volume, kesalahan, kekeliruan, dan ketersediaan.



Lakukan penyaringan pada tabel untuk membuatnya lebih mudah untuk menemukan apa yang Anda cari, dengan memilih satu atau beberapa sifat dari kotak teks filter. Saat Anda memilih setiap sifat, Anda dipandu melalui kriteria filter dan akan melihat filter lengkap di bawah kotak teks filter. Pilih Hapus filter kapan saja untuk menghapus filter tabel tersebut.

Status indikator tingkat layanan (SLI) ditampilkan untuk setiap operasi dalam tabel, termasuk jumlah SLI yang sehat atau tidak sehat dan jumlah total tujuan tingkat layanan (SLO) yang telah Anda buat. SLI dapat memantau latensi, ketersediaan, dan metrik operasional lainnya untuk memastikan kualitas layanan. Pilih status SLI untuk operasi untuk menampilkan pop-up yang berisi tautan ke SLI yang tidak sehat, dan tautan untuk melihat semua SLO untuk operasi.



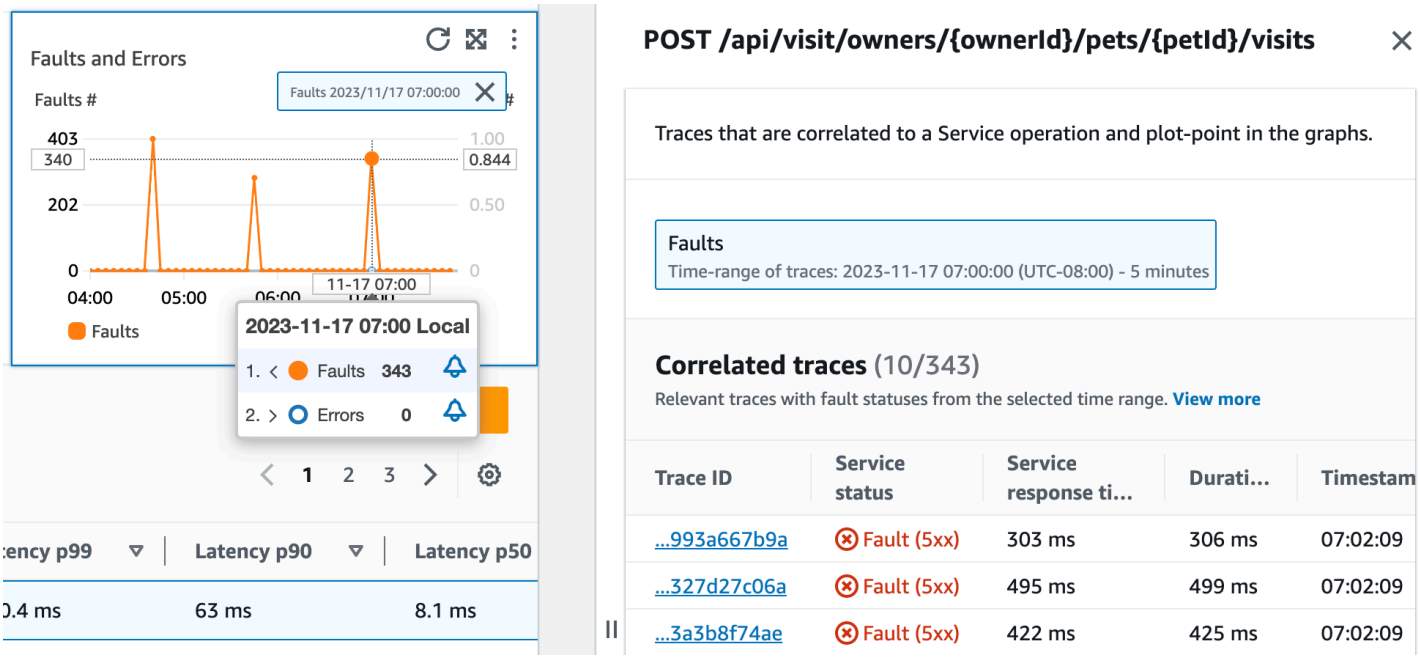
Jika tidak ada SLO yang dibuat untuk operasi, pilih tombol Buat SLO di dalam kolom Status SLI. Untuk membuat SLO tambahan untuk operasi apa pun, pilih tombol radio di sebelah nama operasi, lalu pilih Buat SLO dari dropdown Tindakan di kanan atas tabel. Saat Anda membuat SLO, Anda dapat melihat sekilas layanan dan operasi mana yang berperforma baik dan mana yang tidak sehat. Untuk informasi selengkapnya, silakan lihat Tujuan tingkat layanan (SLO).

Kolom Dependensi menunjukkan jumlah dependensi yang dipanggil oleh operasi ini. Pilih nomor ini untuk membuka tab Dependensi yang difilter ke operasi yang dipilih.

Melihat metrik operasi layanan dan jejak yang berkorelasi

Sinyal Aplikasi menghubungkan metrik operasi layanan dengan jejak AWS X-Ray, sehingga lebih mudah untuk memecahkan masalah kesehatan operasional. Pilih opsi di samping layanan dalam tabel Operasi layanan untuk melihat satu set grafik di atas tabel dengan metrik untuk latensi, volume panggilan, kesalahan, kekeliruan, dan ketersediaan. Arahkan kursor ke titik dalam grafik untuk melihat pop-up yang berisi informasi lebih lanjut.

Pilih titik untuk membuka laci diagnostik yang menunjukkan jejak yang berkorelasi untuk titik yang dipilih dalam grafik tersebut. Pilih ID jejak dari tabel jejak yang Berkorelasi untuk membuka halaman detail Jejak X-Ray untuk jejak yang dipilih.

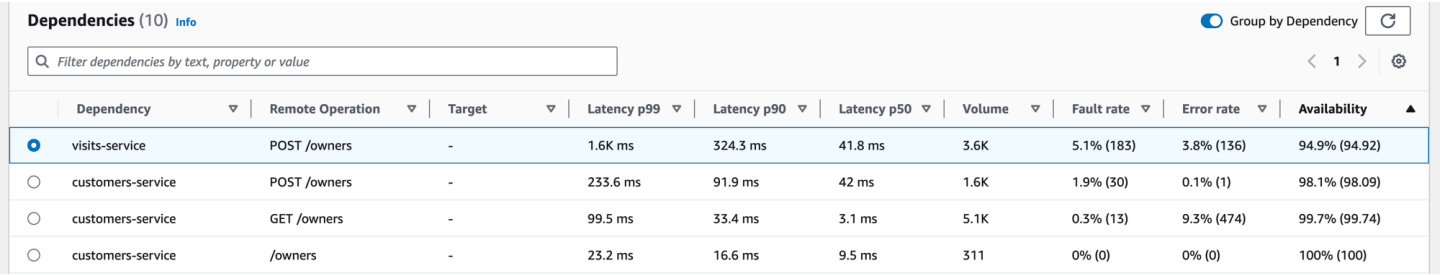


Lihat dependensi layanan Anda

Pilih tab Dependensi untuk menampilkan tabel Dependensi dan satu set metrik untuk dependensi semua operasi layanan atau operasi tunggal. Tabel tersebut berisi daftar dependensi yang ditemukan oleh Sinyal Aplikasi, termasuk metrik untuk latensi, volume panggilan, tingkat kesalahan, tingkat kekeliruan, dan ketersediaan.

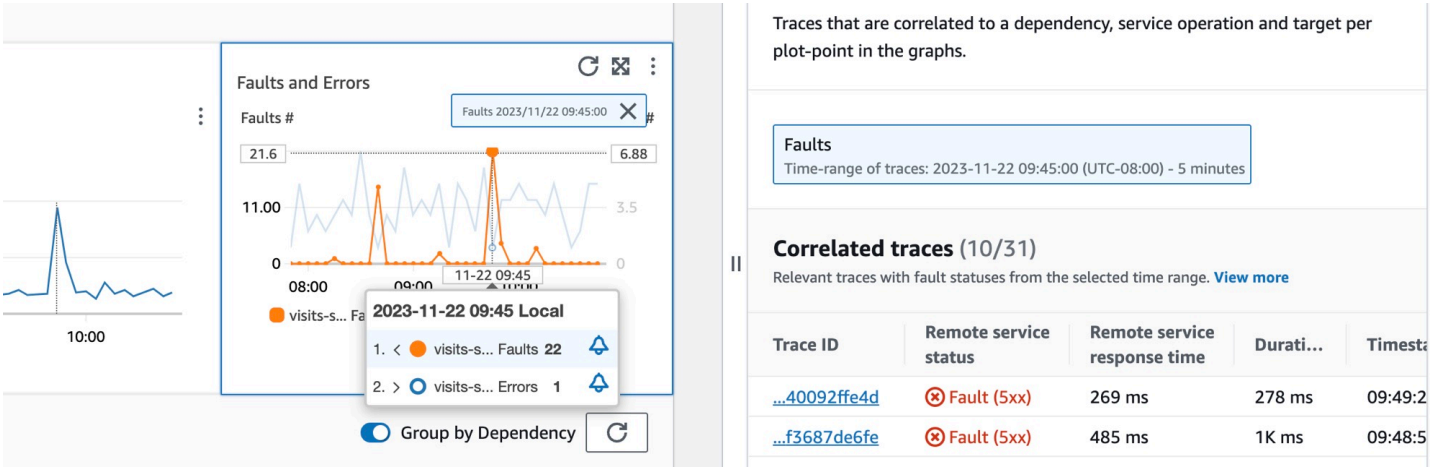
Di bagian atas halaman, pilih operasi dari dropdown untuk melihat dependensinya, atau pilih Semua untuk melihat dependensi untuk semua operasi.

Lakukan penyaringan pada tabel untuk membuatnya lebih mudah untuk menemukan apa yang Anda cari, dengan memilih satu atau beberapa sifat dari kotak teks filter. Saat Anda memilih setiap sifat, Anda dipandu melalui kriteria filter dan akan melihat filter lengkap di bawah kotak teks filter. Pilih Hapus filter kapan saja untuk menghapus filter tabel tersebut. Pilih Grup berdasarkan Dependens di kanan atas tabel untuk mengelompokkan dependensi berdasarkan nama layanan dan operasi. Saat pengelompokan diaktifkan, perluas atau ciutkan satu grup dependensi dengan ikon + di sebelah nama dependensi tersebut.



Kolom Dependensi menampilkan nama layanan dependensi, sedangkan kolom Operasi Jarak Jauh menampilkan nama operasi layanan. Saat memanggil layananAWS, kolom Target menampilkan AWS sumber daya, seperti tabel DynamoDB atau antrean Amazon SNS.

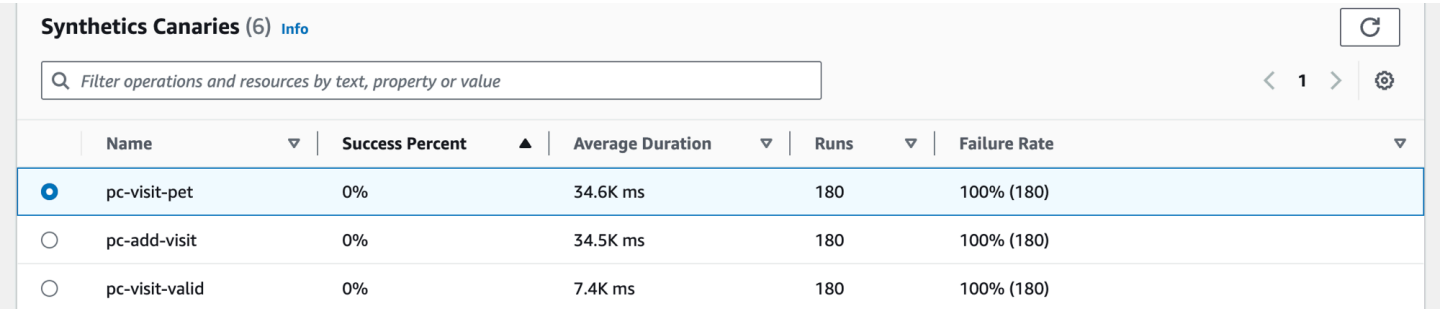
Untuk memilih dependensi, pilih opsi di sebelah dependensi dalam tabel Dependensi. Anda akan melihat satu set grafik yang menampilkan metrik terperinci untuk volume panggilan, ketersediaan, kesalahan, dan kekeliruan. Arahkan kursor ke titik dalam grafik untuk melihat pop-up yang berisi informasi lebih lanjut. Pilih titik dalam grafik untuk membuka laci diagnostik yang menunjukkan jejak yang berkorelasi untuk titik yang dipilih dalam grafik tersebut. Pilih ID jejak dari tabel jejak yang Berkorelasi untuk membuka halaman detail jejak X-Ray untuk jejak yang dipilih.



Lihat canary Synthetics Anda

Pilih tab Canary Synthetics untuk menampilkan tabel Canary Synthetics, dan satu set metrik untuk setiap canary dalam tabel. Tabel ini mencakup metrik untuk persentase keberhasilan, durasi rata-rata, proses, dan tingkat kegagalan. Hanya canary yang diaktifkan untuk pelacakan AWS X-Ray yang ditampilkan.

Lakukan penyaringan pada tabel untuk membuatnya lebih mudah untuk menemukan apa yang Anda cari, dengan memilih satu atau beberapa sifat dari kotak teks filter. Saat Anda memilih setiap sifat, Anda dipandu melalui kriteria filter, dan akan melihat filter lengkap di bawah kotak teks filter. Pilih Hapus filter kapan saja untuk menghapus filter tabel tersebut.



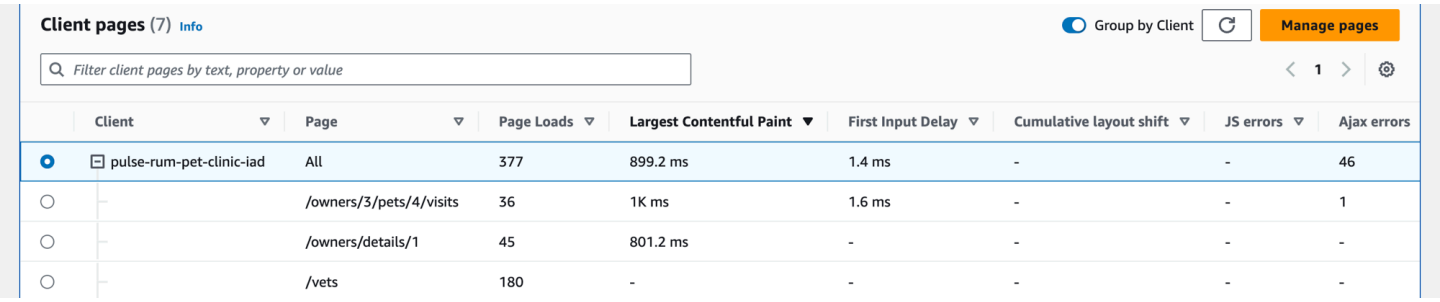
Pilih opsi di sebelah canary di tabel untuk memilih canary dan lihat satu set grafik yang menampilkan persentase dan durasi keberhasilan metrik terperinci. Arahkan kursor ke titik dalam grafik untuk melihat pop-up yang berisi informasi lebih lanjut. Pilih titik dalam grafik untuk membuka laci diagnostik yang menunjukkan run canary yang berkorelasi untuk titik yang dipilih dalam grafik. Untuk membuka halaman Canary Synthetics CloudWatch, pilih Waktu run untuk sebuah run canary.

Lihat halaman klien Anda

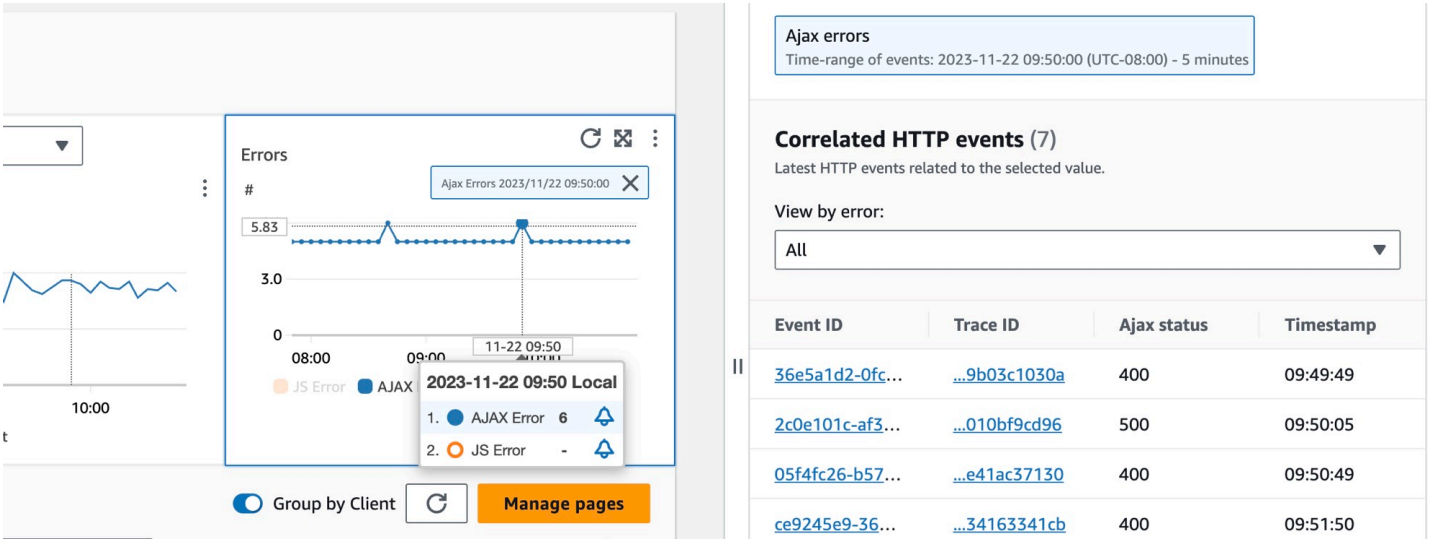
Pilih tab Halaman klien untuk menampilkan tabel halaman Klien dan satu set metrik untuk halaman klien yang dipilih, termasuk pemuatan halaman, vital web, dan kesalahan. Tabel berisi daftar halaman klien yang memanggil layanan Anda.

Untuk menampilkan halaman klien Anda dalam tabel, konfigurasikan klien web RUM CloudWatch Anda untuk pelacakan X-Ray dan aktifkan metrik Sinyal Aplikasi untuk halaman klien Anda. Pilih Kelola halaman di kanan atas tabel untuk mengelola halaman mana yang diaktifkan untuk metrik Sinyal Aplikasi.

Lakukan penyaringan pada tabel halaman Klien agar lebih mudah untuk menemukan apa yang Anda cari, dengan memilih satu atau beberapa sifat dari kotak teks filter. Saat Anda memilih setiap sifat, Anda dipandu melalui kriteria filter dan akan melihat filter lengkap di bawah kotak teks filter. Pilih Hapus filter kapan saja untuk menghapus filter tabel tersebut. Pilih Grup berdasarkan Klien untuk mengelompokkan halaman klien berdasarkan klien. Saat dikelompokkan, pilih ikon + di sebelah nama klien untuk memperluas baris dan melihat semua halaman untuk klien itu.



Untuk memilih halaman klien, pilih opsi di sebelah halaman klien di tabel halaman Klien. Anda akan melihat satu set grafik yang menampilkan metrik terperinci. Arahkan kursor ke titik dalam grafik untuk melihat pop-up yang berisi informasi lebih lanjut. Pilih titik dalam grafik untuk membuka laci diagnostik yang menunjukkan peristiwa navigasi performa yang berkorelasi untuk titik yang dipilih dalam grafik tersebut. Pilih ID peristiwa dari daftar peristiwa navigasi untuk membuka tampilan Tampilan halaman RUM CloudWatch untuk peristiwa yang dipilih.



> **Note**
>
> Untuk melihat kesalahan AJAX di halaman klien Anda, gunakan klien web CloudWatch RUM versi 1.15 atau yang lebih baru.
> Saat ini, hingga 100 operasi, canary, dan halaman klien, dan hingga 250 dependensi, dapat ditampilkan per layanan.

Lihat topologi aplikasi Anda dan pantau kesehatan operasional dengan Peta Layanan CloudWatch

> Sinyal Aplikasi berada dalam rilis pratinjau untuk Amazon CloudWatch dan dapat berubah sewaktu-waktu.

> **Note**
>
> Peta Layanan CloudWatch menggantikan peta ServiceLens. Untuk melihat peta aplikasi Anda berdasarkan jejak AWS X-Ray, buka Peta Jejak X-Ray. Pilih Peta Jejak di bawah bagian X-Ray di panel navigasi kiri konsol CloudWatch.

Gunakan Peta Layanan untuk melihat topologi klien aplikasi Anda, canary Synthetics, layanan dan dependensi, dan memantau kesehatan operasional. Gali lebih dalam untuk melihat informasi yang lebih rinci. Untuk melihat Peta Layanan, buka konsol CloudWatch dan pilih Peta Layanan di bawah bagian Sinyal Aplikasi di panel navigasi kiri.

Setelah Anda mengaktifkan aplikasi untuk Sinyal Aplikasi, gunakan Peta Layanan untuk memudahkan pemantauan kesehatan operasional aplikasi Anda:

- Lihat koneksi antara klien, canary, layanan, dan simpul dependensi untuk membantu Anda memahami topologi aplikasi dan alur eksekusi Anda. Ini sangat membantu jika operator layanan Anda bukan tim pengembangan Anda.
- Lihat layanan mana yang memenuhi atau tidak memenuhi tujuan tingkat layanan (SLO) Anda. Ketika layanan tidak memenuhi SLO Anda, Anda dapat dengan cepat mengidentifikasi apakah

layanan hilir atau dependensi mungkin berkontribusi terhadap masalah atau memengaruhi beberapa layanan hulu.

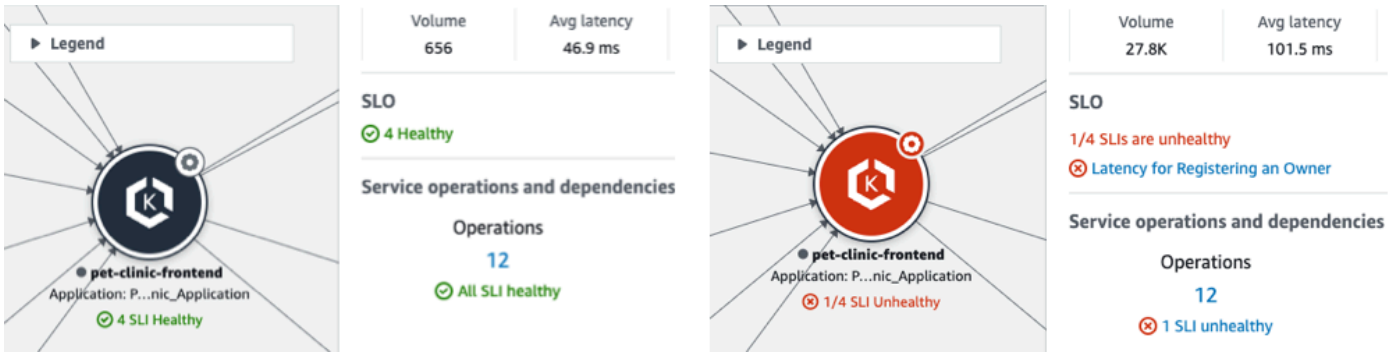
- Pilih klien individual, canary Synthetics, layanan, atau simpul dependensi untuk melihat metrik terkait. Telusuri halaman detail Layanan untuk melihat informasi lebih rinci tentang operasi, dependensi, canary Synthetics, dan halaman klien.
- Filter dan perbesar Peta Layanan untuk mempermudah fokus pada bagian topologi aplikasi Anda, atau melihat seluruh peta. Buat filter dengan memilih satu atau beberapa properti dari kotak teks filter. Saat Anda memilih setiap properti, Anda dipandu melalui kriteria filter. Anda akan melihat filter lengkap di bawah kotak teks filter. Pilih Hapus filter kapan saja untuk menghapus filter tersebut.

Menjelajahi Peta Layanan

Setelah Anda mengaktifkan aplikasi untuk Sinyal Aplikasi, Peta Layanan menampilkan simpul yang mewakili layanan dan dependensi Anda. Aktifkan pelacakan aktif untuk klien RUM CloudWatch dan canary Synthetics Anda untuk melihat simpul klien dan canary di peta. Pilih tab berikut untuk informasi tentang menjelajahi setiap jenis simpul dan edge (koneksi) di antara mereka.

View your application services

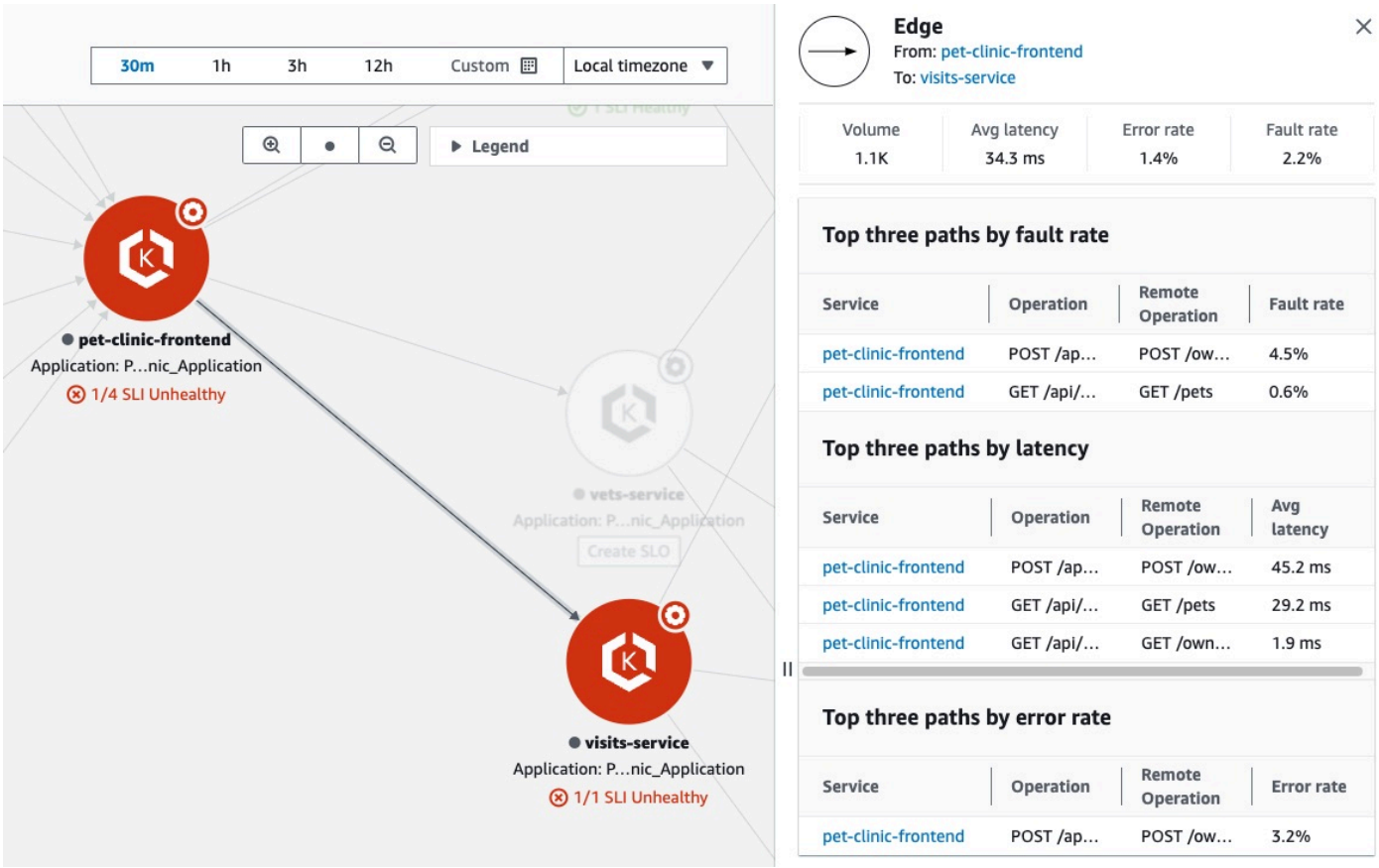
Layanan Anda ditampilkan di Peta Layanan, terhubung ke klien dan canary yang menggunakannya, dan dependensi yang dipanggil oleh layanan Anda. Status indikator tingkat layanan (SLI) Anda saat ini ditampilkan di bawah setiap simpul layanan. Jika suatu layanan memiliki satu atau beberapa SLI yang tidak sehat, simpul layanan ditampilkan dengan warna merah dengan jumlah SLI yang tidak sehat ditampilkan di bawah simpul tersebut. Jika tidak ada SLO yang dibuat untuk layanan, pilih tombol Buat SLO di bawah simpul layanan.



Ketika Anda memilih simpul layanan, laci terbuka menampilkan informasi layanan terperinci:

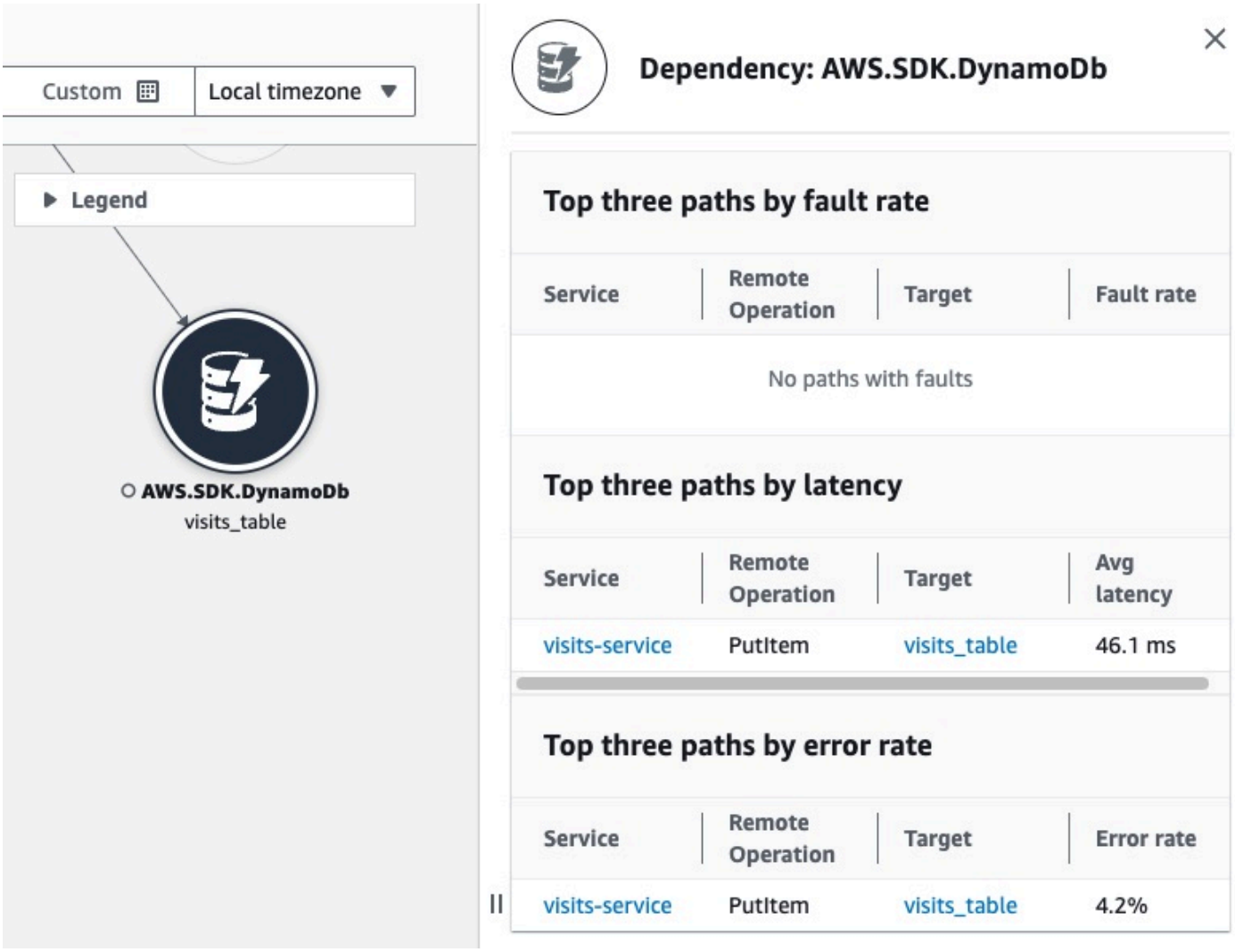
- Metrik untuk volume panggilan, latensi, kekeliruan, dan tingkat kesalahan.
- Jumlah SLI yang sehat atau tidak sehat untuk layanan ini. Pilih SLO untuk melihat informasi lebih lanjut.
- Jumlah operasi layanan, dependensi, canary Synthetics, dan halaman klien. Pilih setiap nomor untuk membuka halaman Detail layanan dan telusuri untuk melihat informasi yang lebih rinci.
- Nama aplikasi, jika Anda telah mengaitkan sumber daya komputasi yang mendasarinya dengan aplikasi yang menggunakan AppRegistry atau kartu Aplikasi di halaman beranda AWS Management Console. Pilih nama aplikasi untuk menampilkan detail aplikasi di halaman konsol MyApplications.
- Klaster, Namespace, dan Beban Kerja untuk layanan yang di-host di Amazon EKS, atau Lingkungan untuk layanan yang di-host di Amazon ECS atau Amazon EC2. Untuk layanan yang di-host Amazon EKS, pilih tautan apa pun untuk membuka Wawasan Kontainer CloudWatch.

Pilih edge (koneksi) antara simpul layanan dan layanan hilir atau simpul dependensi. Ini membuka laci yang berisi jalur teratas berdasarkan tingkat kesalahan, latensi, dan tingkat kekeliruan. Pilih tautan apa pun di laci untuk membuka halaman Detail layanan dan melihat informasi rinci untuk layanan atau dependensi yang dipilih.



View dependencies

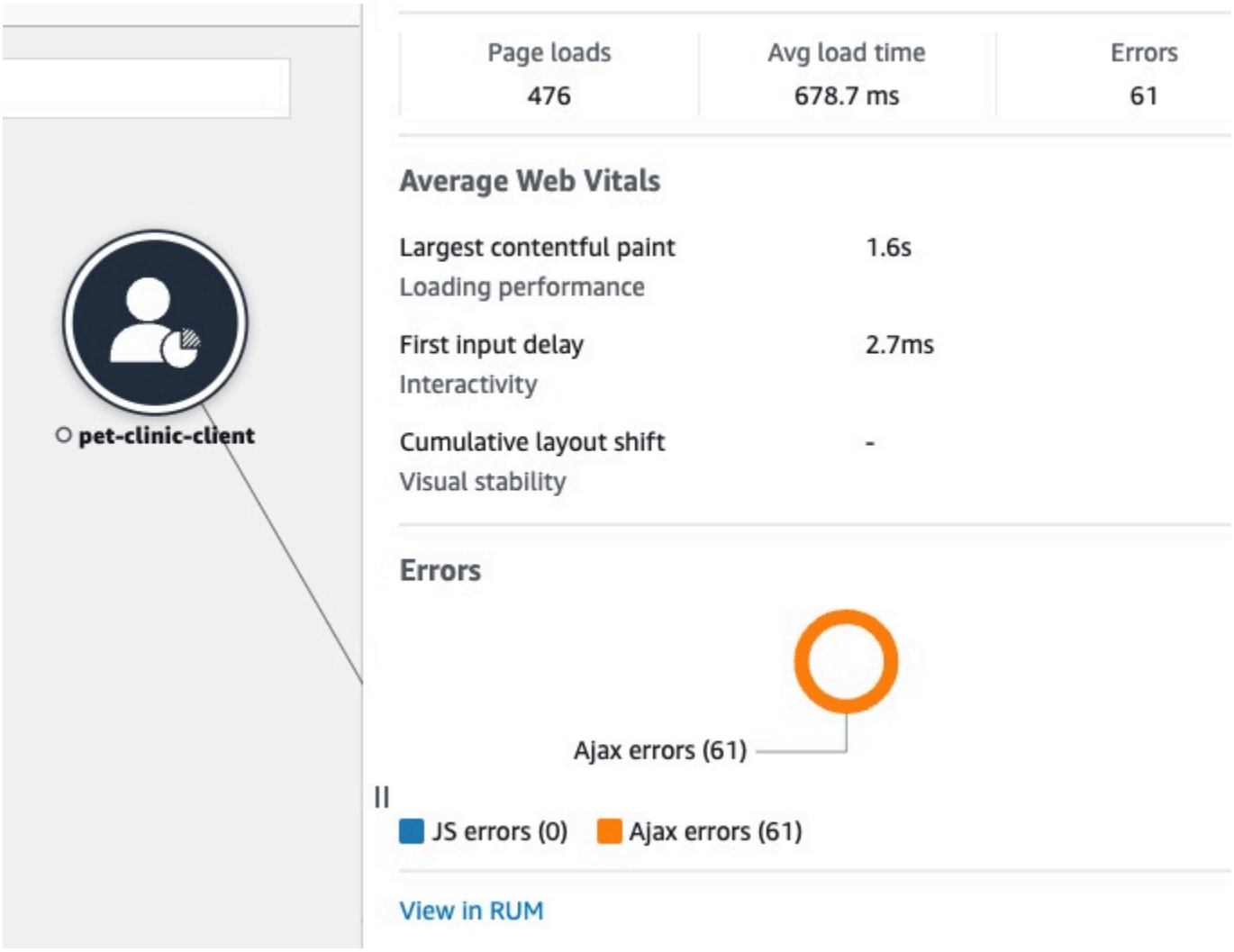
Dependensi aplikasi Anda ditampilkan di Peta Layanan, terhubung ke layanan yang memanggilnya. Pilih simpul dependensi untuk membuka laci yang berisi jalur teratas berdasarkan tingkat kesalahan, latensi, dan tingkat kekeliruan. Pilih layanan atau tautan target apa pun untuk membuka halaman Detail Layanan dan melihat informasi terperinci tentang target layanan atau dependensi yang dipilih.



View clients

Setelah Anda mengaktifkan pelacakan X-Ray untuk klien web RUM CloudWatch Anda, mereka ditampilkan di Peta Layanan yang terhubung ke layanan yang mereka panggil. Pilih simpul klien untuk membuka laci yang menampilkan informasi klien terperinci:

- Metrik untuk pemuatan halaman, waktu muat rata-rata, kesalahan, dan vital web rata-rata.
- Grafik yang menampilkan detail kesalahan.
- Tautan untuk menampilkan detail klien di RUM CloudWatch.



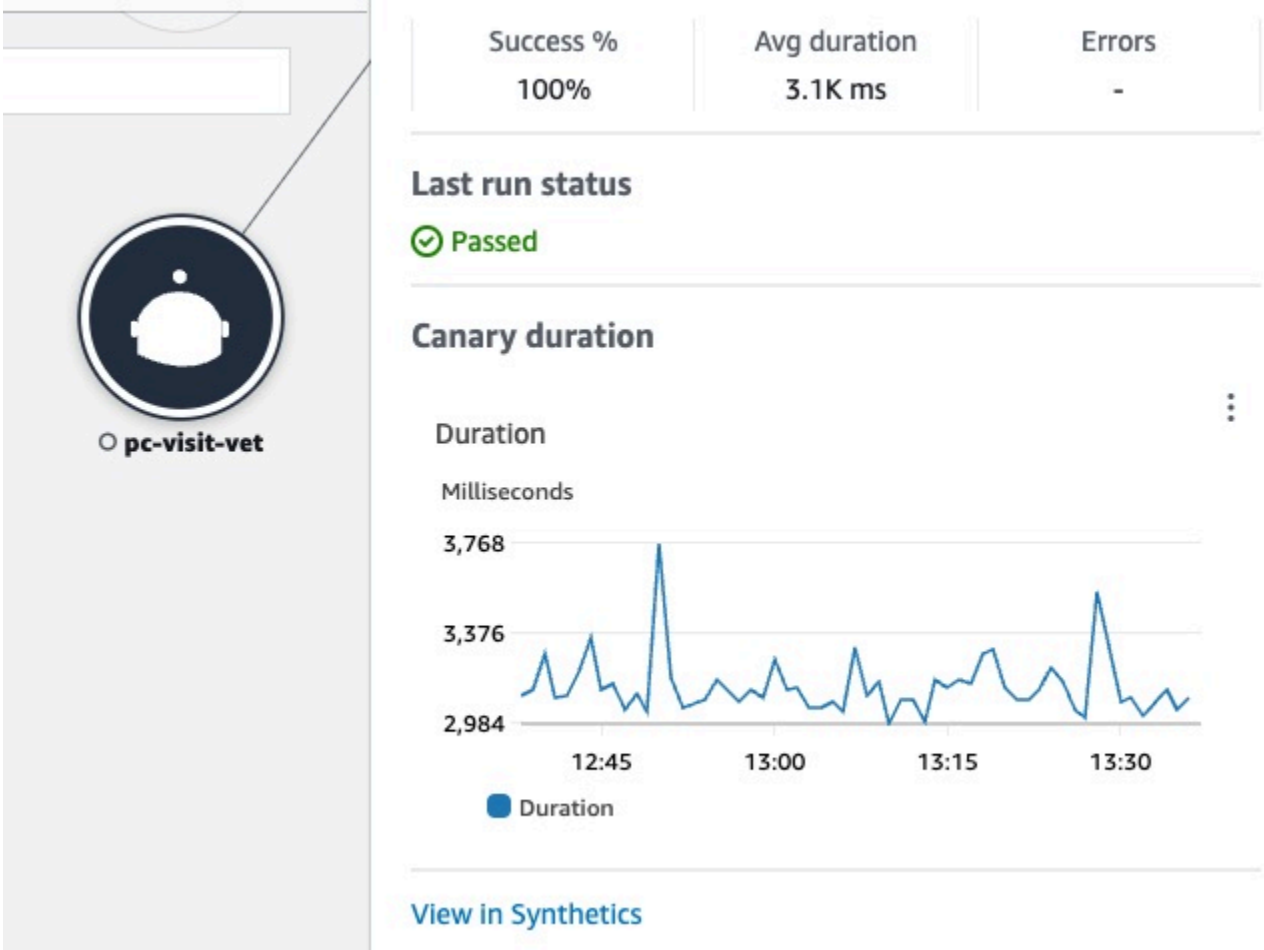
> **Note**
>
> Untuk melihat kesalahan AJAX di halaman klien Anda, gunakan klien web CloudWatch RUM versi 1.15 atau yang lebih baru.

View Synthetics canaries

Setelah Anda mengaktifkan pelacakan AWS X-Ray untuk canary CloudWatch Synthetics Anda, mereka akan ditampilkan di Peta Layanan yang terhubung ke layanan yang mereka panggil. Pilih simpul canary untuk membuka laci yang menampilkan informasi canary terperinci:

- Metrik untuk persentase keberhasilan, durasi rata-rata, dan kesalahan.
- Status run canary terakhir.

- Grafik yang menampilkan durasi run canary. Namun melalui rangkaian grafik untuk melihat pop-up yang berisi informasi lebih lanjut.
- Tautan untuk menampilkan detail canary di CloudWatch Synthetics.



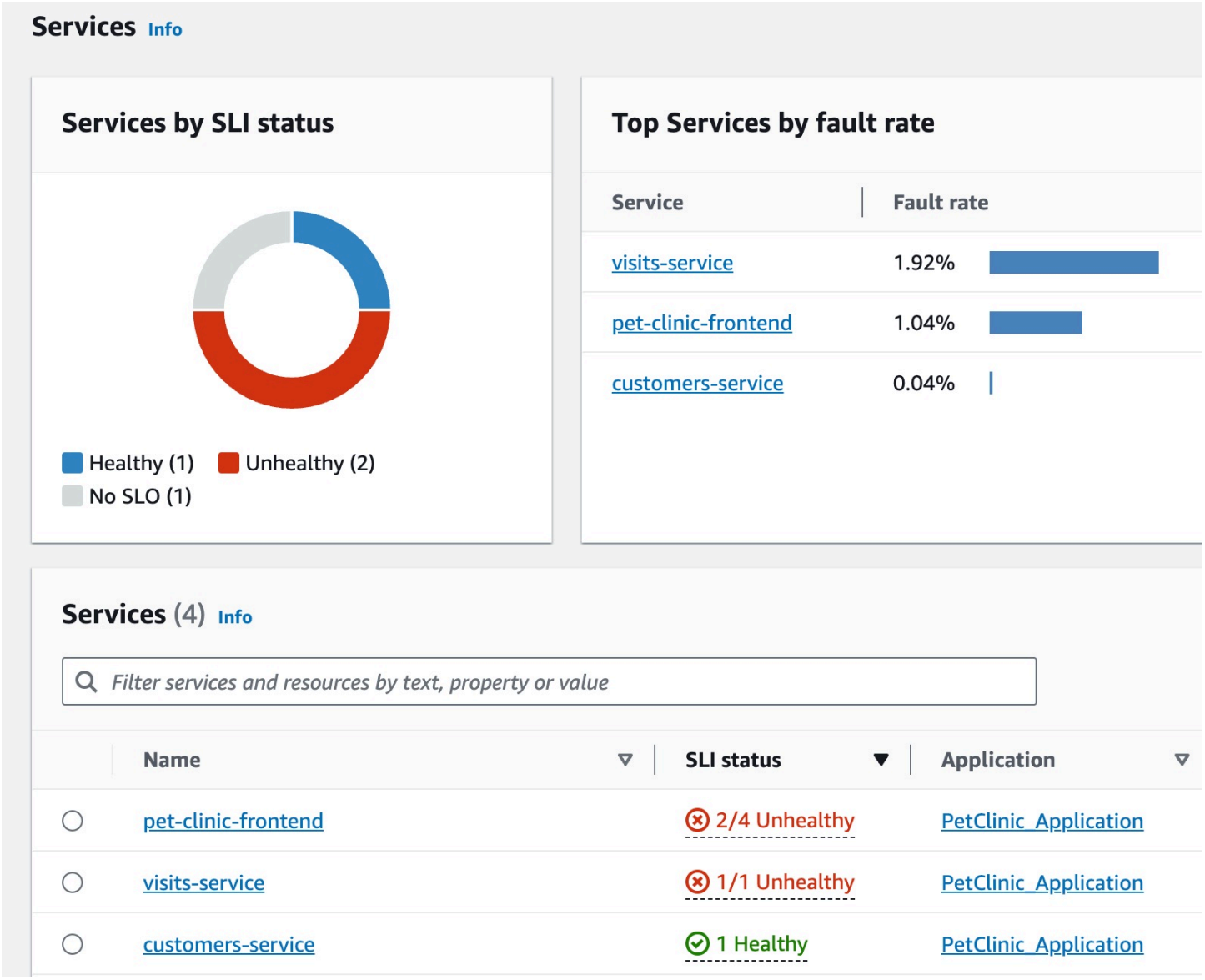
Contoh: Gunakan Sinyal Aplikasi untuk menyelesaikan masalah kesehatan operasional

> ⚠ Sinyal Aplikasi berada dalam rilis pratinjau untuk Amazon CloudWatch dan dapat berubah sewaktu-waktu.

Skenario berikut memberikan contoh cara Sinyal Aplikasi dapat digunakan untuk memantau layanan Anda dan mengidentifikasi masalah kualitas layanan. Telusuri untuk mengidentifikasi akar penyebab

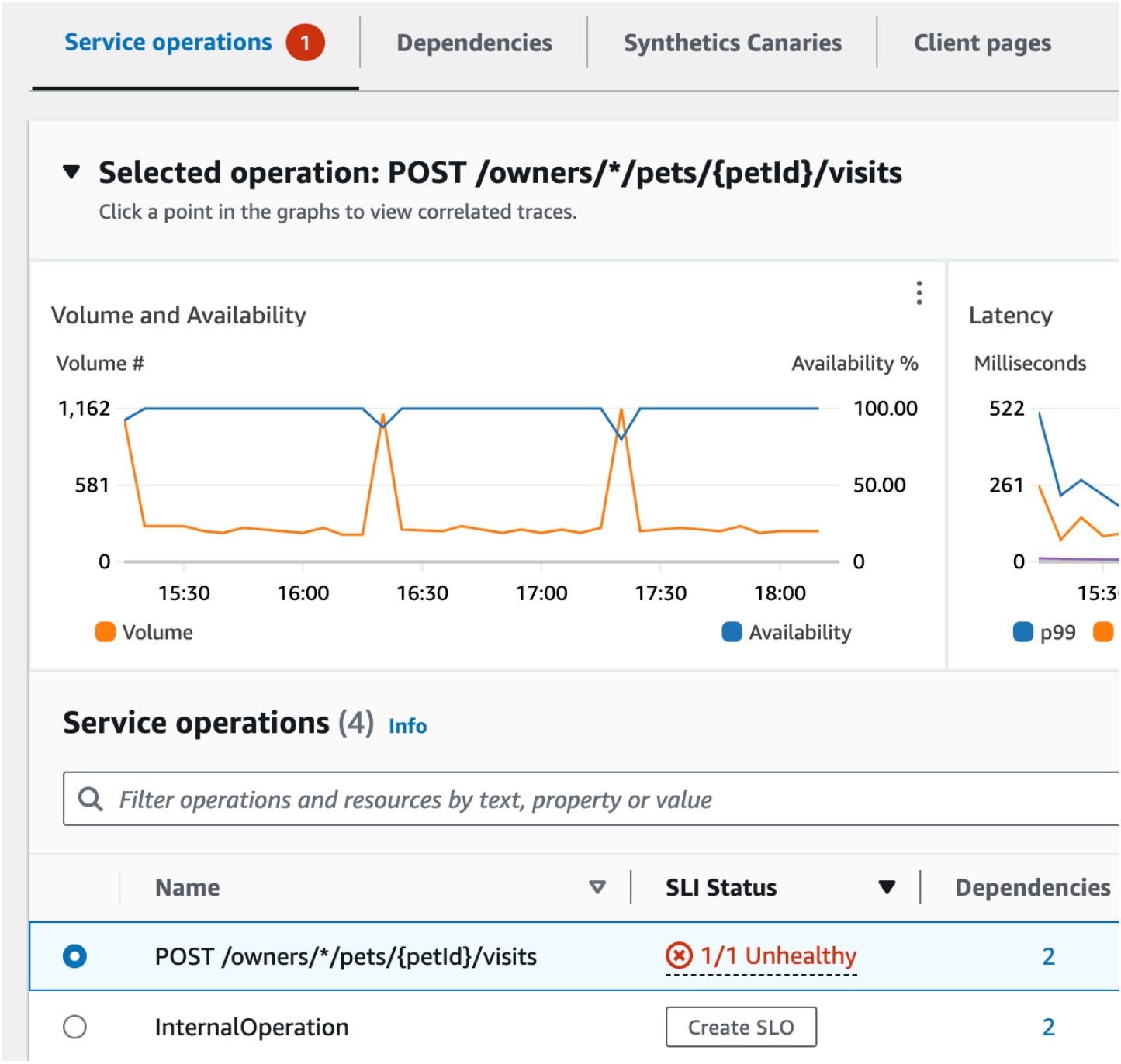
potensial dan mengambil tindakan untuk menyelesaikan masalah. Contoh ini difokuskan pada aplikasi klinik hewan peliharaan yang terdiri dari beberapa layanan mikro yang memanggil Layanan AWS seperti DynamoDB.

Jane adalah bagian dari tim DevOps yang mengawasi kesehatan operasional aplikasi klinik hewan peliharaan. Tim Jane berkomitmen untuk memastikan bahwa aplikasi ini sangat tersedia dan responsif. Mereka menggunakan tujuan tingkat layanan (SLO) untuk mengukur performa aplikasi terhadap komitmen bisnis ini. Dia menerima peringatan tentang beberapa indikator tingkat layanan (SLI) yang tidak sehat. Dia membuka konsol CloudWatch dan menavigasi ke halaman Layanan, di mana dia melihat beberapa layanan dalam keadaan tidak sehat.

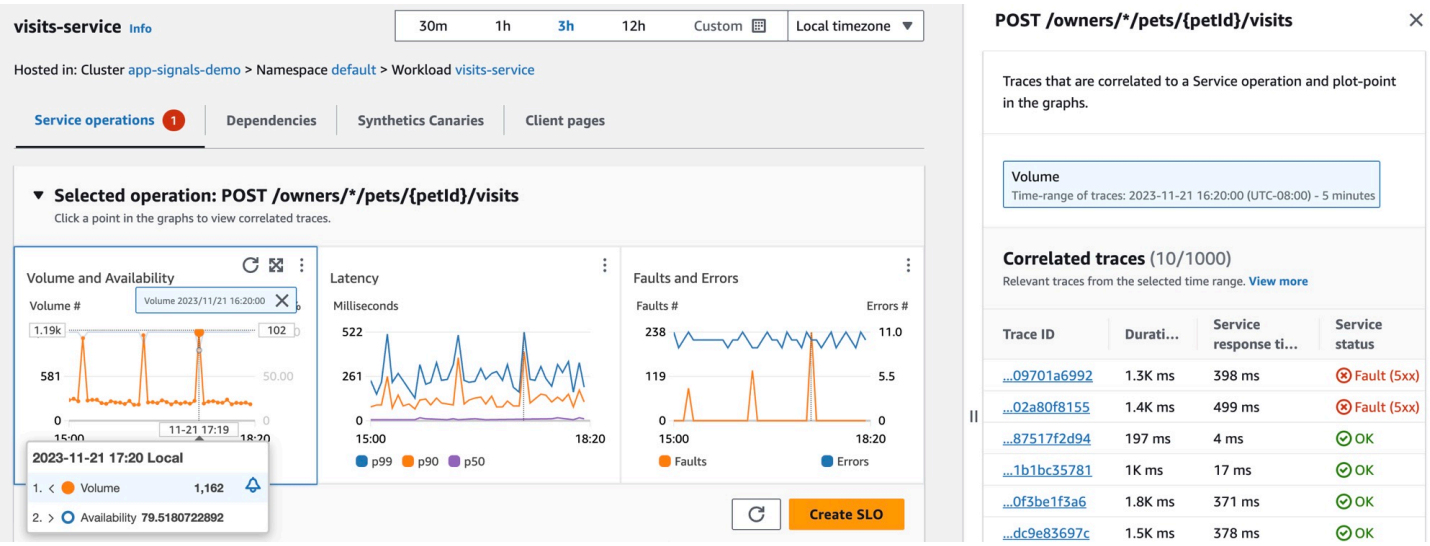


Di bagian atas halaman, Jane melihat bahwa `visits-service` merupakan layanan teratas berdasarkan tingkat kesalahan. Dia memilih tautan dalam grafik, yang membuka halaman detail

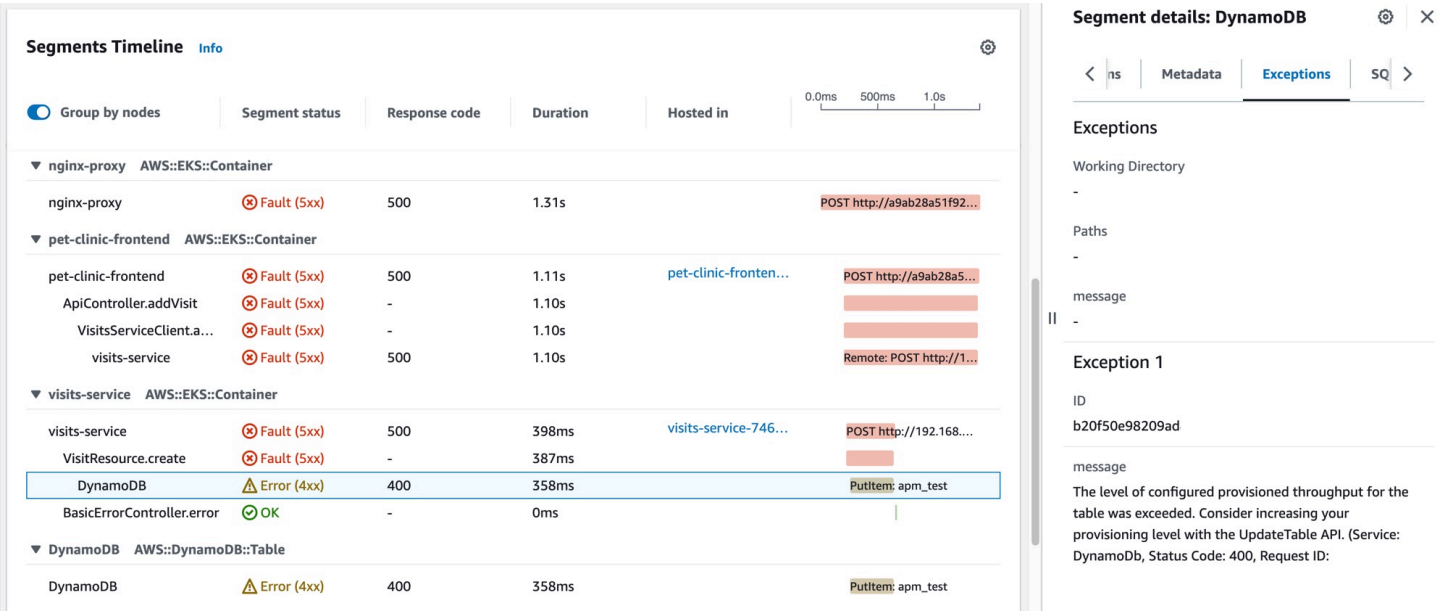
Layanan untuk layanan tersebut. Dia melihat bahwa ada operasi yang tidak sehat di tabel operasi Layanan. Dia memilih operasi ini dan melihat dalam grafik Volume dan Ketersediaan bahwa ada lonjakan volume panggilan berkala yang tampaknya berkorelasi dengan penurunan ketersediaan.



Untuk melihat lebih dekat penurunan ketersediaan layanan, Jane memilih salah satu titik data ketersediaan dalam grafik. Laci terbuka yang menunjukkan jejak X-Ray yang berkorelasi dengan titik data yang dipilih. Dia melihat bahwa ada beberapa jejak yang memuat kesalahan.



Jane memilih salah satu jejak yang berkorelasi dengan status kesalahan, yang membuka halaman detail jejak X-Ray untuk jejak yang dipilih. Jane menggulir ke bawah ke bagian Timeline Segments dan mengikuti jalur panggilan sampai dia melihat bahwa panggilan ke tabel DynamoDB mengembalikan kesalahan. Dia memilih segmen DynamoDB dan menavigasi ke tab Exceptions pada laci sisi kanan.



Jane melihat bahwa sumber daya DynamoDB salah dikonfigurasi, yang mengakibatkan kesalahan selama lonjakan permintaan pelanggan. Tingkat throughput yang disediakan tabel DynamoDB secara berkala terlampaui, yang mengakibatkan masalah ketersediaan layanan dan SLI yang tidak sehat. Berdasarkan informasi ini, timnya dapat mengonfigurasi tingkat throughput yang disediakan yang lebih tinggi dan memastikan ketersediaan aplikasi yang tinggi.

Metrik aplikasi standar yang dikumpulkan

⚠ Sinyal Aplikasi ada di rilis Pratinjau. Jika Anda memiliki umpan balik mengenai fitur ini, Anda dapat menghubungi kami di app-signals-feedback@amazon.com.

Sinyal Aplikasi mengumpulkan metrik aplikasi standar dari layanan-layanan yang ditemukannya. Metrik tersebut berhubungan dengan aspek paling penting dari performa layanan: latensi, gangguan, dan kesalahan. Metrik tersebut dapat membantu Anda mengidentifikasi masalah, memantau tren performa, dan mengoptimalkan sumber daya untuk meningkatkan pengalaman pengguna secara keseluruhan.

Tabel berikut mencantumkan metrik-metrik yang dikumpulkan oleh Sinyal Aplikasi. Metrik ini dikirim ke CloudWatch di tempatnama `AppSignals`.

Metrik	Deskripsi
`Latency`	Penundaan sebelum transfer data akan dimulai setelah permintaan dibuat. Satuan: Milidetik
`Faults`	Hitungan baik gangguan pada server HTTP 5XX maupun kesalahan status rentang TelemetryTerbuka. Satuan: Tidak ada
`Errors`	Hitungan kesalahan pada klien HTTP 4XX. Kesalahan ini dianggap sebagai kesalahan permintaan yang tidak disebabkan oleh masalah layanan. Oleh karena itu, metrik `Availability` yang ditampilkan pada dasbor Sinyal Aplikasi tidak menganggap kesalahan ini sebagai gangguan layanan. Satuan: Tidak ada

Metrik `Availability` yang ditampilkan di dasbor Sinyal Aplikasi dihitung sebagai (1 - **`Faults`** / Berhasil) *100. Respons yang berhasil adalah semua tanggapan tanpa gangguan `5XX`. Tanggapan `4XX` diperlakukan sebagai berhasil ketika Sinyal Aplikasi menghitung `Availability`.

Dimensi-dimensi yang dikumpulkan dan kombinasi dimensi

Dimensi berikut didefinisikan untuk tiap-tiap metrik aplikasi standar. Untuk informasi selengkapnya tentang dimensi, silakan lihat Dimensi.

Dimensi yang berbeda dikumpulkan untuk metrik layanan dan metrik dependensi. Dalam layanan yang ditemukan oleh Sinyal Aplikasi, ketika microservice A memanggil microservice B, microservice B melayani permintaan. Dalam hal ini, microservice A memancarkan metrik dependensi dan microservice B memancarkan metrik layanan. Ketika klien memanggil layanan mikro A, maka layanan mikro A akan melayani permintaan dan memancarkan metrik layanan.

Dimensi untuk metrik layanan

Dimensi berikut dikumpulkan untuk metrik layanan.

Dimensi	Deskripsi
`Service`	Nama layanan.
`Operation`	Nama operasi API atau aktivitas lainnya.
`HostedIn.EKS.Cluster`	Nama klaster Amazon EKS tempat layanan berjalan. Dimensi ini dikumpulkan hanya jika layanan-layanan itu berjalan di Amazon EKS.
`HostedIn.K8s.Namespace`	Nama dari ruang nama Kubernetes tempat layanan berjalan. Dimensi ini dikumpulkan hanya jika layanan-layanan itu berjalan di Amazon EKS.
`HostedIn.Environment`	Nama lingkungan yang ditentukan pengguna tempat di mana layanan berjalan. Dimensi ini dikumpulkan hanya jika layanan-layanan itu berjalan di sebuah lingkungan yang bukan Amazon EKS.

Ketika melihat metrik ini di konsol CloudWatch, Anda dapat memilih melihatnya dengan kombinasi dimensi berikut.

- `Service`, `Operation`, `HostedIn.EKS.Cluster`, `HostedIn.K8s.Namespace`
- `Service`, `HostedIn.EKS.Cluster`, `HostedIn.K8s.Namespace`

Untuk platform yang bukan Amazon EKS, Anda juga dapat melihat metrik-metrik layanan tersebut dengan kombinasi dimensi berikut.

- `Service`, `Operation`, `HostedIn.Environment`
- `Service`, `HostedIn.Environment`

Dimensi untuk metrik dependensi

Dimensi berikut dikumpulkan untuk metrik dependensi.

Dimensi	Deskripsi
`Service`	Nama layanan.
`Operation`	Nama operasi API atau aktivitas lainnya.
`RemoteService`	Nama layanan jarak jauh yang diminta.
`RemoteOperation`	Nama operasi API yang diminta.
`HostedIn.EKS.Cluster`	Nama klaster Amazon EKS tempat layanan berjalan. Dimensi ini dikumpulkan hanya jika layanan-layanan itu berjalan di Amazon EKS.
`HostedIn.K8s.Namespace`	Nama dari ruang nama Kubernetes tempat layanan berjalan. Dimensi ini dikumpulkan hanya jika layanan-layanan itu berjalan di Amazon EKS.
`K8s.RemoteNamespace`	Nama dari ruang nama Kubernetes tempat layanan dependensi berjalan.

Dimensi	Deskripsi
	Dimensi ini dikumpulkan hanya jika layanan-layanan itu berjalan di Amazon EKS.
`RemoteTarget`	Nama sumber daya yang diminta panggilan jarak jauh. Dimensi ini tidak memiliki nilai jika panggilan jarak jauh tidak diarahkan ke sumber daya tertentu. Dimensi ini dikumpulkan hanya jika layanan-layanan itu berjalan di Amazon EKS.
`HostedIn.Environment`	Nama lingkungan yang ditentukan pengguna tempat di mana layanan berjalan. Dimensi ini dikumpulkan hanya jika layanan-layanan itu berjalan di sebuah lingkungan yang bukan Amazon EKS.

Ketika melihat metrik ini di konsol CloudWatch, Anda dapat memilih melihatnya dengan kombinasi dimensi berikut.

Berjalan di platform apa pun

- `RemoteService`

Berjalan di klaster Amazon EKS

- `Service`, `Operation`, `HostedIn.EKS.Cluster`, `HostedIn.K8s.Namespace`, `RemoteService`, `RemoteOperation`, `K8s.RemoteNamespace`, `RemoteTarget`
- `Service`, `Operation`, `HostedIn.EKS.Cluster`, `HostedIn.K8s.Namespace`, `RemoteService`, `RemoteOperation`, `K8s.RemoteNamespace`
- `Service`, `Operation`, `HostedIn.EKS.Cluster`, `HostedIn.K8s.Namespace`, `RemoteService`, `RemoteOperation`, `RemoteTarget`
- `Service`, `Operation`, `HostedIn.EKS.Cluster`, `RemoteService`, `RemoteOperation`,
- `Service`, `HostedIn.EKS.Cluster`, `HostedIn.K8s.Namespace`, `RemoteService`, `K8s.RemoteNamespace`
- `Service`, `HostedIn.EKS.Cluster`, `RemoteService`, `K8s.RemoteNamespace`

- `Service`, `HostedIn.EKS.Cluster`, `HostedIn.K8s.Namespace`, `RemoteService`, `RemoteOperation`, `K8s.RemoteNamespace`, `RemoteTarget`
- `Service`, `HostedIn.EKS.Cluster`, `HostedIn.K8s.Namespace`, `RemoteService`, `RemoteOperation`, `K8s.RemoteNamespace`
- `Service`, `HostedIn.EKS.Cluster`, `HostedIn.K8s.Namespace`, `RemoteService`, `RemoteOperation`, `RemoteTarget`
- `Service`, `HostedIn.EKS.Cluster`, `HostedIn.K8s.Namespace`, `RemoteService`, `RemoteOperation`

Berjalan di platform selain klaster Amazon EKS

- `Service`, `Operation`, `HostedIn.Environment`
- `Service`, `HostedIn.Environment`
- `Service`, `Operation`, `HostedIn.Environment`, `RemoteService`, `RemoteOperation`, `RemoteTarget`
- `Service`, `Operation`, `HostedIn.Environment`, `RemoteService`, `RemoteOperation`,
- `Service`, `HostedIn.Environment`, `RemoteService`
- `Service`, `HostedIn.Environment`, `RemoteService`, `RemoteOperation`, `RemoteTarget`
- `Service`, `HostedIn.Environment`, `RemoteService`, `RemoteOperation`,

Menggunakan pemantauan sintetis

Anda dapat menggunakan Amazon CloudWatch Synthetics untuk membuat canary, skrip yang dapat dikonfigurasi yang berjalan sesuai jadwal, untuk memantau titik akhir dan API Anda. Canary mengikuti rute yang sama dan melakukan tindakan yang sama sebagai pelanggan, yang memungkinkan bagi Anda untuk terus memverifikasi pengalaman pelanggan bahkan ketika Anda tidak memiliki lalu lintas pelanggan pada aplikasi Anda. Dengan menggunakan canary, Anda dapat menemukan masalah sebelum para pelanggan Anda menemukannya.

Canary adalah skrip yang ditulis dalam Node.js atau Python. Mereka membuat fungsi Lambda di akun Anda yang menggunakan Node.js atau Python sebagai kerangka kerja. Canary bekerja di atas protokol HTTP maupun HTTPS. Canary menggunakan lapisan Lambda yang berisi pustaka CloudWatch Synthetics. Pustaka berisi versi NodeJS dari CloudWatch Synthetics untuk kenari NodeJS dan versi Python dari Synthetics untuk kenari Python. CloudWatch Lapisan milik akun

layanan CloudWatch Synthetics. Perpustakaan tidak pernah mengirimkan atau menyimpan informasi pelanggan. Semua data pelanggan disimpan di akun pelanggan saja.

Canary menawarkan akses terprogram ke Browser Google Chrome tanpa tampilan antarmuka grafis melalui Puppeteer atau Selenium Webdriver. Untuk informasi selengkapnya tentang Puppeteer, silakan lihat Puppeteer. Untuk informasi selengkapnya tentang Selenium, silakan lihat www.selenium.dev/.

Canary memeriksa ketersediaan dan latensi titik akhir Anda dan dapat menyimpan data waktu pemuatan dan tangkapan layar UI. Mereka memantau API REST, URL, dan konten situs web, serta mereka dapat memeriksa perubahan tidak sah akibat pengelabuan, injeksi kode, dan pembuatan skrip lintas situs.

CloudWatch Synthetics terintegrasi dengan Application Signals, yang dapat menemukan dan memantau layanan aplikasi, klien, kenari Synthetics, dan dependensi layanan Anda. Gunakan Sinyal Aplikasi untuk melihat daftar atau peta visual layanan Anda, melihat metrik kesehatan berdasarkan tujuan tingkat layanan (SLO) Anda, dan menelusuri lebih dalam untuk melihat jejak X-Ray yang berkorelasi untuk pemecahan masalah yang lebih rinci. Untuk melihat canary Anda di Sinyal Aplikasi, aktifkan pelacakan aktif X-Ray. Canary Anda ditampilkan di Service Map yang terhubung ke layanan Anda, dan di halaman detail Layanan dari layanan yang mereka panggil.

Untuk menonton video demonstrasi tentang canary, silakan lihat tayangan berikut ini:

- Pengantar Amazon CloudWatch Synthetics
- Amazon CloudWatch Synthetics Demo
- Buat Canary Menggunakan Amazon CloudWatch Synthetics
- Pemantauan Visual dengan Amazon CloudWatch Synthetics

Anda dapat mengoperasikan canary satu kali atau dengan jadwal reguler. Canary dapat berjalan dengan frekuensi sekali per menit. Anda dapat menggunakan ekspresi cron dan ekspresi rate untuk membuat jadwal canary.

Untuk mendapatkan informasi tentang masalah-masalah keamanan yang perlu Anda pertimbangkan sebelum membuat dan mengoperasikan canary, silakan lihat Pertimbangan keamanan untuk canary Synthetics.

Secara default, kenari membuat beberapa CloudWatch metrik di namespace. `CloudWatchSynthetics` Metrik ini memiliki `CanaryName` sebagai dimensi. Canary yang

menggunakan fungsi `executeStep()` atau `executeHttpStep()` dari pustaka fungsi juga memiliki `StepName` sebagai dimensi. Untuk informasi selengkapnya tentang pustaka fungsi canary, silakan lihat Fungsi pustaka tersedia untuk skrip canary.

CloudWatch Synthetics terintegrasi dengan baik dengan X-Ray Trace Map, yang digunakan CloudWatch AWS X-Ray untuk memberikan end-to-end tampilan layanan Anda untuk membantu Anda lebih efisien menentukan kemacetan kinerja dan mengidentifikasi pengguna yang terkena dampak. Canary yang Anda buat dengan CloudWatch Synthetics muncul di peta jejak. Untuk informasi selengkapnya, silakan lihat X-Ray Trace Map.

CloudWatch Synthetics saat ini tersedia di semua AWS Wilayah komersial dan Wilayah. GovCloud

> **Note**
>
> Di Asia Pasifik (Osaka), AWS PrivateLink tidak didukung. Di Asia Pasifik (Jakarta), AWS PrivateLink dan X-Ray tidak didukung.

Topik

Peran dan izin yang diperlukan untuk CloudWatch canary

Baik pengguna yang membuat maupun yang mengelola canary, dan canary itu sendiri, harus memiliki izin tertentu.

Peran dan izin yang diperlukan untuk pengguna yang mengelola kenari CloudWatch

Untuk melihat detail canary dan hasil operasi canary, Anda harus masuk sebagai pengguna dengan kebijakan `CloudWatchSyntheticsFullAccess` atau `CloudWatchSyntheticsReadOnlyAccess` yang dilampirkan. Untuk membaca semua data Synthetics di konsol, Anda juga memerlukan kebijakan `AmazonS3ReadOnlyAccess` dan `CloudWatchReadOnlyAccess`. Untuk melihat kode sumber yang digunakan oleh canary, Anda juga memerlukan kebijakan `AWSLambda_ReadOnlyAccess`.

Untuk membuat canary, Anda harus masuk sebagai pengguna yang memiliki kebijakan `CloudWatchSyntheticsFullAccess` atau serangkaian izin serupa. Untuk membuat peran IAM untuk canary, Anda juga memerlukan pernyataan kebijakan selaras berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateRole",
                "iam:CreatePolicy",
                "iam:AttachRolePolicy"
            ],
            "Resource": [
                "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*",
                "arn:aws:iam::*:policy/service-role/CloudWatchSyntheticsPolicy*"
            ]
        }
    ]
}
```

Important

Memberikan penggunaan `iam:CreateRole`, `iam:CreatePolicy`, dan `iam:AttachRolePolicy` izin memberi pengguna akses administratif penuh ke akun Anda AWS . Misalnya, pengguna dengan izin ini dapat membuat kebijakan yang memiliki izin

penuh untuk semua sumber daya dan dapat melampirkan kebijakan tersebut ke peran apa pun. Berhati-hatilah dengan orang yang Anda berikan izin ini.

Untuk informasi tentang cara melampirkan kebijakan dan memberikan izin kepada pengguna, silakan lihat Mengubah Izin untuk Pengguna IAM dan Cara menyematkan kebijakan yang selaras bagi pengguna atau peran.

Peran dan izin yang diperlukan untuk canary

Setiap canary harus dikaitkan dengan peran IAM yang memiliki izin tertentu yang dilampirkan. Saat Anda membuat kenari menggunakan CloudWatch konsol, Anda dapat memilih CloudWatch Synthetics untuk membuat peran IAM untuk kenari. Jika Anda melakukannya, peran tersebut akan memiliki izin yang diperlukan.

Jika Anda ingin membuat peran IAM sendiri, atau membuat peran IAM yang dapat Anda gunakan saat menggunakan AWS CLI atau API untuk membuat canary, peran tersebut harus berisi izin yang tercantum di bagian ini.

Semua peran IAM untuk canary harus menyertakan pernyataan kebijakan kepercayaan berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "lambda.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Selain itu, peran IAM canary membutuhkan salah satu pernyataan berikut.

Kenari dasar yang tidak menggunakan AWS KMS atau membutuhkan akses VPC Amazon

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::path/to/your/s3/bucket/canary/results/folder"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::name/of/the/s3/bucket/that/contains/canary/results"

            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup"
            ],
            "Resource": [
                "arn:aws:logs:canary_region_name:canary_account_id:log-group:/aws/
lambda/cwsyn-canary_name-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "xray:PutTraceSegments"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": "*",
```

```
            "Action": "cloudwatch:PutMetricData",
            "Condition": {
               "StringEquals": {
                "cloudwatch:namespace": "CloudWatchSynthetics"
               }
            }
        }
    ]
}
```

Canary yang digunakan AWS KMS untuk mengenkripsi artefak kenari tetapi tidak memerlukan akses VPC Amazon

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::path/to/your/S3/bucket/canary/results/folder"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::name/of/the/S3/bucket/that/contains/canary/results"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup"
            ],
            "Resource": [
```

```
                "arn:aws:logs:canary_region_name:canary_account_id:log-group:/aws/
lambda/cwsyn-canary_name-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "xray:PutTraceSegments"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": "*",
            "Action": "cloudwatch:PutMetricData",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": "CloudWatchSynthetics"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource":
 "arn:aws:kms:KMS_key_region_name:KMS_key_account_id:key/KMS_key_id",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": [
                        "s3.region_name_of_the_canary_results_S3_bucket.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Canary yang tidak menggunakan AWS KMS tetapi membutuhkan akses VPC Amazon

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::path/to/your/S3/bucket/canary/results/folder"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::name/of/the/S3/bucket/that/contains/canary/results"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup"
            ],
            "Resource": [
                "arn:aws:logs:canary_region_name:canary_account_id:log-group:/aws/
lambda/cwsyn-canary_name-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "xray:PutTraceSegments"
            ],
            "Resource": [
                "*"
            ]
```

```
        },
        {
            "Effect": "Allow",
            "Resource": "*",
            "Action": "cloudwatch:PutMetricData",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": "CloudWatchSynthetics"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DeleteNetworkInterface"
            ],
            "Resource": [
                "*"
            ]
       }
    ]
}
```

Canary yang digunakan AWS KMS untuk mengenkripsi artefak kenari dan juga membutuhkan akses VPC Amazon

Jika Anda memperbarui canary non-VPC untuk mulai menggunakan VPC, Anda harus memperbarui peran canary tersebut untuk menyertakan izin antarmuka jaringan yang tercantum dalam kebijakan berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::path/to/your/S3/bucket/canary/results/folder"
            ]
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::name/of/the/S3/bucket/that/contains/canary/results"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup"
            ],
            "Resource": [
                "arn:aws:logs:canary_region_name:canary_account_id:log-group:/aws/
lambda/cwsyn-canary_name-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "xray:PutTraceSegments"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": "*",
            "Action": "cloudwatch:PutMetricData",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": "CloudWatchSynthetics"
                }
            }
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DeleteNetworkInterface"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource":
 "arn:aws:kms:KMS_key_region_name:KMS_key_account_id:key/KMS_key_id",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": [
                        "s3.region_name_of_the_canary_results_S3_bucket.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

AWS kebijakan terkelola untuk CloudWatch Synthetics

Menambahkan izin ke para pengguna, grup, dan peran lebih mudah dilakukan dengan menggunakan kebijakan terkelola AWS dibandingkan dengan menulis kebijakan sendiri. Dibutuhkan waktu dan keahlian untuk membuat kebijakan terkelola pelanggan IAM yang hanya menyediakan izin sesuai kebutuhan tim Anda. Untuk memulai dengan cepat, Anda dapat menggunakan kebijakan AWS terkelola kami. Kebijakan ini mencakup kasus penggunaan umum dan tersedia di AWS akun Anda. Untuk informasi selengkapnya tentang kebijakan terkelola AWS , silakan lihat kebijakan terkelola AWS kebijakan terkelola AWS dalam Panduan Pengguna IAM.

AWS layanan memelihara dan memperbarui kebijakan AWS terkelola. Anda tidak dapat mengubah izin dalam kebijakan AWS terkelola. Layanan kadang-kadang mengubah izin dalam kebijakan terkelola AWS . Jenis pembaruan ini memengaruhi semua identitas (pengguna, grup, dan peran) tempat kebijakan terlampir.

CloudWatch Synthetics memperbarui kebijakan terkelola AWS

Lihat detail tentang pembaruan kebijakan AWS terkelola untuk CloudWatch Synthetics sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di halaman Riwayat CloudWatch dokumen.

Perubahan	Deskripsi	Tanggal
Tindakan berlebihan dihapus dari CloudWatchSyntheticsFullAccess	CloudWatch Synthetics menghapus `lambda:GetLayerVersionByArn` tindakan `s3:PutBucketEncryption` dan tindakan dari CloudWatchSyntheticsFullAccess kebijakan karena tindakan tersebut berlebihan dengan izin lain dalam kebijakan. Tindakan yang dihapus tidak memberikan izin apa pun, dan tidak ada perubahan bersih untuk izin yang diberikan oleh kebijakan tersebut.	12 Maret 2021
CloudWatch Synthetics mulai melacak perubahan	CloudWatch Synthetics mulai melacak perubahan untuk kebijakan AWS terkelolanya.	10 Maret 2021

CloudWatchSyntheticsFullAccess

Berikut adalah isi dari kebijakan `CloudWatchSyntheticsFullAccess`:

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
```

```
            "synthetics:*"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:CreateBucket",
            "s3:PutEncryptionConfiguration"
         ],
         "Resource":[
            "arn:aws:s3:::cw-syn-results-*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:ListRoles",
            "s3:ListAllMyBuckets",
            "s3:GetBucketLocation",
            "xray:GetTraceSummaries",
            "xray:BatchGetTraces",
            "apigateway:GET"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObject",
            "s3:ListBucket"
         ],
         "Resource":"arn:aws:s3:::cw-syn-*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersion"
         ],
         "Resource":"arn:aws:s3:::aws-synthetics-library-*"
      },
      {
         "Effect":"Allow",
         "Action":[
```

```
            "iam:PassRole"
         ],
         "Resource":[
            "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*"
         ],
         "Condition":{
            "StringEquals":{
               "iam:PassedToService":[
                  "lambda.amazonaws.com",
                  "synthetics.amazonaws.com"
               ]
            }
         }
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:GetRole"
         ],
         "Resource":[
            "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "cloudwatch:GetMetricData",
            "cloudwatch:GetMetricStatistics"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "cloudwatch:PutMetricAlarm",
            "cloudwatch:DeleteAlarms"
         ],
         "Resource":[
            "arn:aws:cloudwatch:*:*:alarm:Synthetics-*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
```

```
            "cloudwatch:DescribeAlarms"
         ],
         "Resource":[
            "arn:aws:cloudwatch:*:*:alarm:*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "lambda:CreateFunction",
            "lambda:AddPermission",
            "lambda:PublishVersion",
            "lambda:UpdateFunctionConfiguration",
            "lambda:GetFunctionConfiguration"
         ],
         "Resource":[
            "arn:aws:lambda:*:*:function:cwsyn-*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "lambda:GetLayerVersion",
            "lambda:PublishLayerVersion"
         ],
         "Resource":[
            "arn:aws:lambda:*:*:layer:cwsyn-*",
            "arn:aws:lambda:*:*:layer:Synthetics:*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "ec2:DescribeVpcs",
            "ec2:DescribeSubnets",
            "ec2:DescribeSecurityGroups"
         ],
         "Resource":[
            "*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
```

```
            "sns:ListTopics"
         ],
         "Resource":[
            "*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "sns:CreateTopic",
            "sns:Subscribe",
            "sns:ListSubscriptionsByTopic"
         ],
         "Resource":[
            "arn:*:sns:*:*:Synthetics-*"
         ]
      }
   ]
}
```

CloudWatchSyntheticsReadOnlyAccess

Berikut adalah isi dari kebijakan `CloudWatchSyntheticsReadOnlyAccess`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "synthetics:Describe*",
                "synthetics:Get*",
                "synthetics:List*"
            ],
            "Resource": "*"
        }
    ]
}
```

Membatasi pengguna untuk melihat canary tertentu

Anda dapat membatasi kemampuan pengguna untuk melihat informasi tentang canary, sehingga mereka hanya dapat melihat informasi tentang canary yang Anda tentukan. Untuk melakukan ini,

gunakan kebijakan IAM dengan pernyataan `Condition` yang mirip dengan berikut ini, dan lampirkan kebijakan ini ke pengguna atau peran IAM.

Contoh berikut membatasi pengguna untuk hanya melihat informasi tentang `name-of-allowed-canary-1` dan `name-of-allowed-canary-2`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "synthetics:DescribeCanaries",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "synthetics:Names": [
                        "name-of-allowed-canary-1",
                        "name-of-allowed-canary-2"
                    ]
                }
            }
        }
    ]
}
```

CloudWatch Synthetics mendukung daftar sebanyak lima item dalam array. `synthetics:Names`

Anda juga dapat membuat kebijakan yang menggunakan * sebagai wildcard dalam nama canary yang diizinkan, seperti pada contoh berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "synthetics:DescribeCanaries",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringLike": {
                    "synthetics:Names": [
                        "my-team-canary-*"
```

```
                    ]
                }
            }
        }
    ]
}
```

Setiap pengguna yang masuk dengan salah satu kebijakan terlampir ini tidak dapat menggunakan CloudWatch konsol untuk melihat informasi kenari apa pun. Mereka dapat melihat informasi kenari hanya untuk kenari yang diotorisasi oleh kebijakan dan hanya dengan menggunakan DescribeCanariesAPI atau perintah AWS CLI deskripsi-kenari.

Membuat canary

> **Important**
>
> Pastikan bahwa Anda menggunakan canary Synthetics untuk memantau hanya titik akhir dan API di mana Anda memiliki kepemilikan atau izin. Bergantung pada pengaturan frekuensi canary, titik akhir ini mungkin mengalami peningkatan lalu lintas.

Saat Anda menggunakan CloudWatch konsol untuk membuat kenari, Anda dapat menggunakan cetak biru yang disediakan oleh CloudWatch untuk membuat kenari Anda atau Anda dapat menulis skrip Anda sendiri. Untuk informasi selengkapnya, lihat Menggunakan cetak biru canary.

Anda juga dapat membuat kenari menggunakan AWS CloudFormation jika Anda menggunakan skrip Anda sendiri untuk kenari. Untuk informasi selengkapnya, lihat AWS::Synthetics::Canarydi Panduan AWS CloudFormation Pengguna.

Jika Anda menulis skrip Anda sendiri, Anda dapat menggunakan beberapa fungsi yang CloudWatch Synthetics telah dibangun ke dalam perpustakaan. Untuk informasi selengkapnya, lihat Versi runtime Synthetics.

> **Note**
>
> Saat Anda membuat kenari, salah satu lapisan yang dibuat adalah layer Synthetics yang dilengkapi dengan. `Synthetics` Lapisan ini dimiliki oleh akun layanan Synthetics dan berisi kode runtime.

Untuk membuat canary

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Application Signals, Synthetics Canaries.
3. Pilih Buat Canary.
4. Pilih salah satu dari berikut:

   - Untuk membuat canary Anda di skrip cetak biru, pilih Gunakan cetak biru, dan kemudian pilih jenis canary yang ingin Anda buat. Untuk informasi selengkapnya tentang halyang dilakukan oleh setiap jenis cetak biru, silakan lihat Menggunakan cetak biru canary.
   - Untuk mengunggah skrip Node.js Anda sendiri untuk membuat sebuah canary kustom, pilih Unggah skrip.

     Kemudian, Anda dapat menyeret skrip Anda ke wilayah Skrip atau pilih Jelajahi file untuk menavigasi ke skrip di sistem file Anda.
   - Untuk mengimpor skrip dari bucket S3, pilih Impor dari S3. Di bawah Lokasi sumber, masukkan alur lengkap ke canary Anda atau pilih Jelajahi S3.

     Anda harus memiliki izin `s3:GetObject` dan `s3:GetObjectVersion` untuk bucket S3 yang digunakan. Ember harus berada di AWS Wilayah yang sama tempat Anda membuat kenari.
5. Di bawah Nama, masukkan nama untuk canary. Nama ini digunakan di banyak halaman, sehingga kami menyarankan Anda memberikan sebuah nama deskriptif yang membedakannya dari canary lainnya.
6. Di bawah Aplikasi atau URL titik akhir, masukkan URL yang Anda ingin lakukan pengujian oleh canary. URL ini harus menyertakan protokol (seperti https://).

   Jika Anda ingin canary menguji titik akhir pada VPC, Anda juga harus memasukkan informasi tentang VPC nantinya dalam prosedur ini.
7. Jika Anda menggunakan skrip milik Anda sendiri untuk canary, di bawah Pengatur Lambda, masukkan titik masuk tempat Anda ingin canary tersebut memulai. Jika Anda menggunakan runtime lebih awal dari `syn-nodejs-puppeteer-3.4` atau `syn-python-selenium-1.1`, string yang Anda masukkan harus diakhiri dengan `.handler`. Jika Anda menggunakan `syn-nodejs-puppeteer-3.4` atau `syn-python-selenium-1.1` atau runtime yang lebih baru, pembatasan ini tidak berlaku.

8. Jika Anda menggunakan variabel lingkungan dalam skrip Anda, pilih Variabel lingkungan dan kemudian menentukan nilai untuk setiap variabel lingkungan yang didefinisikan dalam skrip Anda. Untuk informasi selengkapnya, lihat Variabel-variabel lingkungan.
9. Di bawah Jadwal, pilih apakah akan menjalankan canary sekali saja, menjalankannya terus-menerus menggunakan ekspresi rate, atau menjadwalkan menggunakan ekspresi cron.
   - Saat Anda menggunakan CloudWatch konsol untuk membuat kenari yang berjalan terus menerus, Anda dapat memilih tarif di mana saja antara satu menit dan sekali dalam satu jam.
   - Untuk informasi selengkapnya tentang menulis ekspresi cron untuk penjadwalan canary, silakan lihat Penjadwalan operasi canary yang menggunakan cron.
10. (Opsional) Untuk menetapkan nilai batas waktu untuk canary, pilih Konfigurasi tambahan lalu tentukan nilai batas waktu. Buat tidak lebih pendek dari 15 detik untuk memungkinkan Lambda dingin dimulai dan waktu yang diperlukan untuk mem-boot instrumentasi canary.
11. Di bawah Retensi data, tentukan seberapa lama waktu yang dibutuhkan untuk mempertahankan informasi tentang operasi canary yang gagal dan berhasil. Rentangnya adalah 1-455 hari.

    Pengaturan ini hanya memengaruhi data yang disimpan dan ditampilkan CloudWatch Synthetics di konsol. Ini tidak memengaruhi data yang disimpan di bucket Amazon S3 Anda, atau log atau metrik yang diterbitkan oleh canary.
12. Di bawah Penyimpanan Data, pilih bucket S3 yang akan digunakan untuk menyimpan data dari operasi canary. Nama bucket tidak dapat berisi titik (.). Jika Anda mengosongkan ini, bucket S3 bawaan digunakan atau dibuat.

    Jika Anda menggunakan runtime `syn-nodejs-puppeteer-3.0` atau runtime yang lebih baru, ketika Anda memasukkan URL untuk bucket di kotak teks, Anda dapat menentukan bucket di Wilayah saat ini atau di Wilayah yang berbeda. Jika Anda menggunakan versi runtime sebelumnya, bucket harus berada di Wilayah saat ini.
13. (Opsional) Secara default, kenari menyimpan artefak mereka di Amazon S3, dan artefak dienkripsi saat diam menggunakan kunci -managed. AWS AWS KMS Anda dapat menggunakan opsi enkripsi yang berbeda dengan memilih Konfigurasi tambahan di bagian Penyimpanan Data. Anda kemudian dapat memilih jenis kunci yang akan digunakan untuk enkripsi. Untuk informasi selengkapnya, lihat Mengenkripsi artefak canary.
14. Di bawah Izin akses, pilih apakah akan membuat peran IAM untuk menjalankan canary atau menggunakan yang sudah ada.

Jika Anda memiliki CloudWatch Synthetics membuat peran, secara otomatis menyertakan semua izin yang diperlukan. Jika Anda ingin membuat peran sendiri, silakan lihat Peran dan izin yang diperlukan untuk canary informasi tentang izin yang diperlukan.

Jika Anda menggunakan CloudWatch konsol untuk membuat peran kenari saat membuat kenari, Anda tidak dapat menggunakan kembali peran tersebut untuk kenari lain, karena peran ini khusus untuk hanya satu kenari. Jika Anda telah membuat secara manual peran yang dapat digunakan untuk beberapa canary, Anda dapat menggunakan peran yang ada tersebut.

Untuk menggunakan peran yang ada, Anda harus memiliki `iam:PassRole` izin untuk melewati peran tersebut ke Synthetic dan Lambda. Anda juga harus memiliki `iam:GetRole` izin.

15. (Opsional) Di bawah Alarm, pilih apakah Anda ingin CloudWatch alarm default dibuat untuk kenari ini. Jika Anda memilih untuk membuat alarm, alarm tersebut dibuat dengan kesepakatan nama berikut: `Synthetics-Alarm-`*`canaryName`*`-`*`index`*

    `index` adalah angka yang mewakili setiap alarm berbeda yang dibuat untuk canary ini. Alarm pertama memiliki indeks 1, alarm kedua memiliki indeks 2, dan seterusnya.

16. (Opsional) Agar uji canary ini menjadi titik akhir pada VPC, pilih pengaturan VPC, lalu lakukan hal berikut:

    a. Pilih VPC yang menjadi host titik akhir.

    b. Pilih satu subnet atau lebih di VPC Anda. Anda harus memilih subnet privat karena instans Lambda tidak dapat dikonfigurasi untuk berfungsi di subnet publik ketika alamat IP tidak dapat ditetapkan untuk instans Lambda selama pelaksanaan. Untuk informasi selengkapnya, silakan lihat Mengonfigurasi Fungsi Lambda untuk Mengakses Sumber Daya dalam VPC.

    c. Pilih satu atau beberapa grup keamanan pada VPC Anda.

    Jika titik akhir ada di VPC, Anda harus mengaktifkan kenari Anda untuk mengirim informasi ke dan CloudWatch Amazon S3. Untuk informasi selengkapnya, lihat Menjalankan canary di VPC.

17. (Opsional) Di bawah Tag, tambahkan satu atau beberapa pasangan nilai kunci sebagai tanda untuk canary ini. Tag dapat membantu Anda mengidentifikasi dan mengatur AWS sumber daya Anda dan melacak AWS biaya Anda. Untuk informasi selengkapnya, lihat Pemberian tag pada sumber daya Amazon CloudWatch Anda.

18. (Opsional) Di bawah Penelusuran aktif, pilih apakah mengaktifkan penelusuran X-Ray aktif untuk canary ini. Pilihan ini hanya tersedia jika canary menggunakan versi runtime syn-nodejs-2.0 atau lebih baru. Untuk informasi selengkapnya, lihat Penelusuran Canary dan X-Ray.

Sumber daya yang dibuat untuk canary

Ketika Anda membuat canary, sumber daya berikut dibuat:

- Peran IAM dengan nama `CloudWatchSyntheticsRole-`*`canary-name`*`-`*`uuid`* (jika Anda menggunakan CloudWatch konsol untuk membuat kenari dan menentukan peran baru yang akan dibuat untuk kenari)
- Kebijakan IAM dengan nama `CloudWatchSyntheticsPolicy-`*`canary-name`*`-`*`uuid`*.
- bucket S3 dengan nama `cw-syn-results-`*`accountID`*`-`*`region`*.
- Alarm dengan nama `Synthetics-Alarm-`*`MyCanaryName`*, jika Anda ingin alarm dibuat untuk canary.
- Fungsi Lambda dan lapisan, jika Anda menggunakan cetak biru untuk membuat canary. Sumber daya ini memiliki awalan `cwsyn-`*`MyCanaryName`*.
- CloudWatch Log log grup dengan nama`/aws/lambda/cwsyn-`*`MyCanaryName`*`-`*`randomId`*.

Menggunakan cetak biru canary

Bagian ini menyediakan detail tentang setiap cetak biru canary dan tugas yang paling sesuai untuk setiap cetak biru. Cetak biru disediakan untuk jenis canary berikut:

- Pantau Heartbeat
- Canary API
- Pemeriksa Tautan yang Rusak
- Pemantauan visual
- Perekam Canary
- Alur Kerja GUI

Saat Anda menggunakan cetak biru untuk membuat kenari, saat Anda mengisi bidang di CloudWatch konsol, area editor Script pada halaman menampilkan kenari yang Anda buat sebagai skrip Node.js. Anda juga dapat menyunting canary di area ini untuk menyesuaikannya lebih lanjut.

Pemantauan heartbeat

Skrip heartbeat memuat URL tertentu dan menyimpan tangkapan layar halaman dan file arsip HTTP (file HAR). Skrip heartbeat juga menyimpan log URL yang diakses.

Anda dapat menggunakan file HAR untuk melihat data performa terperinci tentang halaman web. Anda dapat menganalisis daftar permintaan web dan menangkap masalah performa seperti waktu untuk memuat sebuah item.

Jika canary Anda menggunakan `syn-nodejs-puppeteer-3.1` atau versi runtime yang lebih baru, Anda dapat menggunakan cetak biru pemantauan heartbeat untuk memantau beberapa URL dan melihat status, durasi, tangkapan layar terkait, dan alasan kegagalan atau setiap URL dalam ringkasan langkah laporan berjalannya canary.

Canary API

Canary API dapat menguji fungsi Baca dan Tulis dasar dari API REST. REST adalah singkatan dari transfer status representasional dan merupakan seperangkat aturan yang diikuti pengembang ketika membuat API. Salah satu aturan ini menyatakan bahwa tautan ke URL tertentu harus mengembalikan satu data.

Canary dapat bekerja dengan API apa pun dan menguji semua jenis fungsionalitas. Setiap canary dapat membuat beberapa panggilan API.

Dalam canary yang menggunakan versi runtime `syn-nodejs-2.2` atau yang lebih baru, cetak biru canary API mendukung canary multi-langkah yang memantau API Anda sebagai langkah HTTP. Anda dapat menguji beberapa API dalam canary tunggal. Setiap langkah adalah permintaan terpisah yang dapat mengakses URL yang berbeda, menggunakan header yang berbeda, dan menggunakan aturan yang berbeda untuk apakah header dan bodi respons ditangkap. Dengan tidak menangkap header dan bodi respons, Anda dapat mencegah data sensitif dari perekaman.

Setiap permintaan di canary API terdiri atas informasi berikut:

- Titik akhir, yang merupakan URL yang Anda minta.
- Metode, yaitu jenis permintaan yang dikirim ke server. REST API mendukung operasi GET (baca), POST (tulis), PUT (perbarui), PATCH (perbarui), dan DELETE (hapus).
- Header, yang memberikan informasi baik kepada klien maupun server. Cookie ini digunakan untuk autentikasi dan memberikan informasi tentang konten bodi. Untuk daftar header yang valid, silakan lihat Header HTTP.

- Data (atau bodi), yang memuat informasi yang akan dikirimkan ke server. Hanya digunakan untuk permintaan POST, PUT, PATCH, atau DELETE.

Cetak biru canary API mendukung metode GET dan POST. Ketika Anda menggunakan cetak biru ini, Anda harus menentukan header. Misalnya, Anda dapat menentukan **`Authorization`** sebagai Kunci dan menetapkan data otorisasi yang diperlukan sebagai Nilai untuk kunci itu.

Jika Anda menguji permintaan POST, Anda juga menentukan konten yang akan diposting di bidang Data.

Integrasi dengan API Gateway

Cetak biru API terintegrasi dengan Amazon API Gateway. Ini memungkinkan Anda memilih API Gateway API dan tahap dari AWS akun dan Wilayah yang sama dengan canary, atau mengunggah template Swagger dari API Gateway untuk pemantauan API lintas akun dan lintas wilayah. Anda kemudian dapat memilih sisa dari detail di konsol untuk membuat canary, bukan memasukkannya dari awal. Untuk informasi selengkapnya tentang API Gateway, silakan lihat Apa itu Amazon API Gateway?

Menggunakan API privat

Anda dapat membuat canary yang menggunakan API privat di Amazon API Gateway. Untuk informasi selengkapnya, silakan lihat Membuat API privat di Amazon API Gateway?

Pemeriksa tautan yang rusak

Pemeriksa tautan yang rusak mengumpulkan semua tautan di dalam URL yang sedang Anda uji dengan menggunakan `document.getElementsByTagName('a')`. Hanya menguji hingga jumlah tautan yang Anda tentukan, dan URL itu sendiri dihitung sebagai tautan pertama. Sebagai contoh, jika Anda ingin memeriksa semua tautan di halaman yang memuat lima tautan, Anda harus menentukan untuk canary mengikuti enam tautan.

Canary pemeriksa tautan yang rusak dibuat menggunakan runtime `syn-nodejs-2.0-beta` atau yang lebih baru mendukung fitur tambahan berikut:

- Memberikan laporan yang menyertakan tautan yang diperiksa, kode status, alasan kegagalan (jika ada), dan halaman cuplikan layar sumber dan tujuan.
- Ketika melihat hasil canary, Anda dapat menyaring untuk hanya melihat tautan yang rusak dan kemudian memperbaiki tautan didasarkan pada alasan kegagalan.

- Versi ini menangkap cuplikan layar halaman sumber berketerangan untuk setiap tautan dan menyoroti jangkar tempat tautan ditemukan. Komponen tersembunyi tidak diberi keterangan.
- Anda dapat mengonfigurasi versi ini untuk mengambil tangkapan layar dari halaman sumber dan tujuan, hanya halaman sumber, atau hanya halaman tujuan.
- Versi ini memperbaiki masalah dalam versi sebelumnya tempat skrip canary berhenti setelah tautan pertama yang rusak, bahkan ketika lebih banyak tautan yang diambil dari halaman pertama.

Jika Anda ingin memperbarui canary yang sudah ada menggunakan `syn-1.0` untuk menggunakan runtime yang baru, Anda harus menghapus dan membuat ulang canary. Memperbarui canary yang sudah ada ke runtime baru tidak membuat fitur ini tersedia.

Canary pemeriksa tautan yang rusak mendeteksi jenis kesalahan tautan berikut:

- 404 Halaman Tidak Ditemukan
- Nama Host Tidak Benar
- URL Buruk. Misalnya, URL tidak memiliki tanda kurung, memiliki garis miring ekstra, atau menggunakan protokol yang salah.
- Kode respons HTTP tidak benar.
- Server host mengembalikan respons kosong dengan tanpa konten dan tidak ada kode respons.
- Permintaan HTTP secara konstan habis waktu selama berlangsungnya canary.
- Host secara konsisten menurunkan koneksi karena salah konfigurasi atau terlalu sibuk.

Cetak biru pemantauan visual

Cetak biru pemantauan visual mencakup kode untuk membandingkan tangkapan layar yang diambil selama berjalannya canary dengan tangkapan layar yang diambil selama berjalannya canary dasar. Jika perbedaan antara kedua tangkapan layar berada di luar persentase ambang batas, canary gagal. Pemantauan visual didukung di kenari yang menjalankan syn-puppeteer-node-3.2 dan yang lebih baru. Saat ini tidak didukung di canary yang menjalankan Python dan Selenium.

Cetak biru pemantauan visual mencakup baris kode berikut dalam skrip canary cetak biru default, yang memungkinkan pemantauan visual.

```
syntheticsConfiguration.withVisualCompareWithBaseRun(true);
```

Pertama kali canary berjalan dengan sukses setelah baris ini ditambahkan ke skrip, ia menggunakan tangkapan layar yang diambil selama proses itu sebagai dasar untuk perbandingan. Setelah kenari pertama dijalankan, Anda dapat menggunakan CloudWatch konsol untuk mengedit kenari untuk melakukan salah satu hal berikut:

- Tetapkan putaran canary berikutnya sebagai dasar baru.
- Gambar batas pada tangkapan layar dasar saat ini untuk menunjuk area tangkapan layar untuk diabaikan selama perbandingan visual.
- Hapus tangkapan layar agar tidak digunakan untuk pemantauan visual.

Untuk informasi selengkapnya tentang menggunakan CloudWatch konsol untuk mengedit kenari, lihatMengedit atau menghapus canary.

Anda juga dapat mengubah canary run yang digunakan sebagai baseline dengan menggunakan `nextrun` atau `lastrun` parameter atau menentukan ID run canary di API. UpdateCanary

Saat Anda menggunakan cetak biru pemantauan visual, Anda memasukkan URL tempat Anda ingin tangkapan layar diambil, dan menentukan ambang batas perbedaan sebagai persentase. Setelah baseline run, future run dari canary yang mendeteksi perbedaan visual yang lebih besar dari ambang batas itu memicu kegagalan canary. Setelah baseline berjalan, Anda juga dapat mengedit canary untuk "menggambar" batas pada tangkapan layar dasar yang ingin Anda abaikan selama pemantauan visual.

Fitur pemantauan visual didukung oleh toolkit perangkat lunak ImageMagick open source. Untuk informasi lebih lanjut, lihat ImageMagick.

Perekam canary

Dengan cetak biru perekam kenari, Anda dapat menggunakan CloudWatch Synthetics Recorder untuk merekam tindakan klik dan ketik Anda di situs web dan secara otomatis menghasilkan skrip Node.js yang dapat digunakan untuk membuat kenari yang mengikuti langkah yang sama. CloudWatchSynthetics Recorder adalah ekstensi Google Chrome yang disediakan oleh Amazon.

Kredit: Perekam CloudWatch Synthetics didasarkan pada perekam Tanpa Kepala.

Untuk informasi selengkapnya, lihat Menggunakan CloudWatch Synthetics Recorder untuk Google Chrome.

Pembangun alur kerja GUI

Cetak biru Pembangun Alur Kerja GUI memverifikasi bahwa tindakan dapat diambil di halaman web Anda. Sebagai contoh, jika Anda memiliki halaman web dengan formulir login, canary dapat mengisi bidang pengguna dan kata sandi serta mengirimkan formulir untuk memverifikasi bahwa halaman web berfungsi dengan benar.

Ketika Anda menggunakan cetak biru untuk membuat jenis canary ini, Anda menentukan tindakan yang diinginkan canary untuk dilakukan pada halaman web. Tindakan yang dapat Anda gunakan adalah sebagai berikut:

- Klik— Pilih elemen yang Anda tentukan dan simulasikan pengguna yang mengeklik atau memilih elemen tersebut.

  Untuk menentukan elemen dalam skrip Node.js, gunakan `[id=]` atau `a[class=]`.

  Untuk menentukan elemen dalam skrip Python, gunakan `xpath //*[@id=]` atau `//*[@class=]`.
- Verifikasi pemilih— Verifikasi bahwa elemen tertentu ada di halaman web tersebut. Uji ini berguna untuk memverifikasi bahwa tindakan sebelumnya memiliki elemen yang benar untuk mengisi halaman tersebut.

  Untuk menentukan elemen guna memverifikasi dalam skrip Node.js, gunakan `[id=]` atau `a[class=]`.

  Untuk menentukan elemen guna memverifikasi dalam skrip Python, gunakan `xpath //*[@id=]` atau `//*[class=]`.
- Verifikasi teks— Verifikasi bahwa rangkaian tertentu termuat dalam elemen target. Uji ini berguna untuk memverifikasi bahwa tindakan sebelumnya telah menyebabkan teks yang benar ditampilkan.

  Untuk menentukan elemen dalam skrip Node.js, gunakan format seperti `div[@id=]//h1` karena tindakan ini menggunakan fungsi `waitForXPath` dalam Puppeteer.

  Untuk menentukan elemen dalam skrip Python, gunakan format xpath seperti `//*[@id=]` atau //*[@class=] karena tindakan ini menggunakan fungsi `implicitly_wait` dalam Selenium.
- Masukkan teks— Tulis teks tertentu dalam elemen target.

  Untuk menentukan elemen guna memverifikasi dalam skrip Node.js, gunakan `[id=]` atau `a[class=]`.

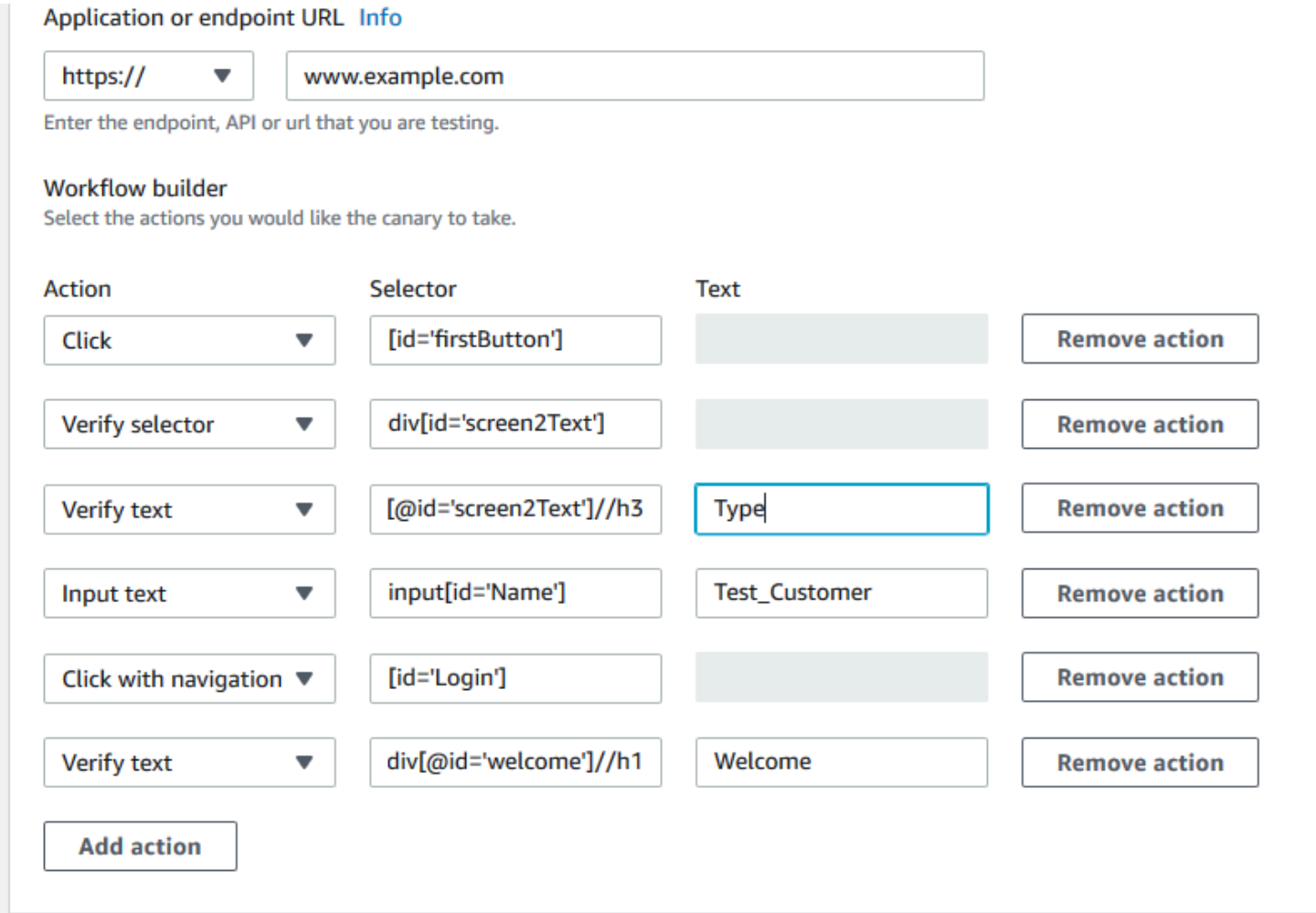
Untuk menentukan elemen guna memverifikasi dalam skrip Python, gunakan `xpath //*[@id=]` atau `//*[@class=]`.

- Klik dengan navigasi— Tunggu seluruh halaman dimuat setelah memilih elemen tertentu. Hal ini paling berguna ketika Anda perlu memuat ulang halaman.

  Untuk menentukan elemen dalam skrip Node.js, gunakan `[id=]` atau `a[class=]`.

  Untuk menentukan elemen dalam skrip Python, gunakan `xpath //*[@id=]` atau `//*[@class=]`.

Sebagai contoh, cetak biru berikut menggunakan Node.js. Cetak biru mengeklik firstButton pada URL tertentu, memverifikasi bahwa pemilih yang diharapkan dengan teks yang diharapkan muncul, memasukkan nama `Test_Customer` ke dalam bidang Nama, mengklik tombol Login, dan kemudian memverifikasi bahwa login berhasil dengan memeriksa teks Sambutan pada halaman berikutnya.



Canary alur kerja GUI yang menggunakan runtime berikut juga memberikan ringkasan langkah-langkah yang dilaksanakan untuk masing-masing operasi canary. Anda dapat menggunakan tangkapan layar dan pesan kesalahan yang terkait dengan setiap langkah untuk menemukan akar masalah kegagalan.

- `syn-nodejs-2.0` atau yang lebih baru
- `syn-python-selenium-1.0` atau yang lebih baru

Menggunakan CloudWatch Synthetics Recorder untuk Google Chrome

Amazon menyediakan Perekam CloudWatch Synthetics untuk membantu Anda membuat kenari dengan lebih mudah. Perekam adalah ekstensi Google Chrome.

Perekam mencatat tindakan klik dan jenis tindakan di situs web dan secara otomatis membuat skrip Node.js yang dapat digunakan untuk membuat canary yang mengikuti langkah serupa.

Setelah Anda mulai merekam, CloudWatch Synthetics Recorder mendeteksi tindakan Anda di browser dan mengubahnya menjadi skrip. Anda dapat menjeda dan melanjutkan perekaman sesuai kebutuhan. Ketika Anda berhenti merekam, perekam membuat skrip tindakan Node.js, yang dapat Anda salin dengan mudah bersama tombol Salin ke Clipboard. Anda kemudian dapat menggunakan skrip ini untuk membuat kenari di CloudWatch Synthetics.

Kredit: Perekam CloudWatch Synthetics didasarkan pada perekam Tanpa Kepala.

Memasang ekstensi CloudWatch Synthetics Recorder untuk Google Chrome

Untuk menggunakan CloudWatch Synthetics Recorder, Anda dapat mulai membuat kenari dan memilih cetak biru Canary Recorder. Jika Anda melakukan ini ketika Anda belum mengunduh perekam, konsol CloudWatch Synthetics menyediakan tautan untuk mengunduhnya.

Atau, Anda dapat mengikuti langkah-langkah ini untuk mengunduh dan melakukan instalasi perekam secara langsung.

Untuk menginstal CloudWatch Synthetics Recorder

1. Menggunakan Google Chrome, buka situs web ini: https://chrome.google.com/webstore/detail/cloudwatch-synthetics-rec /bhdnlmmgiplmbcdmkkdfplenecpegfno
2. Pilih Tambahkan ke Chrome, lalu pilih Tambahkan ekstensi.

Menggunakan CloudWatch Synthetics Recorder untuk Google Chrome

Untuk menggunakan CloudWatch Synthetics Recorder untuk membantu Anda membuat kenari, Anda dapat memilih Buat kenari di CloudWatch konsol, lalu pilih Gunakan cetak biru, Perekam Canary. Untuk informasi selengkapnya, lihat Membuat canary.

Atau, Anda dapat menggunakan perekam untuk mencatat langkah-langkah tanpa langsung menggunakannya guna membuat canary.

Untuk menggunakan CloudWatch Synthetics Recorder untuk merekam tindakan Anda di situs web

1. Navigasi ke halaman yang ingin Anda pantau.
2. Pilih ikon ekstensi Chrome, lalu pilih CloudWatchSynthetics Recorder.
3. Pilih Mulai Perekaman.
4. Lakukan langkah-langkah yang ingin Anda catat. Untuk menjeda perekaman, pilih Jeda.
5. Setelah selesai merekam alur kerja, pilih Hentikan perekaman.
6. Pilih Salin ke clipboard untuk menyalin skrip yang dihasilkan ke clipboard Anda. Atau, jika Anda ingin memulai ulang, pilih Perekaman baru.
7. Untuk membuat canary dengan skrip salinan, Anda dapat menempelkan skrip salinan ke penyunting selaras cetak biru perekam, atau menyimpannya ke bucket Amazon S3 dan mengimpornya dari sana.
8. Jika tidak segera membuat canary, Anda dapat menyimpan skrip rekaman ke dalam file.

Keterbatasan yang diketahui dari Perekam CloudWatch Synthetics

Perekam CloudWatch Synthetics untuk Google Chrome saat ini memiliki batasan berikut.

- Elemen HTML yang tidak memiliki ID akan menggunakan pemilih CSS. Ini dapat memecah canary jika struktur halaman web nanti berubah. Kami berencana untuk menyediakan beberapa pilihan konfigurasi (seperti menggunakan data-id) di sekitar ini dalam versi perekam di masa mendatang.
- Perekam tidak mendukung tindakan seperti klik ganda atau salin/tempel, dan tidak mendukung kombinasi utama seperti CMD+0.
- Untuk memverifikasi keberadaan elemen atau teks di halaman, pengguna harus menambahkan pernyataan setelah skrip dibuat. Perekam tidak mendukung verifikasi elemen tanpa melakukan tindakan apa pun pada elemen tersebut. Hal ini mirip dengan pilihan “Verifikasi teks” atau “Verifikasi elemen” di pembangun alur kerja canary. Kami berencana untuk menambahkan beberapa dukungan pernyataan di versi berikutnya dari perekam.

- Perekam mencatat semua tindakan dalam tab tempat perekaman dimulai. Ini tidak merekam pop-up (misalnya, untuk memungkinkan penelusuran lokasi) atau navigasi ke berbagai halaman dari pop-up.

Versi runtime Synthetics

Ketika Anda membuat atau memperbarui canary, Anda memilih versi runtime Synthetic untuk canary. Runtime Synthetic adalah kombinasi kode Synthetic yang memanggil pengatur skrip Anda, dan lapisan Lambda dari Dependensi gabungan.

CloudWatch Synthetics saat ini mendukung runtime yang menggunakan Node.js untuk skrip dan framework Puppeteer, dan runtime yang menggunakan Python untuk scripting dan Selenium Webdriver untuk framework.

Kami menyarankan agar Anda selalu menggunakan versi runtime terbaru untuk canary Anda, agar dapat menggunakan fitur dan pembaruan terbaru yang dibuat untuk pustaka Synthetic.

Saat Anda membuat kenari, salah satu lapisan yang dibuat adalah layer Synthetics yang dilengkapi dengan. `Synthetics` Lapisan ini dimiliki oleh akun layanan Synthetics dan berisi kode runtime.

> **Note**
>
> Setiap kali Anda meningkatkan canary untuk menggunakan versi baru runtime Synthetics, semua fungsi pustaka Synthetics yang digunakan canary Anda juga secara otomatis ditingkatkan ke versi NodeJS yang sama dengan yang didukung runtime Synthetics.

Topik

CloudWatch Kebijakan dukungan runtime Synthetics

Versi runtime Synthetic bergantung pada pembaruan pemeliharaan dan keamanan. Ketika komponen mana pun dari sebuah versi runtime tidak lagi didukung, maka versi runtime Synthetics itu tidak lagi digunakan.

Anda tidak dapat membuat canary menggunakan versi runtime yang tidak lagi digunakan. Canary yang menggunakan runtime yang tidak lagi digunakan terus berjalan. Anda dapat menghentikan, memulai, dan menghapus canary ini. Anda dapat memperbarui canary yang sudah ada yang menggunakan versi runtime tercatat dengan memperbarui canary untuk menggunakan versi runtime yang didukung.

CloudWatch Synthetics memberi tahu Anda melalui email jika Anda memiliki kenari yang menggunakan runtime yang dijadwalkan tidak digunakan lagi dalam 60 hari ke depan. Kami merekomendasikan Anda memigrasi canary Anda ke versi runtime yang didukung untuk mendapatkan keuntungan dari peningkatan fungsionalitas, keamanan, dan peningkatan performa baru yang disertakan dalam rilis terbaru.

Bagaimana cara memperbarui canary ke versi runtime baru?

Anda dapat memperbarui versi runtime kenari dengan menggunakan CloudWatch konsol, AWS CloudFormation, AWS CLI atau SDK. AWS Saat Anda menggunakan CloudWatch konsol, Anda dapat memperbarui hingga lima kenari sekaligus dengan memilihnya di halaman daftar kenari dan kemudian memilih Tindakan, Perbarui Runtime.

Anda dapat memverifikasi peningkatan dengan terlebih dahulu mengkloning kenari menggunakan CloudWatch konsol dan memperbarui versi runtime-nya. Hal ini menciptakan canary lain yang merupakan klon dari canary asli Anda. Setelah Anda memverifikasi canary Anda dengan versi runtime yang baru, Anda dapat memperbarui versi runtime canary asli Anda dan menghapus canary klon tersebut.

Anda juga dapat memperbarui beberapa canary menggunakan skrip peningkatan. Untuk informasi selengkapnya, lihat Skrip peningkatan runtime canary.

Jika Anda meningkatkan canary dan gagal, silakan lihat Memecahkan masalah canary yang gagal.

Tanggal pengusangan runtime

Versi Runtime	Tanggal pengusangan
`syn-nodejs-puppeteer-6.1`	8 Maret 2024
`syn-nodejs-puppeteer-6.0`	8 Maret 2024

Versi Runtime	Tanggal pengusangan
`syn-nodejs-puppeteer-5.1`	8 Maret 2024
`syn-nodejs-puppeteer-5.0`	8 Maret 2024
`syn-nodejs-puppeteer-4.0`	8 Maret 2024
`syn-nodejs-puppeteer-3.9`	8 Januari 2024
`syn-nodejs-puppeteer-3.8`	8 Januari 2024
`syn-python-selenium-2.0`	8 Maret 2024
`syn-python-selenium-1.3`	8 Maret 2024
`syn-python-selenium-1.2`	8 Maret 2024
`syn-python-selenium-1.1`	8 Maret 2024
`syn-python-selenium-1.0`	8 Maret 2024
`syn-nodejs-puppeteer-3.7`	8 Januari 2024
`syn-nodejs-puppeteer-3.6`	8 Januari 2024
`syn-nodejs-puppeteer-3.5`	8 Januari 2024

Versi Runtime	Tanggal pengusangan
syn-nodejs-puppeteer-3.4	13 November 2022
syn-nodejs-puppeteer-3.3	13 November 2022
syn-nodejs-puppeteer-3.2	13 November 2022
syn-nodejs-puppeteer-3.1	13 November 2022
syn-nodejs-puppeteer-3.0	13 November 2022
syn-nodejs-2.2	28 Mei 2021
syn-nodejs-2.1	28 Mei 2021
syn-nodejs-2.0	28 Mei 2021
syn-nodejs-2.0-beta	8 Februari 2021
syn-1.0	28 Mei 2021

Skrip peningkatan runtime canary

Untuk meningkatkan skrip canary untuk versi runtime yang didukung, gunakan skrip berikut.

```
const AWS = require('aws-sdk');

// You need to configure your AWS credentials and Region.
//   https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/setting-
credentials-node.html
//   https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/setting-
region.html
```

```
const synthetics = new AWS.Synthetics();

const DEFAULT_OPTIONS = {
  /**
   * The number of canaries to upgrade during a single run of this script.
   */
  count: 10,
  /**
   * No canaries are upgraded unless force is specified.
   */
  force: false
};

/**
 * The number of milliseconds to sleep between GetCanary calls when
 * verifying that an update succeeded.
 */
const SLEEP_TIME = 5000;

(async () => {
  try {
    const options = getOptions();

    const versions = await getRuntimeVersions();
    const canaries = await getAllCanaries();
    const upgrades = canaries
      .filter(canary => !versions.isLatestVersion(canary.RuntimeVersion))
      .map(canary => {
        return {
          Name: canary.Name,
          FromVersion: canary.RuntimeVersion,
          ToVersion: versions.getLatestVersion(canary.RuntimeVersion)
        };
      });

    if (options.force) {
      const promises = [];

      for (const upgrade of upgrades.slice(0, options.count)) {
        const promise = upgradeCanary(upgrade);
        promises.push(promise);
        // Sleep for 100 milliseconds to avoid throttling.
        await usleep(100);
      }
```

```
      const succeeded = [];
      const failed = [];
      for (let i = 0; i < upgrades.slice(0, options.count).length; i++) {
        const upgrade = upgrades[i];
        const promise = promises[i];
        try {
          await promise;
          console.log(`The update of ${upgrade.Name} succeeded.`);
          succeeded.push(upgrade.Name);
        } catch (e) {
          console.log(`The update of ${upgrade.Name} failed with error: ${e}`);
          failed.push({
            Name: upgrade.Name,
            Reason: e
          });
        }
      }

      if (succeeded.length) {
        console.group('The following canaries were upgraded successfully.');
        for (const name of succeeded) {
          console.log(name);
        }
        console.groupEnd()
      } else {
        console.log('No canaries were upgraded successfully.');
      }

      if (failed.length) {
        console.group('The following canaries were not upgraded successfully.');
        for (const failure of failed) {
          console.log('\x1b[31m', `${failure.Name}: ${failure.Reason}`, '\x1b[0m');
        }
        console.groupEnd();
      }
    } else {
      console.log('Run with --force [--count <count>] to perform the first <count>
upgrades shown. The default value of <count> is 10.')
      console.table(upgrades);
    }
  } catch (e) {
    console.error(e);
  }
```

```
})();

function getOptions() {
  const force = getFlag('--force', DEFAULT_OPTIONS.force);
  const count = getOption('--count', DEFAULT_OPTIONS.count);
  return { force, count };

  function getFlag(key, defaultValue) {
    return process.argv.includes(key) || defaultValue;
  }
  function getOption(key, defaultValue) {
    const index = process.argv.indexOf(key);
    if (index < 0) {
      return defaultValue;
    }
    const value = process.argv[index + 1];
    if (typeof value === 'undefined' || value.startsWith('-')) {
      throw `The ${key} option requires a value.`;
    }
    return value;
  }
}

function getAllCanaries() {
  return new Promise((resolve, reject) => {
    const canaries = [];

    synthetics.describeCanaries().eachPage((err, data) => {
      if (err) {
        reject(err);
      } else {
        if (data === null) {
          resolve(canaries);
        } else {
          canaries.push(...data.Canaries);
        }
      }
    });
  });
}

function getRuntimeVersions() {
  return new Promise((resolve, reject) => {
    const jsVersions = [];
```

```
    const pythonVersions = [];
    synthetics.describeRuntimeVersions().eachPage((err, data) => {
      if (err) {
        reject(err);
      } else {
        if (data === null) {
          jsVersions.sort((a, b) => a.ReleaseDate - b.ReleaseDate);
          pythonVersions.sort((a, b) => a.ReleaseDate - b.ReleaseDate);
          resolve({
            isLatestVersion(version) {
              const latest = this.getLatestVersion(version);
              return latest === version;
            },
            getLatestVersion(version) {
              if (jsVersions.some(v => v.VersionName === version)) {
                return jsVersions[jsVersions.length - 1].VersionName;
              } else if (pythonVersions.some(v => v.VersionName === version)) {
                return pythonVersions[pythonVersions.length - 1].VersionName;
              } else {
                throw Error(`Unknown version ${version}`);
              }
            }
          });
        } else {
          for (const version of data.RuntimeVersions) {
            if (version.VersionName === 'syn-1.0') {
              jsVersions.push(version);
            } else if (version.VersionName.startsWith('syn-nodejs-2.')) {
              jsVersions.push(version);
            } else if (version.VersionName.startsWith('syn-nodejs-puppeteer-')) {
              jsVersions.push(version);
            } else if (version.VersionName.startsWith('syn-python-selenium-')) {
              pythonVersions.push(version);
            } else {
              throw Error(`Unknown version ${version.VersionName}`);
            }
          }
        }
      }
    });
  });
}

async function upgradeCanary(upgrade) {
```

```
  console.log(`Upgrading canary ${upgrade.Name} from ${upgrade.FromVersion} to
 ${upgrade.ToVersion}`);
  await synthetics.updateCanary({ Name: upgrade.Name, RuntimeVersion:
 upgrade.ToVersion }).promise();
  while (true) {
    await usleep(SLEEP_TIME);
    console.log(`Getting the state of canary ${upgrade.Name}`);
    const response = await synthetics.getCanary({ Name: upgrade.Name }).promise();
    const state = response.Canary.Status.State;
    console.log(`The state of canary ${upgrade.Name} is ${state}`);
    if (state === 'ERROR' || response.Canary.Status.StateReason) {
      throw response.Canary.Status.StateReason;
    }
    if (state !== 'UPDATING') {
      return;
    }
  }
}

function usleep(ms) {
  return new Promise(resolve => setTimeout(resolve, ms));
}
```

Versi runtime yang menggunakan Node.js dan Puppeteer

Versi runtime pertama untuk Node.js dan Puppeteer bernama `syn-1.0`. Kemudian versi runtime selanjutnya memiliki konvensi penamaan `syn-`*`language`*`-`*`majorversion`*`.`*`minorversion`*. Dimulai dengan `syn-nodejs-puppeteer-3.0`, konvensi penamaannya adalah `syn-`*`language`*`-`*`framework`*`-`*`majorversion`*`.`*`minorversion`*

Tambahan `-beta` akhiran menunjukkan bahwa versi runtime saat ini ada dalam rilis pratinjau beta.

Versi runtime dengan nomor versi utama yang sama kompatibel dengan versi sebelumnya.

> **Important**
>
> Versi runtime CloudWatch Synthetics berikut dijadwalkan tidak digunakan lagi pada 8 Maret 2024.
>
> - `syn-nodejs-puppeteer-6.1`
> - `syn-nodejs-puppeteer-6.0`
> - `syn-nodejs-puppeteer-5.1`

- `syn-nodejs-puppeteer-5.0`
- `syn-nodejs-puppeteer-4.0`

Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Important

PENTING: Dependensi AWS SDK untuk JavaScript v2 yang disertakan akan dihapus dan diperbarui untuk menggunakan AWS SDK untuk JavaScript v3 dalam rilis runtime mendatang. Ketika itu terjadi, Anda dapat memperbarui referensi kode canary Anda. Atau, Anda dapat melanjutkan referensi dan menggunakan AWS SDK yang disertakan untuk ketergantungan JavaScript v2 dengan menambahkannya sebagai dependensi ke file zip kode sumber Anda.

Catatan untuk semua versi runtime

Ketika menggunakan versi runtime `syn-nodejs-puppeteer-3.0`, pastikan bahwa skrip canary Anda kompatibel dengan Node.js 12.x. Jika Anda menggunakan versi sebelumnya dari versi runtime `syn-nodejs`, pastikan bahwa skrip Anda kompatibel dengan Node.js 10.x.

Kode Lambda di canary dikonfigurasi untuk memiliki memori maksimum 1 GB. Setiap proses waktu canary di luar setelah nilai waktu habis yang dikonfigurasi. Jika tidak ada nilai batas waktu yang ditentukan untuk kenari, CloudWatch pilih nilai batas waktu berdasarkan frekuensi kenari. Jika Anda mengonfigurasi nilai batas waktu, buatlah tidak lebih pendek dari 15 detik untuk memungkinkan Lambda cold start dan waktu yang diperlukan untuk mem-boot instrumentasi canary.

Note

Versi runtime CloudWatch Synthetics berikut tidak digunakan lagi pada 8 Januari 2024. Ini karena runtime Lambda Node.js 14 tidak digunakan lagi pada 4 AWS Lambda Desember 2023.

- `syn-nodejs-puppeteer-3.9`
- `syn-nodejs-puppeteer-3.8`
- `syn-nodejs-puppeteer-3.7`

- `syn-nodejs-puppeteer-3.6`
- `syn-nodejs-puppeteer-3.5`

Versi runtime CloudWatch Synthetics berikut tidak digunakan lagi pada 13 November 2022. Ini karena AWS Lambda runtime Lambda Node.js 12 tidak digunakan lagi pada 14 November 2022.

- `syn-nodejs-puppeteer-3.4`
- `syn-nodejs-puppeteer-3.3`
- `syn-nodejs-puppeteer-3.2`
- `syn-nodejs-puppeteer-3.1`
- `syn-nodejs-puppeteer-3.0`

Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

syn-nodejs-puppeteer-7.0

`syn-nodejs-puppeteer-7.0`Runtime adalah versi runtime terbaru untuk Lambda runtime Node.js 18.x. Ini menggunakan Node.js dan Dalang.

Dependensi besar:

- Runtime Lambda Node.js 18.x
- Puppeteer-core versi 21.9.0
- Chromium versi 121.0.6167.139

Ukuran kode:

Ukuran kode dan dependensi yang dapat Anda paket ke dalam runtime ini adalah 80 MB.

Fitur baru di syn-nodejs-puppeteer -7.0:

- Versi terbaru dari pustaka yang dibundel di Puppeteer dan Chromium - Dependensi Dalang dan Chromium diperbarui ke versi baru.

> **Important**
>
> Pindah dari Puppeteer 19.7.0 ke Puppeteer 21.9.0 memperkenalkan perubahan yang melanggar terkait pengujian dan filter. Untuk informasi lebih lanjut, lihat bagian BREAKING CHANGES didalang: v20.0.0 dan dalang inti: v21.0.0.

Direkomendasikan upgrade ke AWS SDK v3

Runtime Lambda nodejs18.x tidak mendukung SDK v2. AWS Kami sangat menyarankan Anda bermigrasi ke AWS SDK v3.

syn-nodejs-puppeteer-6.2

Dependensi besar:

- Runtime Lambda Node.js 18.x
- Puppeteer-core versi 19.7.0
- Chromium versi 111.0.5563.146

Fitur baru di syn-nodejs-puppeteer -6.2:

- Versi terbaru dari pustaka yang dibundel di Chromium
- Pemantauan penyimpanan singkat - Runtime ini menambahkan pemantauan penyimpanan sementara di akun pelanggan.
- Perbaikan bug

syn-nodejs-puppeteer-5,2

`syn-nodejs-puppeteer-5.2`Runtime adalah versi runtime terbaru untuk Lambda runtime Node.js 16.x. Ini menggunakan Node.js dan Dalang.

Dependensi besar:

- Runtime Lambda Node.js 16.x
- Puppeteer-core versi 19.7.0

- Chromium versi 111.0.5563.146

Fitur baru di syn-nodejs-puppeteer -5.2:

- Versi terbaru dari pustaka yang dibundel di Chromium
- Perbaikan bug

syn-nodejs-puppeteer-6.1

> **Important**
>
> Versi runtime ini dijadwalkan tidak digunakan lagi pada 8 Maret 2024. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Dependensi besar:

- Runtime Lambda Node.js 18.x
- Puppeteer-core versi 19.7.0
- Chromium versi 111.0.5563.146

Fitur baru di syn-nodejs-puppeteer -6.1:

- Peningkatan stabilitas— Penambahan logika coba ulang otomatis untuk menangani kesalahan peluncuran Puppeteer intermiten.
- Peningkatan Dependensi— Peningkatan untuk beberapa paket dependensi pihak ketiga.
- Canary tanpa izin Amazon S3— Perbaikan bug, sehingga canary yang tidak memiliki izin Amazon S3 masih dapat berjalan. Canary yang tidak memiliki izin Amazon S3 ini tidak akan dapat mengunggah tangkapan layar atau artefak lainnya ke Amazon S3. Untuk informasi selengkapnya tentang izin-izin untuk canary, silakan lihat Peran dan izin yang diperlukan untuk canary.

> **Important**
>
> PENTING: Dependensi AWS SDK untuk JavaScript v2 yang disertakan akan dihapus dan diperbarui untuk menggunakan AWS SDK untuk JavaScript v3 dalam rilis runtime mendatang. Ketika itu terjadi, Anda dapat memperbarui referensi kode canary Anda. Atau,

Anda dapat melanjutkan referensi dan menggunakan AWS SDK yang disertakan untuk ketergantungan JavaScript v2 dengan menambahkannya sebagai dependensi ke file zip kode sumber Anda.

syn-nodejs-puppeteer-6.0

Important

Versi runtime ini dijadwalkan tidak digunakan lagi pada 8 Maret 2024. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Dependensi besar:

- Runtime Lambda Node.js 18.x
- Puppeteer-core versi 19.7.0
- Chromium versi 111.0.5563.146

Fitur baru di syn-nodejs-puppeteer -6.0:

- Peningkatan dependensi — Dependensi Node.js ditingkatkan ke 18.x.
- Dukungan mode intercept — Dukungan mode intercept kooperatif Puppeteer ditambahkan ke pustaka runtime canary Synthetics.
- Melacak perubahan perilaku — Mengubah perilaku pelacakan default untuk melacak hanya permintaan fetch dan xhr, dan tidak melacak permintaan sumber daya. Anda dapat mengaktifkan pelacakan permintaan sumber daya dengan mengonfigurasi opsi `traceResourceRequests`.
- Metrik durasi disempurnakan — `Duration` Metrik sekarang mengecualikan waktu operasi yang digunakan kenari untuk mengunggah artefak, mengambil tangkapan layar, dan menghasilkan metrik. CloudWatch `Duration`nilai metrik dilaporkan CloudWatch, dan Anda juga dapat melihatnya di konsol Synthetics.
- Perbaikan bug— Bersihkan dump inti yang dihasilkan saat Chromium crash saat canary dijalankan.

> **Important**
>
> PENTING: Dependensi AWS SDK untuk JavaScript v2 yang disertakan akan dihapus dan diperbarui untuk menggunakan AWS SDK untuk JavaScript v3 dalam rilis runtime mendatang. Ketika itu terjadi, Anda dapat memperbarui referensi kode canary Anda. Atau, Anda dapat melanjutkan referensi dan menggunakan AWS SDK yang disertakan untuk ketergantungan JavaScript v2 dengan menambahkannya sebagai dependensi ke file zip kode sumber Anda.

syn-nodejs-puppeteer-5.1

> **Important**
>
> Versi runtime ini dijadwalkan tidak digunakan lagi pada 8 Maret 2024. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Dependensi besar:

- Runtime Lambda Node.js 16.x
- Puppeteer-core versi 19.7.0
- Chromium versi 111.0.5563.146

Perbaikan bug di syn-nodejs-puppeteer -5.1:

- Perbaikan bug— Runtime ini memperbaiki bug di `syn-nodejs-puppeteer-5.0` di mana file HAR yang dibuat oleh bcanarytidak memiliki header permintaan.

syn-nodejs-puppeteer-5,0

> **Important**
>
> Versi runtime ini dijadwalkan tidak digunakan lagi pada 8 Maret 2024. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Dependensi besar:

- Runtime Lambda Node.js 16.x
- Puppeteer-core versi 19.7.0
- Chromium versi 111.0.5563.146

Fitur baru di syn-nodejs-puppeteer -5.0:

- Peningkatan dependensi — Versi Puppeteer-core diperbarui ke 19.7.0. Versi Chromium ditingkatkan ke 111.0.5563.146.

> **Important**
>
> Versi Puppeteer-Core yang baru tidak sepenuhnya kompatibel dengan versi Puppeteer sebelumnya. Beberapa perubahan dalam versi ini dapat menyebabkan canary yang ada yang menggunakan fungsi Puppeteer yang tidak digunakan lagi gagal. Untuk informasi selengkapnya, silakan lihat perubahan yang melanggar log perubahan untuk Puppeteer-core versi 19.7.0 hingga 6.0, di log perubahan Puppeteer.

syn-nodejs-puppeteer-4,0

> **Important**
>
> Versi runtime ini dijadwalkan tidak digunakan lagi pada 8 Maret 2024. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Dependensi besar:

- Runtime Lambda Node.js 16.x
- Puppeteer-core versi 5.5.0
- Chromium versi 92.0.4512

Fitur baru di syn-nodejs-puppeteer -4.0:

- Peningkatan Dependensi— Dependensi Node.js diperbarui ke 16.x.

Runtime usang untuk Node.js dan Puppeteer

Runtime berikut untuk Node.js dan Puppeteer telah usang.

syn-nodejs-puppeteer-3,9

> **Important**
>
> Versi runtime ini tidak digunakan lagi pada 8 Januari 2024. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Dependensi besar:

- Runtime Lambda Node.js 14.x
- Puppeteer-core versi 5.5.0
- Chromium versi 92.0.4512

Fitur baru di syn-nodejs-puppeteer -3.9:

- Peningkatan Dependensi— Meningkatkan beberapa paket Dependensi pihak ketiga.

syn-nodejs-puppeteer-3,8

> **Important**
>
> Versi runtime ini tidak digunakan lagi pada 8 Januari 2024. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Dependensi besar:

- Runtime Lambda Node.js 14.x
- Puppeteer-core versi 5.5.0
- Chromium versi 92.0.4512

Fitur baru di syn-nodejs-puppeteer -3.8:

- Pembersihan profil— Profil Chromium sekarang dibersihkan setelah setiap canary dijalankan.

Perbaikan bug di syn-nodejs-puppeteer -3.8:

- Perbaikan bug— Sebelumnya, canary pemantauan visual terkadang berhenti bekerja dengan benar setelah dijalankan tanpa tangkapan layar. Ini sekarang sudah diperbaiki.

syn-nodejs-puppeteer-3,7

> **Important**
>
> Versi runtime ini tidak digunakan lagi pada 8 Januari 2024. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Dependensi besar:

- Runtime Lambda Node.js 14.x
- Puppeteer-core versi 5.5.0
- Chromium versi 92.0.4512

Fitur baru di syn-nodejs-puppeteer -3.7:

- Peningkatan pencatatan — Canary akan mengunggah log ke Amazon S3 meskipun waktu habis atau macet.
- Ukuran lapisan Lambda berkurang— Ukuran lapisan Lambda yang digunakan untuk canary berkurang sebesar 34%.

Perbaikan bug di syn-nodejs-puppeteer -3.7:

- Perbaikan bug — Font Jepang, Mandarin Sederhana, dan Mandarin Tradisional akan di-render dengan benar.

syn-nodejs-puppeteer-3,6

> **Important**
>
> Versi runtime ini tidak digunakan lagi pada 8 Januari 2024. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Dependensi besar:

- Runtime Lambda Node.js 14.x
- Puppeteer-core versi 5.5.0
- Chromium versi 92.0.4512

Fitur baru di syn-nodejs-puppeteer -3.6:

- Stempel waktu yang lebih presisi— Waktu mulai dan waktu berhenti canary sekarang presisi hingga milidetik.

syn-nodejs-puppeteer-3,5

> **Important**
>
> Versi runtime ini tidak digunakan lagi pada 8 Januari 2024. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Dependensi besar:

- Runtime Lambda Node.js 14.x
- Puppeteer-core versi 5.5.0
- Chromium versi 92.0.4512

Fitur baru di syn-nodejs-puppeteer -3.5:

- Dependensi yang diperbarui— Satu-satunya fitur baru dalam runtime ini adalah dependensi yang diperbarui.

syn-nodejs-puppeteer-3,4

> **Important**
>
> Versi runtime ini telah diusangkan pada 13 November 2022. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Dependensi besar:

- Runtime Lambda Node.js 12.x
- Puppeteer-core versi 5.5.0
- Chromium versi 88.0.4298.0

Fitur baru di syn-nodejs-puppeteer -3.4:

- Fungsi handler kustom— Anda sekarang dapat menggunakan fungsi handler kustom untuk skrip canary Anda. Runtime sebelumnya mengharuskan titik masuk skrip untuk menyertakan `.handler`.

  Anda juga dapat menempatkan skrip canary di folder apa pun dan meneruskan nama folder sebagai bagian dari handler. Misalnya, `MyFolder/MyScriptFile.functionname` dapat digunakan sebagai titik masuk.
- Informasi file HAR yang diperluas— Anda sekarang dapat melihat permintaan yang buruk, tertunda, dan tidak lengkap dalam file HAR yang dihasilkan oleh canary.

syn-nodejs-puppeteer-3,3

> **Important**
>
> Versi runtime ini telah diusangkan pada 13 November 2022. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Dependensi besar:

- Runtime Lambda Node.js 12.x
- Puppeteer-core versi 5.5.0

- Chromium versi 88.0.4298.0

Fitur baru di syn-nodejs-puppeteer -3.3:

- Opsi lainnya untuk enkripsi artefak — Untuk kenari yang menggunakan runtime ini atau yang lebih baru, alih-alih menggunakan kunci AWS terkelola untuk mengenkripsi artefak yang disimpan kenari di Amazon S3, Anda dapat memilih untuk menggunakan kunci yang dikelola pelanggan atau kunci AWS KMS yang dikelola Amazon S3. Untuk informasi selengkapnya, lihat Mengenkripsi artefak canary.

syn-nodejs-puppeteer-3,2

> **Important**
>
> Versi runtime ini telah diusangkan pada 13 November 2022. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Dependensi besar:

- Runtime Lambda Node.js 12.x
- Puppeteer-core versi 5.5.0
- Chromium versi 88.0.4298.0

Fitur baru di syn-nodejs-puppeteer -3.2:

- pemantauan visual dengan tangkapan layar— Canary yang menggunakan runtime ini atau yang lebih baru dapat membandingkan tangkapan layar yang diambil saat dijalankan dengan versi dasar dari tangkapan layar yang sama. Jika tangkapan layar lebih berbeda dari ambang persentase yang ditentukan, canary gagal. Untuk informasi selengkapnya, silakan lihat Pemantauan visual atau Cetak biru pemantauan visual.
- Fungsi baru terkait data sensitif Anda dapat mencegah data sensitif muncul di log dan laporan canary. Untuk informasi selengkapnya, lihat SyntheticsLogHelper kelas.
- Fungsi usang Kelas `RequestResponseLogHelper` ini tidak digunakan lagi demi opsi konfigurasi baru lainnya. Untuk informasi selengkapnya, lihat RequestResponseLogHelper kelas.

syn-nodejs-puppeteer-3.1

> **Important**
>
> Versi runtime ini telah diusangkan pada 13 November 2022. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Dependensi besar:

- Runtime Lambda Node.js 12.x
- Puppeteer-core versi 5.5.0
- Chromium versi 88.0.4298.0

Fitur baru di syn-nodejs-puppeteer -3.1:

- Kemampuan untuk mengonfigurasi CloudWatch metrik — Dengan runtime ini, Anda dapat menonaktifkan metrik yang tidak Anda perlukan. Jika tidak, kenari menerbitkan berbagai CloudWatch metrik untuk setiap lari kenari.
- Penautan tangkapan layar— Anda dapat menautkan tangkapan layar ke langkah canary setelah langkah tersebut selesai. Untuk melakukan hal itu, Anda mengambil tangkapan layar dengan menggunakan metode takeScreenshot, menggunakan nama langkah yang ingin Anda kaitkan dengan tangkapan layar. Misalnya, Anda mungkin ingin melakukan langkah, menambahkan waktu tunggu, dan kemudian mengambil tangkapan layar.
- Cetak biru monitor detak jantung dapat memantau beberapa URL — Anda dapat menggunakan cetak biru pemantauan detak jantung di CloudWatch konsol untuk memantau beberapa URL dan melihat status, durasi, tangkapan layar terkait, dan alasan kegagalan untuk setiap URL dalam ringkasan langkah laporan canary run.

syn-nodejs-puppeteer-3,0

> **Important**
>
> Versi runtime ini telah diusangkan pada 13 November 2022. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Dependensi besar:

- Runtime Lambda Node.js 12.x
- Puppeteer-core versi 5.5.0
- Chromium versi 88.0.4298.0

Fitur baru di syn-nodejs-puppeteer -3.0:

- Dependensi yang ditingkatkan— Versi ini menggunakan Puppeteer versi 5.5.0, Node.js 12.x, dan Chromium 88.0.4298.0.
- Akses bucket lintas Wilayah— Anda sekarang dapat menentukan sebuah bucket S3 di Wilayah lain sebagai bucket di mana canary Anda menyimpan file log, tangkapan layar, dan file HAR.
- Fungsi baru yang tersedia— Versi ini menambahkan fungsi pustaka untuk mengambil nama canary dan versi runtime Synthetics.

  Untuk informasi selengkapnya, lihat Kelas Synthetics.

syn-nodejs-2.2

Bagian ini berisi informasi tentang versi runtime `syn-nodejs-2.2`.

> **Important**
>
> Versi runtime ini telah diusangkan pada 28 Mei 2021. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Dependensi besar:

- Runtime Lambda Node.js 10.x
- Puppeteer-core versi 3.3.0
- Chromium versi 83.0.4103.0

Fitur baru di syn-nodejs-2.2:

- Pantau canary Anda sebagai langkah HTTP— Anda sekarang dapat menguji beberapa API dalam sebuah canary tunggal. Setiap API diuji sebagai langkah HTTP terpisah, dan CloudWatch

Synthetics memantau status setiap langkah menggunakan metrik langkah dan laporan langkah Synthetics CloudWatch . CloudWatch Synthetics membuat `SuccessPercent` dan `Duration` metrik untuk setiap langkah HTTP.

Fungsionalitas ini diimplementasikan oleh fungsi executeHttpStep(stepName, requestOptions, callback, stepConfig). Untuk informasi selengkapnya, lihat executeHttpStep(StepName, requestOptions, [callback], [stepConfig]).

Cetak biru canary API diperbarui untuk menggunakan fitur baru ini.

- Pelaporan permintaan HTTP— Anda sekarang dapat melihat laporan permintaan HTTP rinci yang menangkap detail seperti header permintaan/tanggapan, bodi respons, kode status, waktu kesalahan dan performa, waktu koneksi TCP, waktu handshake TLS, waktu byte pertama, dan waktu transfer konten. Semua permintaan HTTP yang menggunakan modul HTTP/HTTPS di bawah hood ditangkap di sini. Header dan bodi respons tidak ditangkap secara default tetapi dapat diaktifkan dengan mengatur opsi konfigurasi.
- Konfigurasi tingkat global dan langkah - Anda dapat mengatur konfigurasi CloudWatch Synthetics di tingkat global, yang diterapkan ke semua langkah kenari. Anda juga dapat mengganti konfigurasi ini pada tingkat langkah dengan melewati pasangan kunci/nilai konfigurasi untuk mengaktifkan atau menonaktifkan opsi tertentu.

  Untuk informasi selengkapnya, lihat SyntheticsConfiguration kelas.
- Lanjutkan pada konfigurasi kegagalan langkah— Anda dapat memilih untuk melanjutkan eksekusi canary ketika langkah gagal. Untuk fungsi `executeHttpStep`, ini diaktifkan secara default. Anda dapat mengatur opsi ini sekali di tingkat global atau mengaturnya berbeda-beda per-langkah.

syn-nodejs-2.1

> **Important**
>
> Versi runtime ini telah diusangkan pada 28 Mei 2021. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Dependensi besar:

- Runtime Lambda Node.js 10.x
- Puppeteer-core versi 3.3.0

- Chromium versi 83.0.4103.0

Fitur baru di syn-nodejs-2.1:

- Perilaku tangkapan layar yang dapat dikonfigurasi— Menyediakan kemampuan untuk mematikan tangkapan layar oleh canary UI. Dalam canary yang menggunakan versi runtime sebelumnya, canary UI selalu menangkap tangkapan layar sebelum dan setelah setiap langkah. Dengan `syn-nodejs-2.1`, ini dapat dikonfigurasi. Mematikan tangkapan layar dapat mengurangi biaya penyimpanan Amazon S3, dan dapat membantu Anda mematuhi peraturan HIPAA. Untuk informasi selengkapnya, lihat SyntheticsConfiguration kelas.
- Kustomisasi parameter peluncuran Google Chrome Sekarang, Anda dapat mengonfigurasi argumen yang digunakan ketika canary meluncurkan jendela browser Google Chrome. Untuk informasi selengkapnya, lihat launch(options).

Mungkin ada peningkatan kecil durasi canary ketika menggunakan syn-nodejs-2.0 atau lebih baru, dibandingkan dengan versi sebelumnya dari runtime canary.

syn-nodejs-2.0

> **Important**
>
> Versi runtime ini telah diusangkan pada 28 Mei 2021. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Dependensi besar:

- Runtime Lambda Node.js 10.x
- Puppeteer-core versi 3.3.0
- Chromium versi 83.0.4103.0

Fitur baru di syn-nodejs-2.0:

- Dependensi yang ditingkatkan— Versi runtime ini menggunakan versi inti Puppeteer 3.3.0 dan versi Chromium 83.0.4103.0
- Dukungan untuk penelusuran aktif X-Ray. Ketika kenari mengaktifkan penelusuran, jejak X-Ray dikirim untuk semua panggilan yang dilakukan oleh kenari yang menggunakan browser, AWS

SDK, atau modul HTTP atau HTTPS. Canary dengan pelacakan yang diaktifkan muncul di X-Ray Trace Map, meskipun tidak mengirim permintaan ke layanan atau aplikasi lain yang mengaktifkan pelacakan. Untuk informasi selengkapnya, lihat Penelusuran Canary dan X-Ray.

- Pelaporan Synthetics — Untuk setiap kenari yang dijalankan, CloudWatch Synthetics membuat laporan bernama `SyntheticsReport-PASSED.json` atau `SyntheticsReport-FAILED.json` yang mencatat data seperti waktu mulai, waktu akhir, status, dan kegagalan. Pelaporan ini juga mencatat status LULUS/GAGAL dari setiap langkah skrip canary, serta kegagalan dan tangkapan layar yang diambil untuk setiap langkah.
- Laporan pemeriksa tautan terputus— Versi baru dari pemeriksa tautan yang rusak yang disertakan dalam runtime membuat laporan yang menyertakan tautan yang diperiksa, kode status, alasan kegagalan (jika ada), dan tangkapan layar halaman sumber dan tujuan.
- CloudWatch Metrik baru — Synthetics menerbitkan metrik `2xx` bernama`4xx`,,`5xx`, `RequestFailed` dan di namespace. `CloudWatchSynthetics` Metrik ini menunjukkan jumlah 200 detik, 400 detik, 500 detik, dan meminta kegagalan dalam operasi canary. Dengan versi runtime ini, metrik ini dilaporkan hanya untuk canary UI, dan tidak dilaporkan untuk canary API. Metrik juga dilaporkan untuk API canary yang dimulai dengan versi runtime `syn-nodejs-puppeteeer-2.2`.
- File HAR yang dapat disortir— Sekarang Anda dapat menyortir file HAR Anda berdasarkan kode status, ukuran permintaan, dan durasi.
- Stempel waktu metrik — CloudWatch metrik sekarang dilaporkan berdasarkan waktu pemanggilan Lambda, bukan waktu akhir kenari.

Perbaikan bug pada syn-nodejs-2.0:

- Memperbaiki masalah kesalahan pengunggahan artefak canary tidak dilaporkan. Kesalahan tersebut sekarang muncul sebagai kesalahan eksekusi.
- Memperbaiki masalah permintaan pengalihan (3xx) yang salah dicatat sebagai kesalahan.
- Memperbaiki masalah tangkapan layar yang sedang diberi nomor mulai dari 0. Mereka sekarang harus dimulai dengan 1.
- Memperbaiki masalah tangkapan layar yang berantakan untuk huruf Mandarin dan Jepang.

Mungkin ada peningkatan kecil durasi canary ketika menggunakan syn-nodejs-2.0 atau lebih baru, dibandingkan dengan versi sebelumnya dari runtime canary.

syn-nodejs-2.0-beta

> **Important**
>
> Versi runtime ini telah diusangkan pada 8 Februari 2021. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Dependensi besar:

- Runtime Lambda Node.js 10.x
- Puppeteer-core versi 3.3.0
- Chromium versi 83.0.4103.0

Fitur baru di syn-nodejs-2.0-beta:

- Dependensi yang ditingkatkan— Versi runtime ini menggunakan versi inti Puppeteer 3.3.0 dan versi Chromium 83.0.4103.0
- Pelaporan Synthetics — Untuk setiap kenari yang dijalankan, CloudWatch Synthetics membuat laporan bernama `SyntheticsReport-PASSED.json` atau `SyntheticsReport-FAILED.json` yang mencatat data seperti waktu mulai, waktu akhir, status, dan kegagalan. Pelaporan ini juga mencatat status LULUS/GAGAL dari setiap langkah skrip canary, serta kegagalan dan tangkapan layar yang diambil untuk setiap langkah.
- Laporan pemeriksa tautan terputus— Versi baru dari pemeriksa tautan yang rusak yang disertakan dalam runtime membuat laporan yang menyertakan tautan yang diperiksa, kode status, alasan kegagalan (jika ada), dan tangkapan layar halaman sumber dan tujuan.
- CloudWatch Metrik baru — Synthetics menerbitkan metrik `2xx` bernama`4xx`,,`5xx`, `RequestFailed` dan di namespace. `CloudWatchSynthetics` Metrik ini menunjukkan jumlah 200 detik, 400 detik, 500 detik, dan meminta kegagalan dalam operasi canary. Metrik ini dilaporkan hanya untuk canary UI, dan tidak dilaporkan untuk canary API.
- File HAR yang dapat disortir— Sekarang Anda dapat menyortir file HAR Anda berdasarkan kode status, ukuran permintaan, dan durasi.
- Stempel waktu metrik — CloudWatch metrik sekarang dilaporkan berdasarkan waktu pemanggilan Lambda, bukan waktu akhir kenari.

Perbaikan bug dalam syn-nodejs-2.0-beta:

- Memperbaiki masalah kesalahan pengunggahan artefak canary tidak dilaporkan. Kesalahan tersebut sekarang muncul sebagai kesalahan eksekusi.
- Memperbaiki masalah permintaan pengalihan (3xx) yang salah dicatat sebagai kesalahan.
- Memperbaiki masalah tangkapan layar yang sedang diberi nomor mulai dari 0. Mereka sekarang harus dimulai dengan 1.
- Memperbaiki masalah tangkapan layar yang berantakan untuk huruf Mandarin dan Jepang.

syn-1.0

> **Important**
>
> Versi runtime ini dijadwalkan akan diusangkan pada 28 Mei 2021. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Versi runtime Synthetic pertama adalah `syn-1.0`.

Dependensi besar:

- Runtime Lambda Node.js 10.x
- Puppeteer-core versi 1.14.0
- Versi Chromium yang sesuai dengan Puppeteer-core 1.14.0

Versi runtime yang menggunakan Python dan Selenium Webdriver

Bagian berikut berisi informasi tentang versi runtime CloudWatch Synthetics untuk Python dan Selenium Webdriver. Selenium adalah alat otomatisasi browser sumber terbuka. Untuk informasi selengkapnya tentang Selenium, silakan lihat www.selenium.dev/

Konvensi penamaan untuk versi runtime ini adalah `syn-`*`language`*`-`*`framework`*`-`*`majorversion`*`.`*`minorversion`*.

> **Important**
>
> Versi runtime CloudWatch Synthetics berikut dijadwalkan tidak digunakan lagi pada 8 Maret 2024.

- `syn-python-selenium-2.0`
- `syn-python-selenium-1.3`
- `syn-python-selenium-1.2`
- `syn-python-selenium-1.1`
- `syn-python-selenium-1.0`

Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

syn-python-selenium-3,0

Versi 3.0 adalah runtime CloudWatch Synthetics terbaru untuk Python dan Selenium.

Dependensi besar:

- Python 3.8
- Selenium 4.15.1
- Chromium versi 121.0.6167.139

Fitur baru di syn-python-selenium -3.0:

- Versi terbaru dari pustaka yang dibundel di Chromium — Ketergantungan Chromium diperbarui ke versi baru.

syn-python-selenium-2.1

Dependensi besar:

- Python 3.8
- Selenium 4.15.1
- Chromium versi 111.0.5563.146

Fitur baru di syn-python-selenium -2.1:

- Versi terbaru dari pustaka yang dibundel di Chromium - Dependensi Chromium dan Selenium diperbarui ke versi baru.

syn-python-selenium-2,0

> **Important**
>
> Versi runtime ini dijadwalkan tidak digunakan lagi pada 8 Maret 2024. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Dependensi besar:

- Python 3.8
- Selenium 4.10.0
- Chromium versi 111.0.5563.146

Fitur baru di syn-python-selenium -2.0:

- Dependensi yang diperbarui— Dependensi Chromium dan Selenium diperbarui ke versi baru.

Perbaikan bug di syn-python-selenium -2.0:

- Stempel waktu ditambahkan — Sebuah stempel waktu telah ditambahkan ke log canary.
- Penggunaan kembali sesi — Bug telah diperbaiki sehingga canary sekarang dicegah untuk menggunakan kembali sesi dari canary sebelumnya.

syn-python-selenium-1,3

> **Important**
>
> Versi runtime ini dijadwalkan tidak digunakan lagi pada 8 Maret 2024. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Dependensi besar:

- Python 3.8
- Selenium 3.141.0
- Chromium versi 92.0.4512.0

Fitur baru di syn-python-selenium -1.3:

- Stempel waktu yang lebih presisi— Waktu mulai dan waktu berhenti canary sekarang presisi hingga milidetik.

syn-python-selenium-1,2

> **Important**
>
> Versi runtime ini dijadwalkan tidak digunakan lagi pada 8 Maret 2024. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Dependensi besar:

- Python 3.8
- Selenium 3.141.0
- Chromium versi 92.0.4512.0

- Dependensi yang diperbarui— Satu-satunya fitur baru dalam runtime ini adalah dependensi yang diperbarui.

syn-python-selenium-1.1

> **Important**
>
> Versi runtime ini dijadwalkan tidak digunakan lagi pada 8 Maret 2024. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Dependensi besar:

- Python 3.8
- Selenium 3.141.0
- Chromium versi 83.0.4103.0

Fitur-fitur:

- Fungsi handler kustom— Anda sekarang dapat menggunakan fungsi handler kustom untuk skrip canary Anda. Runtime sebelumnya mengharuskan titik masuk skrip untuk menyertakan `.handler`.

  Anda juga dapat menempatkan skrip canary di folder apa pun dan meneruskan nama folder sebagai bagian dari handler. Misalnya, `MyFolder/MyScriptFile.functionname` dapat digunakan sebagai titik masuk.
- Opsi konfigurasi untuk menambahkan metrik dan konfigurasi kegagalan langkah— Opsi ini sudah tersedia di runtime untuk canary Node.js. Untuk informasi selengkapnya, lihat SyntheticsConfiguration kelas.
- Argumen kustom di Chrome — Anda sekarang dapat membuka browser dalam mode penyamaran atau meneruskan konfigurasi server proxy. Untuk informasi selengkapnya, lihat Chrome().
- Bucket artefak Lintas Wilayah — Sebuah canary dapat menyimpan artefaknya di bucket Amazon S3 di Wilayah yang berbeda.
- Perbaikan bug, termasuk perbaikan untuk masalah **`index.py`** — Dengan runtime sebelumnya, file canary bernama `index.py` menyebabkan pengecualian karena bertentangan dengan nama file pustaka. Masalah ini sekarang telah diperbaiki.

syn-python-selenium-1,0

> **Important**
>
> Versi runtime ini dijadwalkan tidak digunakan lagi pada 8 Maret 2024. Untuk informasi selengkapnya, lihat CloudWatch Kebijakan dukungan runtime Synthetics.

Dependensi besar:

- Python 3.8
- Selenium 3.141.0
- Chromium versi 83.0.4103.0

Fitur-fitur:

- Dukungan selenium— Anda dapat menulis skrip canary menggunakan kerangka kerja uji Selenium. Anda dapat membawa skrip Selenium Anda dari tempat lain ke CloudWatch Synthetics dengan sedikit perubahan, dan mereka akan bekerja dengan layanan. AWS

Menulis skrip canary

Bagian berikut menjelaskan cara menulis skrip kenari dan cara mengintegrasikan kenari dengan AWS layanan lain dan dengan dependensi dan pustaka eksternal.

Topik

- Menulis skrip canary Node.js
- Menulis skrip canary Python
- Mengubah skrip Selenium yang sudah ada untuk menggunakan canary Synthetics
- Mengubah skrip Puppeteer Synthetics yang ada untuk mengautentikasi sertifikat non-standar

Menulis skrip canary Node.js

Topik

- Membuat kenari CloudWatch Synthetics dari awal
- Mengemas file kenari Node.js Anda
- Mengubah skrip Puppeteer yang sudah ada untuk digunakan sebagai canary Synthetic
- Variabel-variabel lingkungan
- Mengintegrasikan kenari Anda dengan layanan lain AWS
- Menyempurnakan canary Anda untuk menggunakan alamat IP statis

Membuat kenari CloudWatch Synthetics dari awal

Berikut adalah contoh skrip Canary Synthetic minimal. skrip ini berhasil dijalani, dan mengembalikan rangkaian. Untuk melihat bentuk canary yang gagal, ubah `let fail = false;` untuk `let fail = true;`.

Anda harus menetapkan fungsi titik masuk untuk skrip canary. Untuk melihat cara mengunggah file ke lokasi Amazon S3 yang ditentukan sebagai `ArtifactS3Location` canary, buat file ini di bawah folder /tmp. Setelah skrip berjalan, status lulus/gagal dan metrik durasi dipublikasikan ke CloudWatch dan file di bawah/tmp diunggah ke S3.

```
const basicCustomEntryPoint = async function () {

    // Insert your code here
```

```
    // Perform multi-step pass/fail check

    // Log decisions made and results to /tmp

    // Be sure to wait for all your code paths to complete
    // before returning control back to Synthetics.
    // In that way, your canary will not finish and report success
    // before your code has finished executing

    // Throw to fail, return to succeed
    let fail = false;
    if (fail) {
        throw "Failed basicCanary check.";
    }

    return "Successfully completed basicCanary checks.";
};

exports.handler = async () => {
    return await basicCustomEntryPoint();
};
```

Selanjutnya, kita akan memperluas script untuk menggunakan Synthetics logging dan membuat panggilan menggunakan SDK AWS . Untuk tujuan demonstrasi, skrip ini akan menciptakan klien Amazon DynamoDB dan memanggil API listTables DynamoDB. skrip ini mencatat respons terhadap permintaan dan log akan lulus atau gagal bergantung pada permintaan berhasil atau tidak.

```
const log = require('SyntheticsLogger');
const AWS = require('aws-sdk');
// Require any dependencies that your script needs
// Bundle additional files and dependencies into a .zip file with folder structure
// nodejs/node_modules/additional files and folders

const basicCustomEntryPoint = async function () {

    log.info("Starting DynamoDB:listTables canary.");

    let dynamodb = new AWS.DynamoDB();
    var params = {};
    let request = await dynamodb.listTables(params);
    try {
        let response = await request.promise();
```

```
        log.info("listTables response: " + JSON.stringify(response));
    } catch (err) {
        log.error("listTables error: " + JSON.stringify(err), err.stack);
        throw err;
    }

    return "Successfully completed DynamoDB:listTables canary.";
};

exports.handler = async () => {
    return await basicCustomEntryPoint();
};
```

Mengemas file kenari Node.js Anda

Jika Anda mengunggah skrip canary Anda menggunakan lokasi Amazon S3, file zip Anda harus menyertakan skrip Anda di bawah struktur folder ini: `nodejs/node_modules/`*`myCanaryFilename.js file`*.

Jika Anda memiliki lebih dari satu file `.js` tunggal atau Anda memiliki dependensi yang bergantung pada skrip, Anda dapat menggabungkan semuanya ke dalam satu paketan file ZIP yang memuat struktur folder `nodejs/node_modules/`*`myCanaryFilename.js file and other folders and files`*. Jika Anda menggunakan `syn-nodejs-puppeteer-3.4` atau yang lebih baru, Anda dapat secara opsional meletakkan file canary Anda di folder lain dan membuat struktur folder Anda seperti ini: `nodejs/node_modules/`*`myFolder`*`/`*`myCanaryFilename.js file and other folders and files`*.

Nama handler

Pastikan untuk mengatur titik masuk skrip canary Anda (handler) `myCanaryFilename.functionName` agar cocok dengan nama file dari titik masuk skrip Anda. Jika Anda menggunakan runtime yang lebih awal dari `syn-nodejs-puppeteer-3.4`, `functionName` harus `handler`. Jika Anda menggunakan `syn-nodejs-puppeteer-3.4` atau yang lebih baru, Anda dapat memilih nama fungsi apa pun sebagai handler. Jika Anda menggunakan `syn-nodejs-puppeteer-3.4` atau lebih baru, Anda juga dapat secara opsional menyimpan canary di folder terpisah seperti `nodejs/node_modules/myFolder/my_canary_filename`. Jika Anda menyimpannya di folder terpisah, tentukan jalur itu di titik entri skrip Anda, seperti `myFolder/my_canary_filename.functionName`.

Mengubah skrip Puppeteer yang sudah ada untuk digunakan sebagai canary Synthetic

Bagian ini menjelaskan cara mengambil skrip Puppeteer dan memodifikasinya untuk berjalan sebagai skrip canary Synthetic. Untuk informasi selengkapnya tentang Puppeteer, silakan lihat API Puppeteer v1.14.0.

Kita akan mulai dengan contoh skrip Puppeteer:

```
const puppeteer = require('puppeteer');

(async () => {
  const browser = await puppeteer.launch();
  const page = await browser.newPage();
  await page.goto('https://example.com');
  await page.screenshot({path: 'example.png'});

  await browser.close();
})();
```

Langkah konversinya adalah sebagai berikut:

- Membuat dan mengekspor `handler` fungsi. Penanggung jawab adalah fungsi titik masuk untuk skrip. Jika Anda menggunakan runtime yang lebih awal dari `syn-nodejs-puppeteer-3.4`, fungsi handler harus diberi nama `handler`. Jika Anda menggunakan `syn-nodejs-puppeteer-3.4` atau lebih baru, fungsi tersebut dapat memiliki nama apa pun, tetapi harus nama yang sama yang digunakan dalam skrip. Juga, jika Anda menggunakan `syn-nodejs-puppeteer-3.4` atau yang lebih baru, Anda akan dapat menyimpan skrip Anda di bawah folder apa pun dan menentukan folder itu sebagai bagian dari nama handler.

  ```
  const basicPuppeteerExample = async function () {};

  exports.handler = async () => {
      return await basicPuppeteerExample();
  };
  ```

- Gunakan dependensi `Synthetics`.

  ```
  var synthetics = require('Synthetics');
  ```

- Gunakan fungsi `Synthetics.getPage` untuk mendapatkan objek `Page` Puppeteer.

```
const page = await synthetics.getPage();
```

Objek halaman yang dikembalikan oleh fungsi Synthetics.getPage memiliki peristiwa page.on `request`, `response` dan `requestfailed` yang terinstrumentasi untuk pembuatan log. Synthetics juga mengatur pembuatan file HAR untuk permintaan dan respons di halaman tersebut, serta menambahkan ARN canary ke header agen-pengguna atas permintaan keluar di halaman tersebut.

skripnya sekarang siap untuk dijalankan sebagai canary Synthetics. Berikut adalah skrip yang diperbarui:

```
var synthetics = require('Synthetics');  // Synthetics dependency

const basicPuppeteerExample = async function () {
    const page = await synthetics.getPage(); // Get instrumented page from Synthetics
    await page.goto('https://example.com');
    await page.screenshot({path: '/tmp/example.png'}); // Write screenshot to /tmp
 folder
};

exports.handler = async () => {  // Exported handler function
    return await basicPuppeteerExample();
};
```

Variabel-variabel lingkungan

Anda dapat menggunakan variabel lingkungan saat membuat canary. Hal ini memungkinkan Anda untuk menulis skrip canary tunggal dan kemudian menggunakan skrip tersebut dengan nilai yang berbeda untuk dengan cepat membuat beberapa canary yang memiliki tugas serupa.

Misalnya, andaikan organisasi Anda memiliki titik akhir seperti `prod`, `dev`, dan `pre-release` untuk berbagai tahap pengembangan perangkat lunak Anda, dan Anda perlu membuat canary untuk menguji setiap titik akhir ini. Anda dapat menulis skrip canary tunggal yang menguji perangkat lunak Anda dan kemudian menentukan nilai yang berbeda untuk variabel lingkungan titik akhir ketika Anda membuat masing-masing dari tiga canary. Kemudian, ketika Anda membuat canary, Anda menentukan skrip dan nilai-nilai yang akan digunakan untuk variabel lingkungan.

Nama-nama variabel lingkungan dapat memuat huruf, angka, dan karakter garis bawah. Nama variavel harus dimulai dengan sebuah huruf dan setidaknya dua karakter. Ukuran total variabel

lingkungan Anda tidak dapat lebih dari 4 KB. Anda tidak dapat menentukan variabel lingkungan yang dicadangkan Lambda sebagai nama variabel lingkungan Anda. Untuk informasi selengkapnya tentang variabel lingkungan yang dicadangkan, silakan lihat Variabel lingkungan runtime.

> **Important**
>
> Kunci dan nilai variabel lingkungan tidak dienkripsi. Jangan menyimpan informasi sensitif di dalamnya.

Contoh skrip berikut menggunakan dua variabel lingkungan. Skrip ini adalah untuk canary yang memeriksa apakah sebuah halaman web tersedia. Ini menggunakan variabel lingkungan untuk membuat parameter URL yang diperiksa dan tingkat log CloudWatch Synthetics yang digunakannya.

Fungsi berikut menetapkan `LogLevel` ke nilai variabel lingkungan `LOG_LEVEL`.

```
synthetics.setLogLevel(process.env.LOG_LEVEL);
```

Fungsi ini menetapkan URL ke nilai variabel lingkungan URL.

```
const URL = process.env.URL;
```

Ini adalah skrip yang lengkap. Ketika Anda membuat canary menggunakan skrip ini, Anda menentukan nilai untuk variabel lingkungan `LOG_LEVEL` dan URL.

```
var synthetics = require('Synthetics');
const log = require('SyntheticsLogger');

const pageLoadEnvironmentVariable = async function () {

    // Setting the log level (0-3)
    synthetics.setLogLevel(process.env.LOG_LEVEL);
    // INSERT URL here
    const URL = process.env.URL;

    let page = await synthetics.getPage();
    //You can customize the wait condition here. For instance,
    //using 'networkidle2' may be less restrictive.
    const response = await page.goto(URL, {waitUntil: 'domcontentloaded', timeout:
 30000});
    if (!response) {
```

```
        throw "Failed to load page!";
    }
    //Wait for page to render.
    //Increase or decrease wait time based on endpoint being monitored.
    await page.waitFor(15000);
    await synthetics.takeScreenshot('loaded', 'loaded');
    let pageTitle = await page.title();
    log.info('Page title: ' + pageTitle);
    log.debug('Environment variable:' + process.env.URL);

    //If the response status code is not a 2xx success code
    if (response.status() < 200 || response.status() > 299) {
        throw "Failed to load page!";
    }
};

exports.handler = async () => {
    return await pageLoadEnvironmentVariable();
};
```

Meneruskan variabel lingkungan ke script Anda

Untuk meneruskan variabel lingkungan ke skrip Anda ketika Anda membuat canary di konsol, tentukan kunci dan nilai-nilai variabel lingkungan di bagian Variabel lingkungan pada konsol. Untuk informasi selengkapnya, lihat Membuat canary.

Untuk meneruskan variabel lingkungan melalui API atau AWS CLI, gunakan `EnvironmentVariables` parameter di `RunConfig` bagian. Berikut ini adalah contoh AWS CLI perintah yang menciptakan kenari yang menggunakan dua variabel lingkungan dengan kunci `Environment` dan`Region`.

```
aws synthetics create-canary --cli-input-json '{
   "Name":"nameofCanary",
   "ExecutionRoleArn":"roleArn",
   "ArtifactS3Location":"s3://cw-syn-results-123456789012-us-west-2",
   "Schedule":{
      "Expression":"rate(0 minute)",
      "DurationInSeconds":604800
   },
   "Code":{
      "S3Bucket": "canarycreation",
      "S3Key": "cwsyn-mycanaryheartbeat-12345678-d1bd-1234-
abcd-123456789012-12345678-6a1f-47c3-b291-123456789012.zip",
```

```
      "Handler":"pageLoadBlueprint.handler"
   },
   "RunConfig": {
      "TimeoutInSeconds":60,
      "EnvironmentVariables": {
         "Environment":"Production",
         "Region": "us-west-1"
      }
   },
   "SuccessRetentionPeriodInDays":13,
   "FailureRetentionPeriodInDays":13,
   "RuntimeVersion":"syn-nodejs-2.0"
}'
```

Mengintegrasikan kenari Anda dengan layanan lain AWS

Semua kenari dapat menggunakan pustaka AWS SDK. Anda dapat menggunakan perpustakaan ini ketika Anda menulis kenari Anda untuk mengintegrasikan kenari dengan layanan lain AWS .

Untuk melakukan hal itu, Anda perlu menambahkan kode berikut ke canary. Untuk contoh-contoh AWS Secrets Manager ini, digunakan sebagai layanan yang terintegrasi dengan kenari.

- Impor AWS SDK.

```
const AWS = require('aws-sdk');
```

- Buat klien untuk AWS layanan yang Anda integrasikan.

```
const secretsManager = new AWS.SecretsManager();
```

- Gunakan klien untuk melakukan panggilan API ke layanan tersebut.

```
var params = {
  SecretId: secretName
};
return await secretsManager.getSecretValue(params).promise();
```

Kode skrip snippet canary berikut menunjukkan contoh integrasi dengan Secrets Manager secara lebih terperinci.

```
var synthetics = require('Synthetics');
```

```
const log = require('SyntheticsLogger');

const AWS = require('aws-sdk');
const secretsManager = new AWS.SecretsManager();

const getSecrets = async (secretName) => {
    var params = {
        SecretId: secretName
    };
    return await secretsManager.getSecretValue(params).promise();
}

const secretsExample = async function () {
    let URL = "<URL>";
    let page = await synthetics.getPage();

    log.info(`Navigating to URL: ${URL}`);
    const response = await page.goto(URL, {waitUntil: 'domcontentloaded', timeout:
 30000});

    // Fetch secrets
    let secrets = await getSecrets("secretname")

    /**
    * Use secrets to login.
    *
    * Assuming secrets are stored in a JSON format like:
    * {
    *   "username": "<USERNAME>",
    *   "password": "<PASSWORD>"
    * }
    **/
    let secretsObj = JSON.parse(secrets.SecretString);
    await synthetics.executeStep('login', async function () {
        await page.type(">USERNAME-INPUT-SELECTOR<", secretsObj.username);
        await page.type(">PASSWORD-INPUT-SELECTOR<", secretsObj.password);

        await Promise.all([
          page.waitForNavigation({ timeout: 30000 }),
          await page.click(">SUBMIT-BUTTON-SELECTOR<")
        ]);
    });

    // Verify login was successful
```

```
    await synthetics.executeStep('verify', async function () {
        await page.waitForXPath(">SELECTOR<", { timeout: 30000 });
    });
};

exports.handler = async () => {
    return await secretsExample();
};
```

Menyempurnakan canary Anda untuk menggunakan alamat IP statis

Anda dapat menyiapkan canary sehingga menggunakan alamat IP statis.

Untuk memaksa canary menggunakan alamat IP statis

1. Buat VPC baru. Untuk informasi selengkapnya, silakan lihat Menggunakan DNS dengan VPC Anda.
2. Membuat gateway internet baru. Untuk informasi selengkapnya, silakan lihat Menambahkan gateway internet ke VPC Anda.
3. Buat subnet publik di dalam VPC Anda yang baru.
4. Tambahkan tabel rute baru ke VPC.
5. Tambahkan rute di tabel rute baru, yang dimulai dari `0.0.0.0/0` ke gateway internet.
6. Kaitkan tabel rute baru dengan subnet publik.
7. Buat alamat IP elastis. Untuk informasi selengkapnya, silakan lihat Alamat IP elastis.
8. Buat gateway NAT baru dan tetapkan ke subnet publik dan alamat IP elastis.
9. Buat subnet privat di dalam VPC.
10. Tambahkan rute ke tabel rute bawaan VPC, yang dimulai dari `0.0.0.0/0` ke gateway NAT
11. Buat canary Anda.

Menulis skrip canary Python

Skrip ini berhasil dijalani, dan mengembalikan rangkaian. Untuk melihat bentuk canary yang gagal, ubah fail = False ke fail = True

```
def basic_custom_script():
    # Insert your code here
    # Perform multi-step pass/fail check
```

```
    # Log decisions made and results to /tmp
    # Be sure to wait for all your code paths to complete
    # before returning control back to Synthetics.
    # In that way, your canary will not finish and report success
    # before your code has finished executing
    fail = False
    if fail:
        raise Exception("Failed basicCanary check.")
    return "Successfully completed basicCanary checks."
def handler(event, context):
    return basic_custom_script()
```

Mengemas file kenari Python Anda

Jika Anda memiliki lebih dari satu file .py atau skrip Anda memiliki dependensi, Anda dapat menggabungkan file itu semua ke dalam sebuah paket file ZIP tunggal. Jika Anda menggunakan runtime `syn-python-selenium-1.1`, file ZIP harus berisi file .py canary utama Anda di dalam folder `python`, seperti `python/my_canary_filename.py`. Jika Anda menggunakan `syn-python-selenium-1.1` atau yang lebih baru, Anda dapat menggunakan folder yang berbeda secara opsional, seperti `python/myFolder/my_canary_filename.py`.

File ZIP ini harus memuat semua folder dan file yang diperlukan, tetapi file lain tidak perlu berada dalam folder `python`.

Pastikan untuk mengatur titik masuk skrip canary Anda agar `my_canary_filename.functionName` cocok dengan nama file dan nama fungsi dari titik masuk skrip Anda. Jika Anda menggunakan runtime `syn-python-selenium-1.0`, `functionName` harus `handler`. Jika Anda menggunakan `syn-python-selenium-1.1` atau yang lebih baru, pembatasan nama handler ini tidak berlaku, dan Anda juga dapat secara opsional menyimpan canary di folder terpisah seperti `python/myFolder/my_canary_filename.py`. Jika Anda menyimpannya di folder terpisah, tentukan jalur itu di titik entri skrip Anda, seperti `myFolder/my_canary_filename.functionName`.

Mengubah skrip Selenium yang sudah ada untuk menggunakan canary Synthetics

Anda dapat dengan cepat memodifikasi skrip yang ada untuk Python dan Selenium untuk digunakan sebagai canary. Untuk informasi selengkapnya tentang Selenium, silakan lihat www.selenium.dev/.

Untuk contoh ini, kita akan mulai dengan skrip Selenium berikut:

```
from selenium import webdriver
```

```
def basic_selenium_script():
    browser = webdriver.Chrome()
    browser.get('https://example.com')
    browser.save_screenshot('loaded.png')

basic_selenium_script()
```

Langkah konversinya adalah sebagai berikut.

Untuk mengkonversi skrip Selenium yang akan digunakan sebagai canary

1. Ubah pernyataan `import` untuk menggunakan Selenium dari modul `aws_synthetics`:

   ```
   from aws_synthetics.selenium import synthetics_webdriver as webdriver
   ```

   Modul Selenium dari `aws_synthetics` memastikan bahwa kenari dapat memancarkan metrik dan log, menghasilkan file HAR, dan bekerja dengan fitur Synthetics lainnya. CloudWatch
2. Buat fungsi handler dan panggil metode Selenium Anda. Penanggung jawab adalah fungsi titik masuk untuk skrip.

   Jika Anda menggunakan `syn-python-selenium-1.0`, fungsi handler harus diberi nama `handler`. Jika Anda menggunakan `syn-python-selenium-1.1` atau lebih baru, fungsi tersebut dapat memiliki nama apa pun, tetapi harus nama yang sama yang digunakan dalam skrip. Juga, jika Anda menggunakan `syn-python-selenium-1.1` atau yang lebih baru, Anda akan dapat menyimpan skrip Anda di bawah folder apa pun dan menentukan folder itu sebagai bagian dari nama handler.

   ```
   def handler(event, context):
       basic_selenium_script()
   ```

Skrip sekarang diperbarui menjadi kenari CloudWatch Synthetics. Berikut adalah skrip yang diperbarui:

```
from aws_synthetics.selenium import synthetics_webdriver as webdriver

def basic_selenium_script():
    browser = webdriver.Chrome()
    browser.get('https://example.com')
```

```
    browser.save_screenshot('loaded.png')

def handler(event, context):
    basic_selenium_script()
```

Mengubah skrip Puppeteer Synthetics yang ada untuk mengautentikasi sertifikat non-standar

Salah satu kasus penggunaan penting untuk kenari Synthetics adalah bagi Anda untuk memantau titik akhir Anda sendiri. Jika Anda ingin memantau titik akhir yang tidak siap untuk lalu lintas eksternal, pemantauan ini terkadang dapat berarti bahwa Anda tidak memiliki sertifikat yang tepat yang ditandatangani oleh otoritas sertifikat pihak ketiga tepercaya.

Dua solusi yang mungkin untuk skenario ini adalah sebagai berikut:

- Untuk mengautentikasi sertifikat klien, lihat Cara memvalidasi autentikasi menggunakan Amazon CloudWatch Synthetics — Bagian 2.
- Untuk mengautentikasi sertifikat yang ditandatangani sendiri, lihat Cara memvalidasi autentikasi dengan sertifikat yang ditandatangani sendiri di Amazon Synthetics CloudWatch

Anda tidak terbatas pada dua opsi ini ketika Anda menggunakan kenari CloudWatch Synthetics. Anda dapat memperluas fitur-fitur ini dan menambahkan logika bisnis Anda dengan memperluas kode canary.

> **Note**
>
> Synthetics canary yang berjalan pada runtime Python secara bawaan mengaktifkan `--ignore-certificate-errors` flag, jadi kenari tersebut seharusnya tidak memiliki masalah dalam menjangkau situs dengan konfigurasi sertifikat non-standar.

Fungsi pustaka tersedia untuk skrip canary

CloudWatch Synthetics mencakup beberapa kelas dan fungsi bawaan yang dapat Anda panggil saat menulis skrip Node.js untuk digunakan sebagai kenari.

Sebagian berlaku untuk canary UI dan API. Lainnya hanya berlaku untuk canary UI. Canary UI adalah canary yang menggunakan fungsi `getPage()` dan menggunakan Puppeteer sebagai pemicu web untuk menavigasi dan berinteraksi dengan halaman web.

> **Note**
>
> Setiap kali Anda meningkatkan canary untuk menggunakan versi baru runtime Synthetics, semua fungsi pustaka Synthetics yang digunakan canary Anda juga secara otomatis ditingkatkan ke versi NodeJS yang sama dengan yang didukung runtime Synthetics.

Topik

Fungsi pustaka tersedia untuk skrip canary Node.js

Bagian ini mencantumkan fungsi pustaka yang tersedia untuk skrip canary Node.js.

Topik

Kelas dan fungsi pustaka Node.js yang berlaku untuk semua canary

Fungsi pustaka CloudWatch Synthetics berikut untuk Node.js berguna untuk semua kenari.

Topik

Kelas Synthetics

Fungsi berikut untuk semua canary berada di kelas Synthetics.

addExecutionError(ErrorMessage, ex);

`errorMessage` menjelaskan kesalahan dan `ex` adalah pengecualian yang ditemui

Anda dapat menggunakan `addExecutionError` untuk mengatur kesalahan eksekusi bagi canary Anda. Itu menggagalkan canary tanpa mengganggu eksekusi skrip. Itu juga tidak memengaruhi metrik `successPercent` Anda.

Anda harus melacak kesalahan sebagai kesalahan eksekusi hanya jika kesalahan itu tidak penting untuk menunjukkan keberhasilan atau kegagalan skrip canary Anda.

Contoh penggunaan `addExecutionError` adalah sebagai berikut. Anda memantau ketersediaan titik akhir Anda dan mengambil tangkapan layar setelah halaman dimuat. Karena kegagalan mengambil tangkapan layar tidak menentukan ketersediaan titik akhir, Anda dapat menangkap kesalahan apa pun yang dihadapi saat mengambil tangkapan layar dan menambahkannya sebagai kesalahan eksekusi. Metrik ketersediaan Anda masih akan menunjukkan bahwa titik akhir telah aktif dan berjalan, namun status canary Anda akan ditandai sebagai gagal. Blok kode sampel berikut menangkap kesalahan tersebut dan menambahkannya sebagai kesalahan eksekusi.

```
try {
        await synthetics.takeScreenshot(stepName, "loaded");
    } catch(ex) {
        synthetics.addExecutionError('Unable to take screenshot ', ex);
    }
```

getCanaryName();

Mengembalikan nama dari canary.

getCanaryArn();

Mengembalikan ARN dari canary.

getCanaryUserAgentString();

Mengembalikan agen pengguna kustom dari canary.

getRuntimeVersion();

Fungsi ini tersedia dalam versi runtime `syn-nodejs-puppeteer-3.0` dan yang lebih baru. Fungsi ini mengembalikan versi runtime Synthetics dari canary. Misalnya,nilai yang dikembalikan bisa menjadi `syn-nodejs-puppeteer-3.0`.

getLogLevel();

Mengambil level log saat ini untuk pustaka Synthetics. Nilai yang mungkin adalah sebagai berikut:

- `0` – Debug
- `1` – Info
- `2` – Peringatan
- `3` – Kesalahan

Contoh:

```
let logLevel = synthetics.getLogLevel();
```

setLogLevel();

Mengatur tingkat log untuk pustaka Sintetis. Nilai yang mungkin adalah sebagai berikut:

- `0` – Debug
- `1` – Info
- `2` – Peringatan
- `3` – Kesalahan

Contoh:

```
synthetics.setLogLevel(0);
```

SyntheticsConfiguration kelas

Kelas ini hanya tersedia di versi runtime `syn-nodejs-2.1` atau lebih baru.

Anda dapat menggunakan SyntheticsConfiguration kelas untuk mengkonfigurasi perilaku fungsi perpustakaan Synthetics. Misalnya, Anda dapat menggunakan kelas ini untuk mengonfigurasi fungsi `executeStep()` untuk tidak menangkap tangkapan layar.

Anda dapat mengatur konfigurasi CloudWatch Synthetics di tingkat global, yang diterapkan ke semua langkah kenari. Anda juga dapat mengganti konfigurasi ini pada tingkat langkah dengan meneruskan pasangan kunci/nilai konfigurasi.

Anda dapat memberikan opsi di tingkat langkah. Sebagai contoh, silakan lihat async executeStep (StepName,, [StepConfig]); functionToExecute dan executeHttpStep(StepName, requestOptions, [callback], [stepConfig])

Definisi fungsi:

setConfig(options)

`options` adalah sebuah objek, yang merupakan kumpulan opsi yang dapat dikonfigurasi untuk canary Anda. Bagian berikut menjelaskan bidang-bidang yang memungkinkan dalam *`options`*.

setConfig(options) untuk semua canary

Untuk canary yang menggunakan `syn-nodejs-puppeteer-3.2` atau yang lebih baru, (opsi) untuk setConfig dapat menyertakan parameter-parameter berikut:

- `includeRequestHeaders` (boolean)— Apakah akan menyertakan header permintaan dalam laporan. Nilai default-nya `false`.
- `includeResponseHeaders` (boolean)— Apakah akan menyertakan header respons dalam laporan tersebut. Nilai default-nya `false`.
- `restrictedHeaders` (array)— Sebuah daf.ar nilai header untuk diabaikan, jika header disertakan. Ini berlaku untuk header permintaan dan respons. Misalnya, Anda dapat menyembunyikan kredensialnya dengan meneruskan includeRequestHeadersas `true` dan RestrictedHeaders as. `['Authorization']`
- `includeRequestBody` (boolean)— Apakah akan menyertakan bodi permintaan dalam laporan. Nilai default-nya `false`.
- `includeResponseBody` (boolean)— Apakah akan menyertakan bodi respons dalam laporan. Standarnya adalah `false`.

setConfig (opsi) mengenai metrik CloudWatch

Untuk canary yang menggunakan `syn-nodejs-puppeteer-3.1` atau yang lebih baru, (options) untuk setConfig dapat mencakup parameter Boolean berikut yang menentukan metrik mana yang diterbitkan oleh canary. Bawaan untuk masing-masing opsi ini adalah `true`. Pilihan yang dimulai dengan `aggregated` menentukan apakah metrik dipancarkan tanpa dimensi `CanaryName`. Anda dapat menggunakan metrik ini untuk melihat hasil gabungan untuk semua canary Anda. Pilihan lain menentukan apakah metrik dipancarkan dengan dimensi `CanaryName`. Anda dapat menggunakan metrik ini untuk melihat hasil untuk setiap canary individu.

Untuk daftar CloudWatch metrik yang dipancarkan oleh burung kenari, lihat. CloudWatch metrik yang diterbitkan oleh kenari

- `failedCanaryMetric` (boolean)— Apakah akan memancarkan metrik `Failed` (dengan dimensi `CanaryName`) untuk canary ini. Nilai default-nya `true`.
- `failedRequestsMetric` (boolean)— Apakah akan memancarkan metrik `Failed requests` (dengan dimensi `CanaryName`) untuk canary ini. Nilai default-nya `true`.
- `_2xxMetric` (boolean)— Apakah akan memancarkan metrik `2xx` (dengan dimensi `CanaryName`) untuk canary ini. Nilai default-nya `true`.
- `_4xxMetric` (boolean)— Apakah akan memancarkan metrik `4xx` (dengan dimensi `CanaryName`) untuk canary ini. Nilai default-nya `true`.
- `_5xxMetric` (boolean)— Apakah akan memancarkan metrik `5xx` (dengan dimensi `CanaryName`) untuk canary ini. Nilai default-nya `true`.
- `stepDurationMetric` (boolean)— Apakah akan memancarkan metrik `Step duration` (dengan dimensi `CanaryName StepName`) untuk canary ini. Bawaannya adalah `true`.
- `stepSuccessMetric` (boolean)— Apakah akan memancarkan metrik `Step success` (dengan dimensi `CanaryName StepName`) untuk canary ini. Bawaannya adalah `true`.
- `aggregatedFailedCanaryMetric` (boolean)— Apakah akan memancarkan metrik `Failed` (tanpa dimensi `CanaryName`) untuk canary ini. Bawaannya adalah `true`.
- `aggregatedFailedRequestsMetric` (boolean)— Apakah akan memancarkan metrik `Failed Requests` (tanpa dimensi `CanaryName`) untuk canary ini. Bawaannya adalah `true`.
- `aggregated2xxMetric` (boolean)— Apakah akan memancarkan metrik `2xx` (tanpa dimensi `CanaryName`) untuk canary ini. Bawaannya adalah `true`.
- `aggregated4xxMetric` (boolean)— Apakah akan memancarkan metrik `4xx` (tanpa dimensi `CanaryName`) untuk canary ini. Bawaannya adalah `true`.
- `aggregated5xxMetric` (boolean)— Apakah akan memancarkan metrik `5xx` (tanpa dimensi `CanaryName`) untuk canary ini. Bawaannya adalah `true`.
- `visualMonitoringSuccessPercentMetric` (boolean)— Apakah akan memancarkan metrik `visualMonitoringSuccessPercent` untuk canary ini. Bawaannya adalah `true`.
- `visualMonitoringTotalComparisonsMetric` (boolean)— Apakah akan memancarkan metrik `visualMonitoringTotalComparisons` untuk canary ini. Bawaannya adalah `false`.
- `stepsReport` (boolean)— Apakah akan melaporkan ringkasan eksekusi langkah. Nilai default-nya `true`.
- `includeUrlPassword` (boolean)— Apakah akan menyertakan kata sandi yang muncul di URL. Secara default, kata sandi yang muncul di URL disunting dari log dan laporan, untuk mencegah pengungkapan data sensitif. Bawaannya adalah `false`.

- `restrictedUrlParameters` (array)— Daftar jalur URL atau parameter kueri untuk disunting. Ini berlaku untuk URL yang muncul di log, laporan, dan kesalahan. Parameter tersebut tidak peka terhadap huruf besar/kecil. Anda dapat meneruskan tanda bintang (*) sebagai nilai untuk menyunting semua jalur URL dan nilai parameter kueri. Default-nya adalah array kosong.
- `logRequest` (boolean)— Apakah akan mencatat setiap permintaan di log canary. Untuk canary UI, ini mencatat setiap permintaan yang dikirim oleh browser. Nilai default-nya `true`.
- `logResponse` (boolean)— Apakah akan mencatat setiap respons di log canary. Untuk canary UI, ini membuat log terhadap setiap respons yang diterima oleh browser. Bawaannya adalah `true`.
- `logRequestBody` (boolean)— Apakah akan mencatat bodi permintaan bersama dengan permintaan di log canary. Konfigurasi ini hanya berlaku jika `logRequest` merupakan `true`. Nilai default-nya `false`.
- `logResponseBody` (boolean)— Apakah akan mencatat bodi respons bersama dengan respons di log canary. Konfigurasi ini hanya berlaku jika `logResponse` merupakan `true`. Nilai default-nya `false`.
- `logRequestHeaders` (boolean)— Apakah akan mencatat header permintaan bersama dengan permintaan di log canary. Konfigurasi ini hanya berlaku jika `logRequest` merupakan `true`. Nilai default-nya `false`.

  Perhatikan bahwa `includeRequestHeaders` memungkinkan header dalam artefak.
- `logResponseHeaders` (boolean)— Apakah akan mencatat header respons bersama dengan respons di log canary. Konfigurasi ini hanya berlaku jika `logResponse` merupakan `true`. Nilai default-nya `false`.

  Perhatikan bahwa `includeResponseHeaders` memungkinkan header dalam artefak.

> **Note**
>
> Metrik `Duration` dan `SuccessPercent` selalu dipancarkan untuk setiap canary, baik dengan dan tanpa metrik `CanaryName`.

Metode untuk mengaktifkan atau menonaktifkan metrik

disableAggregatedRequestMetrik ()

Menonaktifkan canary dari memancarkan semua metrik permintaan yang dipancarkan tanpa dimensi `CanaryName`.

disableRequestMetrics()

Menonaktifkan semua metrik permintaan, termasuk metrik per canary dan metrik yang dikumpulkan di semua canary.

disableStepMetrics()

Menonaktifkan semua metrik langkah, termasuk metrik langkah sukses dan metrik durasi langkah.

enableAggregatedRequestMetrik ()

Mengaktifkan canary untuk memancarkan semua metrik permintaan yang dipancarkan tanpa dimensi `CanaryName`.

enableRequestMetrics()

Mengaktifkan semua metrik permintaan, termasuk metrik per canary dan metrik yang dikumpulkan di semua canary.

enableStepMetrics()

Mengaktifkan semua metrik langkah, termasuk metrik keberhasilan langkah dan metrik durasi langkah.

get2xxMetric()

Mengembalikan apakah canary memancarkan metrik `2xx` dengan dimensi `CanaryName`.

get4xxMetric()

Mengembalikan apakah canary memancarkan metrik `4xx` dengan dimensi `CanaryName`.

get5xxMetric()

Mengembalikan apakah canary memancarkan metrik `5xx` dengan dimensi `CanaryName`.

getAggregated2xxMetric()

Mengembalikan apakah canary memancarkan metrik `2xx` tanpa dimensi.

getAggregated4xxMetric()

Mengembalikan apakah canary memancarkan metrik `4xx` tanpa dimensi.

getAggregatedFailedCanaryMetric()

Mengembalikan apakah canary memancarkan metrik `Failed` tanpa dimensi.

getAggregatedFailedRequestsMetric()

Mengembalikan apakah canary memancarkan metrik `Failed requests` tanpa dimensi.

getAggregated5xxMetric()

Mengembalikan apakah canary memancarkan metrik `5xx` tanpa dimensi.

getFailedCanaryMetrik ()

Mengembalikan apakah canary memancarkan metrik `Failed` dengan dimensi `CanaryName`.

getFailedRequestsMetrik ()

Mengembalikan apakah canary memancarkan metrik `Failed requests` dengan dimensi `CanaryName`.

getStepDurationMetrik ()

Mengembalikan apakah canary memancarkan metrik `Duration` dengan dimensi `CanaryName` untuk canary ini.

getStepSuccessMetrik ()

Mengembalikan apakah canary memancarkan metrik `StepSuccess` dengan dimensi `CanaryName` untuk canary ini.

with2xxMetric(_2xxMetric)

Menerima argumen Boolean, yang menentukan apakah akan memancarkan metrik `2xx` dengan dimensi `CanaryName` untuk canary ini.

with4xxMetric(_4xxMetric)

Menerima argumen Boolean, yang menentukan apakah akan memancarkan metrik `4xx` dengan dimensi `CanaryName` untuk canary ini.

with5xxMetric(_5xxMetric)

Menerima argumen Boolean, yang menentukan apakah akan memancarkan metrik `5xx` dengan dimensi `CanaryName` untuk canary ini.

withAggregated2xxMetric(aggregated2xxMetric)

Menerima argumen Boolean, yang menentukan apakah akan memancarkan metrik `2xx` tanpa dimensi untuk canary ini.

withAggregated4xxMetric(aggregated4xxMetric)

Menerima argumen Boolean, yang menentukan apakah akan memancarkan metrik `4xx` tanpa dimensi untuk canary ini.

withAggregated5xxMetric(aggregated5xxMetric)

Menerima argumen Boolean, yang menentukan apakah akan memancarkan metrik `5xx` tanpa dimensi untuk canary ini.

withAggregatedFailedCanaryMetric(aggregatedFailedCanaryMetrik)

Menerima argumen Boolean, yang menentukan apakah akan memancarkan metrik `Failed` tanpa dimensi untuk canary ini.

withAggregatedFailedRequestsMetric(aggregatedFailedRequestsMetrik)

Menerima argumen Boolean, yang menentukan apakah akan memancarkan metrik `Failed requests` tanpa dimensi untuk canary ini.

withFailedCanaryMetrik (failedCanaryMetric)

Menerima argumen Boolean, yang menentukan apakah akan memancarkan metrik `Failed` dengan dimensi `CanaryName` untuk canary ini.

withFailedRequestsMetrik (failedRequestsMetric)

Menerima argumen Boolean, yang menentukan apakah akan memancarkan metrik `Failed requests` dengan dimensi `CanaryName` untuk canary ini.

withStepDurationMetrik (stepDurationMetric)

Menerima argumen Boolean, yang menentukan apakah akan memancarkan metrik `Duration` dengan dimensi `CanaryName` untuk canary ini.

withStepSuccessMetrik (stepSuccessMetric)

Menerima argumen Boolean, yang menentukan apakah akan memancarkan metrik `StepSuccess` dengan dimensi `CanaryName` untuk canary ini.

Metode untuk mengaktifkan atau menonaktifkan fitur lainnya

withHarFile()

Menerima argumen Boolean, yang menentukan apakah akan membuat file HAR untuk canary ini.

withStepsReport()

Menerima argumen Boolean, yang menentukan apakah akan melaporkan ringkasan eksekusi langkah untuk canary ini.

withIncludeUrlKata Sandi ()

Menerima argumen Boolean, yang menentukan apakah akan menyertakan kata sandi yang muncul di URL di log dan laporan.

withRestrictedUrlParameter ()

Menerima array jalur URL atau parameter kueri untuk disunting. Ini berlaku untuk URL yang muncul di log, laporan, dan kesalahan. Anda dapat meneruskan tanda bintang (*) sebagai nilai untuk menyunting semua jalur URL dan nilai parameter kueri

withLogRequest()

Menerima argumen Boolean, yang menentukan apakah akan membuat log setiap permintaan di log canary.

withLogResponse()

Menerima argumen Boolean, yang menentukan apakah akan membuat log setiap respons di log canary.

withLogRequestTubuh ()

Menerima argumen Boolean, yang menentukan apakah akan membuat log setiap bodi permintaan di log canary.

withLogResponseTubuh ()

Menerima argumen Boolean, yang menentukan apakah akan mencatat setiap bodi respons di log canary.

withLogRequestHeader ()

Menerima argumen Boolean, yang menentukan apakah akan mencatat setiap header permintaan di log canary.

withLogResponseHeader ()

Menerima argumen Boolean, yang menentukan apakah akan mencatat setiap header respons di log canary.

getHarFile()

Mengembalikan apakah canary membuat file HAR.

getStepsReport()

Mengembalikan apakah canary melaporkan ringkasan eksekusi langkah.

getIncludeUrlKata Sandi ()

Mengembalikan apakah canary menyertakan kata sandi yang muncul di URL di log dan laporan.

getRestrictedUrlParameter ()

Mengembalikan apakah canary menyunting jalur URL atau parameter kueri.

getLogRequest()

Mengembalikan apakah canary mencatat setiap permintaan di log canary.

getLogResponse()

Mengembalikan apakah canary mencatat setiap respons di log canary.

getLogRequestTubuh ()

Mengembalikan apakah canary mencatat setiap badan permintaan di log canary.

getLogResponseTubuh ()

Mengembalikan apakah canary mencatat setiap bodi respons di log canary.

getLogRequestHeader ()

Mengembalikan apakah canary mencatat setiap header permintaan di log canary.

getLogResponseHeader ()

Mengembalikan apakah canary mencatat setiap header respons di log canary.

Fungsi untuk semua canary

- `withIncludeRequestHeaders`(includeRequestHeaders)
- `withIncludeResponseHeaders`(includeResponseHeaders)
- `withRestrictedHeaders`(restrictedHeaders)
- `withIncludeRequestBody`(includeRequestBody)
- `withIncludeResponseBody`(includeResponseBody)
- `enableReportingOptions`() - Mengaktifkan semua opsi pelaporan-- includeRequestHeadersincludeResponseHeaders,, includeRequestBody, dan includeResponseBody,.
- `disableReportingOptions`() — Menonaktifkan semua opsi pelaporan-- includeRequestHeaders, includeResponseHeaders, includeRequestBody, dan includeResponseBody,.

setConfig(options) untuk canary UI

Untuk canary UI, setConfig dapat mencakup parameter Boolean berikut.

- `continueOnStepFailure` (boolean)— Apakah akan melanjutkan menjalankan skrip canary setelah langkah gagal (ini mengacu pada fungsi executeStep). Jika ada langkah yang gagal, proses canary akan tetap ditandai sebagai gagal. Bawaannya adalah `false`.
- `harFile` (boolean)— Apakah akan membuat file HAR. Bawaannya adalah `True`.
- `screenshotOnStepStart` (boolean)— Apakah mengambil tangkapan layar sebelum memulai langkah.
- `screenshotOnStepSuccess` (boolean)— Apakah mengambil tangkapan layar setelah menyelesaikan langkah yang berhasil.

- `screenshotOnStepFailure` (boolean)— Apakah mengambil tangkapan layar setelah langkah gagal.

Metode untuk mengaktifkan atau menonaktifkan tangkapan layar

disableStepScreenshots()

Menonaktifkan semua opsi tangkapan layar (screenshotOnStepMulai, screenshotOnStep Sukses, dan screenshotOnStep Kegagalan).

enableStepScreenshots()

Mengaktifkan semua opsi tangkapan layar (screenshotOnStepMulai, screenshotOnStep Sukses, dan screenshotOnStep Kegagalan). Secara bawaan, semua metode ini diaktifkan.

getScreenshotOnStepFailure()

Mengembalikan apakah canary mengambil tangkapan layar setelah langkah gagal.

getScreenshotOnStepStart()

Mengembalikan apakah canary mengambil tangkapan layar sebelum memulai langkah.

getScreenshotOnStepSuccess()

Mengembalikan apakah canary mengambil tangkapan layar setelah menyelesaikan satu langkah dengan sukses.

withScreenshotOnStepStart(screenshotOnStepMulai)

Menerima argumen Boolean, yang menunjukkan apakah akan mengambil tangkapan layar sebelum memulai sebuah langkah.

withScreenshotOnStepSuccess(screenshotOnStepSukses)

Menerima argumen Boolean, yang menunjukkan apakah akan mengambil tangkapan layar setelah menyelesaikan langkah dengan sukses.

withScreenshotOnStepFailure(screenshotOnStepKegagalan)

Menerima argumen Boolean, yang menunjukkan apakah akan mengambil tangkapan layar setelah langkah gagal.

Penggunaan di canary UI

Pertama, impor dependensi synthetics dan ambil konfigurasinya.

```
// Import Synthetics dependency
const synthetics = require('Synthetics');

// Get Synthetics configuration
const synConfig = synthetics.getConfiguration();
```

Lalu, atur konfigurasi untuk setiap pilihan dengan memanggil metode setConfig menggunakan salah satu pilihan berikut.

```
// Set configuration values
    synConfig.setConfig({
        screenshotOnStepStart: true,
        screenshotOnStepSuccess: false,
        screenshotOnStepFailure: false
    });
```

Atau

```
synConfig.withScreenshotOnStepStart(false).withScreenshotOnStepSuccess(true).withScreenshotOnSt
```

Untuk menonaktifkan semua tangkapan layar, gunakan fungsi disableStepScreenshots () seperti pada contoh ini.

```
synConfig.disableStepScreenshots();
```

Anda dapat mengaktifkan dan menonaktifkan tangkapan layar kapan saja di kode tersebut. Misalnya, untuk menonaktifkan tangkapan layar hanya untuk satu langkah, nonaktifkan sebelum menjalankan langkah tersebut dan kemudian aktifkan setelah langkah.

setConfig(options) untuk canary API

Untuk canary API, setConfig dapat mencakup parameter Boolean berikut:

- `continueOnHttpStepFailure`(boolean) - Apakah akan melanjutkan menjalankan skrip kenari setelah langkah HTTP gagal (ini mengacu pada executeHttpStepfungsi). Jika ada langkah yang gagal, proses canary akan tetap ditandai sebagai gagal. Bawaannya adalah `true`.

Pemantauan visual

Pemantauan visual membandingkan tangkapan layar yang diambil selama berjalannya canary dengan tangkapan layar yang diambil selama berjalannya canary dasar. Jika perbedaan antara kedua tangkapan layar berada di luar persentase ambang batas, canary gagal dan Anda dapat melihat area dengan perbedaan warna yang disorot dalam laporan lari canary. Pemantauan visual didukung di kenari yang menjalankan syn-puppeteer-node-3.2 dan yang lebih baru. Saat ini tidak didukung di canary yang menjalankan Python dan Selenium.

Untuk mengaktifkan pemantauan visual, tambahkan baris kode berikut ke skrip canary. Untuk detail selengkapnya, silakan lihat SyntheticsConfiguration kelas.

```
syntheticsConfiguration.withVisualCompareWithBaseRun(true);
```

Pertama kali canary berjalan dengan sukses setelah baris ini ditambahkan ke skrip, ia menggunakan tangkapan layar yang diambil selama proses itu sebagai dasar untuk perbandingan. Setelah kenari pertama dijalankan, Anda dapat menggunakan CloudWatch konsol untuk mengedit kenari untuk melakukan salah satu hal berikut:

- Tetapkan putaran canary berikutnya sebagai dasar baru.
- Gambar batas pada tangkapan layar dasar saat ini untuk menunjuk area tangkapan layar untuk diabaikan selama perbandingan visual.
- Hapus tangkapan layar agar tidak digunakan untuk pemantauan visual.

Untuk informasi selengkapnya tentang menggunakan CloudWatch konsol untuk mengedit kenari, lihatMengedit atau menghapus canary.

Opsi lain untuk pemantauan visual

SyntheticsConfiguration. withVisualVarianceThresholdPercentage(DiinginkanPersentase)

Tetapkan persentase yang dapat diterima untuk varians tangkapan layar dalam perbandingan visual.

SyntheticsConfiguration. withVisualVarianceHighlightHexColor(” #fafa00 “)

Atur warna sorotan yang menunjuk area varians saat Anda melihat laporan canary run yang menggunakan pemantauan visual.

SyntheticsConfiguration. withFailCanaryRunOnVisualVariance(FailCanary)

Atur apakah canary gagal atau tidak ketika ada perbedaan visual yang lebih dari ambang batas. Default-nya adalah menggagalkan canary.

Logger Synthetic

SyntheticsLogger menulis log keluar ke konsol dan ke file log lokal pada tingkat log yang sama. File log ini ditulis ke kedua lokasi hanya jika tingkat log berada pada atau di bawah tingkat log yang diinginkan dari fungsi log yang dipanggil tersebut.

Pernyataan pembuatan log di file log lokal diawali dengan "DEBUG: ", "INFO: ", dan seterusnya untuk mencocokkan tingkat log dari fungsi yang dipanggil tersebut.

Anda dapat menggunakan SyntheticsLogger, dengan asumsi Anda ingin menjalankan Synthetics Library pada tingkat log yang sama dengan logging canary Synthetics Anda.

Menggunakan tidak SyntheticsLogger diperlukan untuk membuat file log yang diunggah ke lokasi hasil S3 Anda. Sebagai penggantinya, Anda dapat membuat file log lain di `/tmp` folder. File apa pun yang dibuat di bawah `/tmp` folder diunggah ke lokasi hasil dalam S3 sebagai artefak.

Untuk menggunakan pencatat Pustaka Synthetics:

```
const log = require('SyntheticsLogger');
```

Definisi fungsi yang berguna:

log.debug(***message***, ***ex***);

Parameter: *message* adalah pesan untuk log. *ex* adalah pengecualian, jika ada, untuk log.

Contoh:

```
log.debug("Starting step - login.");
```

log.error(***message***, ***ex***);

Parameter: *message* adalah pesan untuk log. *ex* adalah pengecualian, jika ada, untuk log.

Contoh:

```
try {
  await login();
catch (ex) {
```

```
  log.error("Error encountered in step - login.", ex);
}
```

log.info(***message***, ***ex***);

Parameter: *message* adalah pesan untuk log. *ex* adalah pengecualian, jika ada, untuk log.

Contoh:

```
log.info("Successfully completed step - login.");
```

log.log(***message***, ***ex***);

Ini adalah nama lain untuk `log.info`.

Parameter: *message* adalah pesan untuk log. *ex* adalah pengecualian, jika ada, untuk log.

Contoh:

```
 log.log("Successfully completed step - login.");
```

log.warn(***message***, ***ex***);

Parameter: *message* adalah pesan untuk log. *ex* adalah pengecualian, jika ada, untuk log.

Contoh:

```
log.warn("Exception encountered trying to publish CloudWatch Metric.", ex);
```

SyntheticsLogHelper kelas

Kelas `SyntheticsLogHelper` tersedia di runtime `syn-nodejs-puppeteer-3.2` dan runtime yang lebih baru. Ini sudah diinisialisasi di perpustakaan CloudWatch Synthetics dan dikonfigurasi dengan konfigurasi Synthetics. Anda dapat menambahkan ini sebagai Dependensi dalam skrip Anda. Kelas ini memungkinkan Anda untuk membersihkan URL, header, dan pesan kesalahan untuk menyunting informasi sensitif.

> **Note**
>
> Synthetics membersihkan semua URL dan pesan kesalahan yang dicatat sebelum memasukkannya ke dalam log, laporan, file HAR, dan kesalahan canary run berdasarkan

pengaturan konfigurasi Synthetics `restrictedUrlParameters`. Anda harus menggunakan `getSanitizedUrl` atau `getSanitizedErrorMessage` hanya jika Anda membuat log URL atau kesalahan dalam skrip Anda. Synthetics tidak menyimpan artefak canary kecuali kesalahan canary yang dilemparkan oleh skrip tersebut. Artefak Canary run disimpan di akun pelanggan Anda. Untuk informasi selengkapnya, lihat Pertimbangan keamanan untuk canary Synthetics.

getSanitizedUrl(url, StepConfig = nol)

Fungsi ini tersedia di `syn-nodejs-puppeteer-3.2` dan nanti. Ia mengembalikan string url sanitasi berdasarkan konfigurasi. Anda dapat memilih untuk menyunting parameter URL sensitif seperti kata sandi dan access_token dengan menyetel properti `restrictedUrlParameters`. Secara default, kata sandi di URL disunting. Anda dapat mengaktifkan kata sandi URL jika diperlukan dengan menyetel `includeUrlPassword` ke true.

Fungsi ini memunculkan kesalahan jika URL yang dilewatkan bukan sebuah URL yang valid.

Parameter

- *`url`* adalah string dan merupakan URL untuk membersihkan.
- *`StepConfig`* (Opsional) mengganti konfigurasi Synthetics global untuk fungsi ini. Jika `stepConfig` tidak diteruskan, konfigurasi global digunakan untuk membersihkan URL.

Contoh

Contoh ini menggunakan URL sampel berikut: `https://example.com/learn/home?access_token=12345&token_type=Bearer&expires_in=1200`. Dalam contoh ini, `access_token` berisi informasi sensitif Anda yang tidak boleh dibuat log. Perhatikan bahwa layanan Synthetics tidak menyimpan artefak canary run. Artefak seperti log, tangkapan layar, dan laporan semuanya disimpan dalam bucket Amazon S3 di akun pelanggan Anda.

Langkah pertama adalah mengatur konfigurasi Synthetics.

```
// Import Synthetics dependency
const synthetics = require('Synthetics');

// Import Synthetics logger for logging url
const log = require('SyntheticsLogger');
```

```
// Get Synthetics configuration
const synConfig = synthetics.getConfiguration();

// Set restricted parameters
synConfig.setConfig({
   restrictedUrlParameters: ['access_token'];
});
```

Berikutnya, bersihkan dan log URL

```
// Import SyntheticsLogHelper dependency
const syntheticsLogHelper = require('SyntheticsLogHelper');

const sanitizedUrl = synthetics.getSanitizedUrl('https://example.com/learn/home?
access_token=12345&token_type=Bearer&expires_in=1200');
```

Ini mencatat yang berikut ini di log canary Anda.

```
My example url is: https://example.com/learn/home?
access_token=REDACTED&token_type=Bearer&expires_in=1200
```

Anda dapat mengganti konfigurasi Synthetics untuk URL dengan meneruskan parameter opsional yang berisi opsi konfigurasi Synthetics, seperti pada contoh berikut.

```
const urlConfig = {
   restrictedUrlParameters = ['*']
};
const sanitizedUrl = synthetics.getSanitizedUrl('https://example.com/learn/home?
access_token=12345&token_type=Bearer&expires_in=1200', urlConfig);
logger.info('My example url is: ' + sanitizedUrl);
```

Contoh sebelumnya menyunting semua parameter kueri, dan dibuat log sebagai berikut:

```
My example url is: https://example.com/learn/home?
access_token=REDACTED&token_type=REDACTED&expires_in=REDACTED
```

getSanitizedErrorPesan

Fungsi ini tersedia di `syn-nodejs-puppeteer-3.2` dan nanti. Ini mengembalikan string kesalahan yang dibersihkan dengan membersihkan URL apa pun yang ada berdasarkan konfigurasi Synthetics.

Anda dapat memilih untuk mengganti konfigurasi Synthetics global ketika Anda memanggil fungsi ini dengan meneruskan parameter `stepConfig` opsional.

Parameter

- *`kesalahan`* adalah kesalahan untuk membersihkan. Ini bisa berupa objek Kesalahan atau string.
- *`StepConfig`* (Opsional) mengganti konfigurasi Synthetics global untuk fungsi ini. Jika `stepConfig` tidak diteruskan, konfigurasi global digunakan untuk membersihkan URL.

Contoh

Contoh ini menggunakan kesalahan berikut: `Failed to load url: https://example.com/learn/home?access_token=12345&token_type=Bearer&expires_in=1200`

Langkah pertama adalah mengatur konfigurasi Synthetics.

```
// Import Synthetics dependency
const synthetics = require('Synthetics');

// Import Synthetics logger for logging url
const log = require('SyntheticsLogger');

// Get Synthetics configuration
const synConfig = synthetics.getConfiguration();

// Set restricted parameters
synConfig.setConfig({
   restrictedUrlParameters: ['access_token'];
});
```

Berikutnya, bersihkan dan buat log pesan kesalahan

```
// Import SyntheticsLogHelper dependency
const syntheticsLogHelper = require('SyntheticsLogHelper');

try {
   // Your code which can throw an error containing url which your script logs
} catch (error) {
    const sanitizedErrorMessage = synthetics.getSanitizedErrorMessage(errorMessage);
    logger.info(sanitizedErrorMessage);
}
```

Ini mencatat yang berikut ini di log canary Anda.

```
Failed to load url: https://example.com/learn/home?
access_token=REDACTED&token_type=Bearer&expires_in=1200
```

getSanitizedHeaders(header, StepConfig = null)

Fungsi ini tersedia di `syn-nodejs-puppeteer-3.2` dan nanti. Ini mengembalikan header yang dibersihkan berdasarkan properti `restrictedHeaders` dari `syntheticsConfiguration`. Header yang ditentukan dalam `restrictedHeaders` properti disunting dari log, file HAR, dan laporan.

Parameter

- *`header`* adalah objek yang berisi header untuk membersihkan.
- *`StepConfig`* (Opsional) mengganti konfigurasi Synthetics global untuk fungsi ini. Jika `stepConfig` tidak diteruskan, konfigurasi global digunakan untuk membersihkan header.

Kelas dan fungsi pustaka Node.js yang hanya berlaku untuk canary UI

Fungsi pustaka CloudWatch Synthetics berikut untuk Node.js hanya berguna untuk kenari UI.

Topik

- Kelas Synthetics
- BrokenLinkCheckerReport kelas
- SyntheticsLink kelas

Kelas Synthetics

Fungsi-fungsi berikut berada dalam kelas Synthetics.

async addUserAgent (halaman, userAgentString);

Fungsi ini ditambahkan *`userAgentString`*ke header user-agent halaman yang ditentukan.

Contoh:

```
await synthetics.addUserAgent(page, "MyApp-1.0");
```

Hasil di header pengguna-agen halaman yang diatur untuk *browsers-user-agent-header-value*`MyApp-1.0`

async executeStep (StepName,, [StepConfig]); functionToExecute

Mengeksekusi langkah yang disediakan, membungkusnya dengan pembuatan log mulai/lulus/gagal, tangkapan layar mulai/lulus/gagal, dan metrik lulus/gagal dan durasi.

> **Note**
>
> Jika Anda menggunakan runtime `syn-nodejs-2.1` atau runtime yang lebih baru, Anda dapat mengonfigurasi apakah dan kapan tangkapan layar diambil. Untuk informasi selengkapnya, lihat SyntheticsConfiguration kelas.

Fungsi `executeStep` juga melakukan hal berikut:

- Membuat log bahwa langkah tersebut dimulai.
- Mengambil tangkapan layar yang bernama `<stepName>-starting`.
- Memulai pengatur waktu.
- Menjalankan fungsi yang disediakan.
- Jika fungsi kembali secara normal, ini dihitung sebagai lulus. Jika fungsi tersebut throw, maka ini dihitung sebagai gagal.
- Mengakhiri pengatur waktu.
- Mencatat log jika langkah berhasil atau gagal
- Mengambil tangkapan layar yang bernama `<stepName>-succeeded` atau `<stepName>-failed`.
- Memancarkan metrik `stepName SuccessPercent`, 100 untuk lulus atau 0 untuk gagal.
- Memancarkan metrik `stepName Duration`, dengan nilai didasarkan pada mulai melangkah dan waktu selesai.
- Akhirnya, mengembalikan apa yang dikembalikan oleh `functionToExecute` atau melempar kembali apa yang dilempar oleh `functionToExecute`.

Jika canary menggunakan runtime `syn-nodejs-2.0` atau lebih baru, fungsi ini juga menambahkan ringkasan pelaksanaan langkah ke laporan canary. Ringkasan mencakup detail tentang setiap

langkah, seperti waktu mulai, waktu akhir, status (LULUS/GAGAL), alasan kegagalan (jika gagal), dan tangkapan layar yang diambil selama pelaksanaan setiap langkah.

Contoh:

```
await synthetics.executeStep('navigateToUrl', async function (timeoutInMillis = 30000)
 {
           await page.goto(url, {waitUntil: ['load', 'networkidle0'], timeout:
 timeoutInMillis});});
```

Respons:

Mengembalikan apa yang `functionToExecute` kembalikan.

Pembaruan dengan syn-nodejs-2.2

Dimulai dengan`syn-nodejs-2.2`, Anda dapat secara opsional meneruskan konfigurasi langkah untuk mengganti konfigurasi CloudWatch Synthetics pada level langkah. Untuk daftar opsi yang dapat Anda teruskan ke `executeStep`, silakan lihat SyntheticsConfiguration kelas.

Contoh berikut menimpa konfigurasi `false` default untuk `continueOnStepFailure` hingga `true` dan menentukan kapan harus mengambil tangkapan layar.

```
var stepConfig = {
    'continueOnStepFailure': true,
    'screenshotOnStepStart': false,
    'screenshotOnStepSuccess': true,
    'screenshotOnStepFailure': false
}

await executeStep('Navigate to amazon', async function (timeoutInMillis = 30000) {
      await page.goto(url, {waitUntil: ['load', 'networkidle0'], timeout:
 timeoutInMillis});
 }, stepConfig);
```

getDefaultLaunchPilihan ();

`getDefaultLaunchOptions()`Fungsi mengembalikan opsi peluncuran browser yang digunakan oleh CloudWatch Synthetics. Untuk informasi selengkapnya, silakan lihat Jenis opsi peluncuran

```
// This function returns default launch options used by Synthetics.
```

```
const defaultOptions = await synthetics.getDefaultLaunchOptions();
```

getPage();

Mengembalikan halaman terbuka saat ini sebagai objek Puppeteer. Untuk informasi selengkapnya, silakan lihat API Puppeteer v1.14.0.

Contoh:

```
let page = synthetics.getPage();
```

Respons:

Halaman (objek Puppeteer) yang saat ini terbuka di sesi browser saat ini.

getRequestResponseLogHelper();

> **Important**
>
> Pada canary yang menggunakan `syn-nodejs-puppeteer-3.2` runtime atau yang lebih baru, fungsi ini tidak digunakan lagi bersama dengan kelas `RequestResponseLogHelper`. Setiap penggunaan fungsi ini menyebabkan peringatan muncul di log canary Anda. Fungsi ini akan dihapus di versi runtime yang akan datang. Jika Anda menggunakan fungsi ini, gunakan RequestResponseLogHelper kelas sebagai gantinya.

Gunakan fungsi ini sebagai pola pembangun untuk mengubah bendera pembuatan log permintaan dan respons.

Contoh:

```
synthetics.setRequestResponseLogHelper(getRequestResponseLogHelper().withLogRequestHeaders(fals
```

Respons:

```
{RequestResponseLogHelper}
```

launch(options)

Opsi untuk fungsi ini hanya tersedia di versi runtime `syn-nodejs-2.1` atau lebih baru.

Fungsi ini hanya digunakan untuk canary UI. Hal ini menutup browser yang sudah ada dan meluncurkan browser baru.

> **Note**
>
> CloudWatch Synthetics selalu meluncurkan browser sebelum mulai menjalankan skrip Anda. Anda tidak perlu memanggil launch() kecuali ingin meluncurkan browser baru dengan pilihan-pilihan kustom.

(options) adalah serangkaian pilihan yang dapat dikonfigurasi untuk diatur di browser. Untuk informasi selengkapnya, silakan lihat Jenis opsi peluncuran.

Jika Anda memanggil fungsi ini tanpa opsi, Synthetics meluncurkan browser dengan argumen default, `executablePath`, dan `defaultViewport`. Tampilan default di CloudWatch Synthetics adalah 1920 x 1080.

Anda dapat mengganti parameter peluncuran yang digunakan oleh CloudWatch Synthetics dan meneruskan parameter tambahan saat meluncurkan browser. Misalnya, snippet kode berikut meluncurkan browser dengan argumen default dan jalur yang dapat dieksekusi secara default, tetapi dengan viewport berukuran 800 x 600.

```
await synthetics.launch({
        defaultViewport: {
            "deviceScaleFactor": 1,
            "width": 800,
            "height": 600
    }});
```

Kode contoh berikut menambahkan `ignoreHTTPSErrors` parameter baru ke parameter peluncuran CloudWatch Synthetics:

```
await synthetics.launch({
        ignoreHTTPSErrors: true
 });
```

Anda dapat menonaktifkan keamanan web dengan menambahkan `--disable-web-security` flag ke args dalam parameter peluncuran CloudWatch Synthetics:

```
// This function adds the --disable-web-security flag to the launch parameters
```

```
const defaultOptions = await synthetics.getDefaultLaunchOptions();
const launchArgs = [...defaultOptions.args, '--disable-web-security'];
await synthetics.launch({
     args: launchArgs
  });
```

RequestResponseLogHelper kelas

> **Important**
>
> Pada canary yang menggunakan runtime `syn-nodejs-puppeteer-3.2` atau yang lebih baru, kelas ini tidak digunakan lagi. Setiap penggunaan kelas ini menyebabkan peringatan muncul di log canary Anda. Fungsi ini akan dihapus di versi runtime yang akan datang. Jika Anda menggunakan fungsi ini, gunakan RequestResponseLogHelper kelas sebagai gantinya.

Menangani konfigurasi halus dan pembuatan representasi string atas muatan permintaan dan respons.

```
class RequestResponseLogHelper {

    constructor () {
        this.request = {url: true, resourceType: false, method: false, headers: false,
 postData: false};
        this.response = {status: true, statusText: true, url: true, remoteAddress:
 false, headers: false};
    }

    withLogRequestUrl(logRequestUrl);

    withLogRequestResourceType(logRequestResourceType);

    withLogRequestMethod(logRequestMethod);

    withLogRequestHeaders(logRequestHeaders);

    withLogRequestPostData(logRequestPostData);


    withLogResponseStatus(logResponseStatus);

    withLogResponseStatusText(logResponseStatusText);
```

```
    withLogResponseUrl(logResponseUrl);

    withLogResponseRemoteAddress(logResponseRemoteAddress);

    withLogResponseHeaders(logResponseHeaders);
```

Contoh:

```
synthetics.setRequestResponseLogHelper(getRequestResponseLogHelper()
.withLogRequestPostData(true)
.withLogRequestHeaders(true)
.withLogResponseHeaders(true));
```

Jawaban:

```
{RequestResponseLogHelper}
```

setRequestResponseLogHelper();

> **Important**
>
> Pada canary yang menggunakan `syn-nodejs-puppeteer-3.2` runtime atau yang lebih baru, fungsi ini tidak digunakan lagi bersama dengan kelas `RequestResponseLogHelper`. Setiap penggunaan fungsi ini menyebabkan peringatan muncul di log canary Anda. Fungsi ini akan dihapus di versi runtime yang akan datang. Jika Anda menggunakan fungsi ini, gunakan RequestResponseLogHelper kelas sebagai gantinya.

Gunakan fungsi ini sebagai pola pembangun untuk mengatur bendera pembuatan log permintaan dan respons.

Contoh:

```
synthetics.setRequestResponseLogHelper().withLogRequestHeaders(true).withLogResponseHeaders(tru
```

Respons:

```
{RequestResponseLogHelper}
```

async takeScreenshot(name, suffix);

Mengambil tangkapan layar (.PNG) halaman saat ini dengan nama dan akhiran (opsional).

Contoh:

```
await synthetics.takeScreenshot("navigateToUrl", "loaded")
```

Contoh ini menangkap dan mengunggah tangkapan layar yang dinamai `01-navigateToUrl-loaded.png` ke bucket S3 canary.

Anda dapat mengambil tangkapan layar untuk langkah canary tertentu dengan meneruskan `stepName` sebagai parameter pertama. Tangkapan layar ditautkan ke langkah canary dalam laporan Anda, untuk membantu Anda melacak setiap langkah saat melakukan debug.

CloudWatch Synthetics canaries secara otomatis mengambil tangkapan layar sebelum memulai langkah (`executeStep`fungsi) dan setelah langkah selesai (kecuali jika Anda mengonfigurasi kenari untuk menonaktifkan tangkapan layar). Anda dapat mengambil lebih banyak tangkapan layar dengan meneruskan nama langkah di fungsi `takeScreenshot`.

Contoh berikut mengambil tangkapan layar dengan `signupForm` sebagai nilai dari `stepName`. Tangkapan layar akan dinamai `02-signupForm-address` dan akan ditautkan dengan langkah yang dinamai `signupForm` dalam laporan canary.

```
await synthetics.takeScreenshot('signupForm', 'address')
```

BrokenLinkCheckerReport kelas

Kelas ini menyediakan metode untuk menambahkan tautan synthetics. Dukungannya hanya pada canary yang menggunakan versi `syn-nodejs-2.0-beta` runtime atau lebih baru.

Untuk menggunakan `BrokenLinkCheckerReport`, sertakan baris berikut dalam skrip:

```
const BrokenLinkCheckerReport = require('BrokenLinkCheckerReport');

const brokenLinkCheckerReport = new BrokenLinkCheckerReport();
```

Definisi fungsi yang berguna:

addLink(***syntheticsLink***, isBroken)

`syntheticsLink` adalah objek `SyntheticsLink` yang mewakili suatu tautan. Fungsi ini menambahkan tautan sesuai dengan kode status. Secara bawaan, ini menganggap tautan rusak jika kode status tidak tersedia atau kode status 400 atau lebih tinggi. Anda dapat mengganti perilaku bawaan ini dengan memasukkan parameter `isBrokenLink` opsional dengan nilai `true` atau `false`.

Fungsi ini tidak memiliki nilai balik.

getLinks()

Fungsi ini mengembalikan susunan objek `SyntheticsLink` yang disertakan dalam laporan pemeriksa tautan yang rusak.

getTotalBrokenTautan ()

Fungsi ini mengembalikan angka yang mewakili total tautan yang rusak.

getTotalLinksDiperiksa ()

Fungsi ini mengembalikan angka yang mewakili total tautan yang disertakan dalam laporan.

Cara menggunakan BrokenLinkCheckerReport

Snippet kode skrip canary berikut menunjukkan contoh navigasi ke tautan dan menambahkannya ke laporan pemeriksa tautan yang rusak.

1. Impor `SyntheticsLink`, `BrokenLinkCheckerReport`, dan `Synthetics`.

```
const BrokenLinkCheckerReport = require('BrokenLinkCheckerReport');
const SyntheticsLink = require('SyntheticsLink');

// Synthetics dependency
const synthetics = require('Synthetics');
```

2. Untuk menambahkan tautan ke laporan, buat instans `BrokenLinkCheckerReport`.

```
let brokenLinkCheckerReport = new BrokenLinkCheckerReport();
```

3. Navigasi ke URL dan tambahkan ke laporan pemeriksa tautan yang rusak.

```
let url = "https://amazon.com";
```

```
let syntheticsLink = new SyntheticsLink(url);

// Navigate to the url.
let page = await synthetics.getPage();

// Create a new instance of Synthetics Link
let link = new SyntheticsLink(url)

try {
    const response = await page.goto(url, {waitUntil: 'domcontentloaded', timeout:
 30000});
} catch (ex) {
    // Add failure reason if navigation fails.
    link.withFailureReason(ex);
}

if (response) {
    // Capture screenshot of destination page
    let screenshotResult = await synthetics.takeScreenshot('amazon-home', 'loaded');

    // Add screenshot result to synthetics link
    link.addScreenshotResult(screenshotResult);

    // Add status code and status description to the link
    link.withStatusCode(response.status()).withStatusText(response.statusText())
}

// Add link to broken link checker report.
brokenLinkCheckerReport.addLink(link);
```

4. Tambahkan laporan ke Synthetics. Hal ini membuat file JSON yang diberi nama `BrokenLinkCheckerReport.json` dalam bucket S3 Anda untuk setiap operasi canary. Anda dapat melihat laporan tautan di konsol untuk setiap operasi canary berikut tangkapan layar, log, dan file HAR.

```
await synthetics.addReport(brokenLinkCheckerReport);
```

SyntheticsLink kelas

Kelas ini menyediakan metode untuk merangkum informasi. Dukungannya hanya pada canary yang menggunakan versi `syn-nodejs-2.0-beta` runtime atau lebih baru.

Untuk menggunakan `SyntheticsLink`, sertakan baris berikut dalam skrip:

```
const SyntheticsLink = require('SyntheticsLink');

const syntheticsLink = new SyntheticsLink("https://www.amazon.com");
```

Fungsi ini mengembalikan `syntheticsLink`*`Object`*

Definisi fungsi yang berguna:

withUrl(***`url`***)

`url` adalah sebuah string URL. Fungsi ini mengembalikan `syntheticsLink`*`Object`*

withText(***`text`***)

`text` adalah sebuah string yang mewakili teks jangkar. Fungsi ini mengembalikan `syntheticsLink`*`Object`*. Ini menambahkan teks jangkar yang sesuai dengan tautan.

withParentUrl(***`ParenTurl`***)

`parentUrl` adalah sebuah string yang mewakili URL induk (halaman sumber). Fungsi ini mengembalikan `syntheticsLink`*`Object`*

withStatusCode(***`StatusCode`***)

`statusCode` adalah sebuah string yang mewakili kode status. Fungsi ini mengembalikan `syntheticsLink`*`Object`*

withFailureReason(***`FailureReason`***)

`failureReason` adalah sebuah string yang mewakili alasan kegagalan. Fungsi ini mengembalikan `syntheticsLink`*`Object`*

addScreenshotResult(***`ScreenshotResult`***)

`screenshotResult` adalah sebuah objek. Ini adalah instans `ScreenshotResult` yang dikembalikan oleh fungsi Synthetics `takeScreenshot`. Objek tersebut meliputi hal berikut:

- `fileName`— Sebuah string yang mewakili `screenshotFileName`
- `pageUrl` (opsional)

- `error` (opsional)

Kelas dan fungsi pustaka Node.js yang hanya berlaku untuk canary API

Fungsi pustaka CloudWatch Synthetics berikut untuk Node.js hanya berguna untuk kenari API.

Topik

executeHttpStep(StepName, requestOptions, [callback], [stepConfig])

Mengeksekusi permintaan HTTP yang disediakan sebagai langkah, dan menerbitkan `SuccessPercent` (lulus/gagal) dan metrik `Duration`.

executeHttpStepmenggunakan fungsi asli HTTP atau HTTPS di bawah tenda, tergantung pada protokol yang ditentukan dalam permintaan.

Fungsi ini juga menambahkan ringkasan pelaksanaan langkah ke laporan canary. Ringkasan mencakup detail tentang setiap permintaan HTTP, seperti berikut ini:

- Waktu mulai
- Waktu akhir
- Status (LULUS/GAGAL)
- Alasan kegagalan, jika gagal
- Detail panggilan HTTP seperti header permintaan/respons, bodi, kode status, pesan status, dan pengaturan waktu performa.

Parameter

stepName(***`String`***)

Menentukan nama dari langkah tersebut. Nama ini juga digunakan untuk menerbitkan CloudWatch metrik untuk langkah ini.

requestOptions(***`Objek or String`***)

Nilai parameter ini dapat berupa URL, string URL, atau sebuah objek. Jika itu adalah objek, maka itu harus berupa serangkaian opsi yang dapat dikonfigurasi untuk membuat permintaan HTTP. Ini mendukung semua opsi di http.request(options[, callback]) dalam dokumentasi Node.js.

Selain opsi Node.js ini, requestOptions mendukung parameter tambahan body. Anda dapat menggunakan parameter body untuk meneruskan data sebagai bodi permintaan.

callback(***respons***)

(Opsional) Ini adalah fungsi pengguna yang diinvokasi dengan respons HTTP. Responsnya adalah dari tipe Class: http. IncomingMessage.

stepConfig(***objek***)

(Opsional) Gunakan parameter ini untuk mengganti konfigurasi synthetics global dengan konfigurasi yang berbeda untuk langkah ini.

Contoh penggunaan executeHttpStep

Rangkaian contoh berikut membangun satu sama lain untuk menggambarkan berbagai penggunaan opsi ini.

Contoh pertama ini mengonfigurasi parameter permintaan. Anda dapat meneruskan URL sebagai requestOptions:

```
let requestOptions = 'https://www.amazon.com';
```

Atau Anda dapat meneruskan satu set opsi:

```
let requestOptions = {
        'hostname': 'myproductsEndpoint.com',
        'method': 'GET',
        'path': '/test/product/validProductName',
        'port': 443,
        'protocol': 'https:'
    };
```

Contoh berikutnya menciptakan fungsi callback yang menerima respons. Secara default, jika Anda tidak menentukan callback, CloudWatch Synthetics memvalidasi bahwa statusnya antara 200 dan 299 inklusif.

```
// Handle validation for positive scenario
    const callback = async function(res) {
        return new Promise((resolve, reject) => {
            if (res.statusCode < 200 || res.statusCode > 299) {
```

```
                throw res.statusCode + ' ' + res.statusMessage;
            }

            let responseBody = '';
            res.on('data', (d) => {
                responseBody += d;
            });

            res.on('end', () => {
                // Add validation on 'responseBody' here if required. For ex, your
 status code is 200 but data might be empty
                resolve();
            });
        });
    };
```

Contoh berikutnya membuat konfigurasi untuk langkah ini yang mengesampingkan konfigurasi Synthetics global CloudWatch . Konfigurasi langkah dalam contoh ini memungkinkan header permintaan, header respons, bodi permintaan (data posting), dan bodi respons dalam laporan Anda dan membatasi nilai header 'X-Amz-Security-Token' dan 'Authorization'. Secara bawaan, nilai-nilai ini tidak termasuk dalam laporan untuk alasan keamanan. Jika Anda memilih untuk memasukkannya, data hanya disimpan dalam bucket S3 Anda.

```
// By default headers, post data, and response body are not included in the report for
 security reasons.
// Change the configuration at global level or add as step configuration for individual
 steps
let stepConfig = {
    includeRequestHeaders: true,
    includeResponseHeaders: true,
    restrictedHeaders: ['X-Amz-Security-Token', 'Authorization'], // Restricted header
 values do not appear in report generated.
    includeRequestBody: true,
    includeResponseBody: true
};
```

Contoh terakhir ini meneruskan permintaan Anda ke executeHttpStepdan memberi nama langkahnya.

```
await synthetics.executeHttpStep('Verify GET products API', requestOptions, callback,
 stepConfig);
```

Dengan kumpulan contoh ini, CloudWatch Synthetics menambahkan detail dari setiap langkah dalam laporan Anda dan menghasilkan metrik untuk setiap langkah menggunakan StepName.

Anda akan melihat metrik `successPercent` dan `duration` untuk langkah `Verify GET products API`. Anda dapat memantau performa API Anda dengan memantau metrik untuk langkah panggilan API Anda.

Untuk sampel skrip lengkap yang menggunakan fungsi-fungsi ini, silakan lihat Canary API multi-langkah.

Fungsi pustaka tersedia untuk skrip canary Python yang menggunakan Selenium

Bagian ini mencantumkan fungsi pustaka Selenium yang tersedia untuk skrip canary Python.

Topik

- Kelas dan fungsi pustaka Python dan Selenium yang berlaku untuk semua canary
- Kelas dan fungsi pustaka Python dan Selenium yang hanya berlaku untuk canary UI

Kelas dan fungsi pustaka Python dan Selenium yang berlaku untuk semua canary

Fungsi perpustakaan CloudWatch Synthetics Selenium berikut untuk Python berguna untuk semua burung kenari.

Topik

- SyntheticsConfiguration kelas
- SyntheticsLogger kelas

SyntheticsConfiguration kelas

Anda dapat menggunakan SyntheticsConfiguration kelas untuk mengkonfigurasi perilaku fungsi perpustakaan Synthetics. Misalnya, Anda dapat menggunakan kelas ini untuk mengonfigurasi fungsi `executeStep()` untuk tidak menangkap tangkapan layar.

Anda dapat mengatur konfigurasi CloudWatch Synthetics di tingkat global.

Definisi fungsi:

set_config(options)

```
from aws_synthetics.common import synthetics_configuration
```

options adalah sebuah objek, yang merupakan kumpulan opsi yang dapat dikonfigurasi untuk canary Anda. Bagian berikut menjelaskan bidang-bidang yang memungkinkan dalam *options*.

- `screenshot_on_step_start` (boolean)— Apakah mengambil tangkapan layar sebelum memulai langkah.
- `screenshot_on_step_success` (boolean)— Apakah mengambil tangkapan layar setelah menyelesaikan langkah yang berhasil.
- `screenshot_on_step_failure` (boolean)— Apakah mengambil tangkapan layar setelah langkah gagal.

with_screenshot_on_step_start(screenshot_on_step_start)

Menerima argumen Boolean, yang menunjukkan apakah akan mengambil tangkapan layar sebelum memulai sebuah langkah.

with_screenshot_on_step_success(screenshot_on_step_success)

Menerima argumen Boolean, yang menunjukkan apakah akan mengambil tangkapan layar setelah menyelesaikan langkah dengan sukses.

with_screenshot_on_step_failure(screenshot_on_step_failure)

Menerima argumen Boolean, yang menunjukkan apakah akan mengambil tangkapan layar setelah langkah gagal.

get_screenshot_on_step_start()

Mengembalikan apakah mengambil tangkapan layar sebelum memulai langkah.

get_screenshot_on_step_success()

Mengembalikan apakah mengambil tangkapan layar setelah menyelesaikan sebuah langkah dengan sukses.

get_screenshot_on_step_failure()

Mengembalikan apakah mengambil tangkapan layar setelah langkah gagal.

disable_step_screenshots()

Menonaktifkan semua pilihan tangkapan layar (get_screenshot_on_step_start, get_screenshot_on_step_success, dan get_screenshot_on_step_failure)

enable_step_screenshots()

Mengaktifkan semua pilihan tangkapan layar (get_screenshot_on_step_start, get_screenshot_on_step_success, dan get_screenshot_on_step_failure). Secara bawaan, semua metode ini diaktifkan.

setConfig (opsi) mengenai metrik CloudWatch

Untuk canary yang menggunakan `syn-python-selenium-1.1` atau yang lebih baru, (options) untuk setConfig dapat mencakup parameter Boolean berikut yang menentukan metrik mana yang diterbitkan oleh canary. Bawaan untuk masing-masing opsi ini adalah `true`. Pilihan yang dimulai dengan `aggregated` menentukan apakah metrik dipancarkan tanpa dimensi `CanaryName`. Anda dapat menggunakan metrik ini untuk melihat hasil gabungan untuk semua canary Anda. Pilihan lain menentukan apakah metrik dipancarkan dengan dimensi `CanaryName`. Anda dapat menggunakan metrik ini untuk melihat hasil untuk setiap canary individu.

Untuk daftar CloudWatch metrik yang dipancarkan oleh burung kenari, lihat. CloudWatch metrik yang diterbitkan oleh kenari

- `failed_canary_metric` (boolean)— Apakah akan memancarkan metrik `Failed` (dengan dimensi `CanaryName`) untuk canary ini. Nilai default-nya `true`.
- `failed_requests_metric` (boolean)— Apakah akan memancarkan metrik `Failed requests` (dengan dimensi `CanaryName`) untuk canary ini. Nilai default-nya `true`.
- `2xx_metric` (boolean)— Apakah akan memancarkan metrik `2xx` (dengan dimensi `CanaryName`) untuk canary ini. Nilai default-nya `true`.
- `4xx_metric` (boolean)— Apakah akan memancarkan metrik `4xx` (dengan dimensi `CanaryName`) untuk canary ini. Nilai default-nya `true`.
- `5xx_metric` (boolean)— Apakah akan memancarkan metrik `5xx` (dengan dimensi `CanaryName`) untuk canary ini. Nilai default-nya `true`.
- `step_duration_metric` (boolean)— Apakah akan memancarkan metrik `Step duration` (dengan dimensi `CanaryName StepName`) untuk canary ini. Bawaannya adalah `true`.
- `step_success_metric` (boolean)— Apakah akan memancarkan metrik `Step success` (dengan dimensi `CanaryName StepName`) untuk canary ini. Bawaannya adalah `true`.
- `aggregated_failed_canary_metric` (boolean)— Apakah akan memancarkan metrik `Failed` (tanpa dimensi `CanaryName`) untuk canary ini. Bawaannya adalah `true`.

- `aggregated_failed_requests_metric` (boolean)— Apakah akan memancarkan metrik `Failed Requests` (tanpa dimensi `CanaryName`) untuk canary ini. Bawaannya adalah `true`.
- `aggregated_2xx_metric` (boolean)— Apakah akan memancarkan metrik `2xx` (tanpa dimensi `CanaryName`) untuk canary ini. Bawaannya adalah `true`.
- `aggregated_4xx_metric` (boolean)— Apakah akan memancarkan metrik `4xx` (tanpa dimensi `CanaryName`) untuk canary ini. Bawaannya adalah `true`.
- `aggregated_5xx_metric` (boolean)— Apakah akan memancarkan metrik `5xx` (tanpa dimensi `CanaryName`) untuk canary ini. Bawaannya adalah `true`.

with_2xx_metric(2xx_metric)

Menerima argumen Boolean, yang menentukan apakah akan memancarkan metrik `2xx` dengan dimensi `CanaryName` untuk canary ini.

with_4xx_metric(4xx_metric)

Menerima argumen Boolean, yang menentukan apakah akan memancarkan metrik `4xx` dengan dimensi `CanaryName` untuk canary ini.

with_5xx_metric(5xx_metric)

Menerima argumen Boolean, yang menentukan apakah akan memancarkan metrik `5xx` dengan dimensi `CanaryName` untuk canary ini.

withAggregated2xxMetric(aggregated2xxMetric)

Menerima argumen Boolean, yang menentukan apakah akan memancarkan metrik `2xx` tanpa dimensi untuk canary ini.

withAggregated4xxMetric(aggregated4xxMetric)

Menerima argumen Boolean, yang menentukan apakah akan memancarkan metrik `4xx` tanpa dimensi untuk canary ini.

with_aggregated_5xx_metric(aggregated_5xx_metric)

Menerima argumen Boolean, yang menentukan apakah akan memancarkan metrik `5xx` tanpa dimensi untuk canary ini.

with_aggregated_failed_canary_metric(aggregated_failed_canary_metric)

Menerima argumen Boolean, yang menentukan apakah akan memancarkan metrik `Failed` tanpa dimensi untuk canary ini.

with_aggregated_failed_requests_metric(aggregated_failed_requests_metric)

Menerima argumen Boolean, yang menentukan apakah akan memancarkan metrik `Failed requests` tanpa dimensi untuk canary ini.

with_failed_canary_metric(failed_canary_metric)

Menerima argumen Boolean, yang menentukan apakah akan memancarkan metrik `Failed` dengan dimensi `CanaryName` untuk canary ini.

with_failed_requests_metric(failed_requests_metric)

Menerima argumen Boolean, yang menentukan apakah akan memancarkan metrik `Failed requests` dengan dimensi `CanaryName` untuk canary ini.

with_step_duration_metric(step_duration_metric)

Menerima argumen Boolean, yang menentukan apakah akan memancarkan metrik `Duration` dengan dimensi `CanaryName` untuk canary ini.

with_step_success_metric(step_success_metric)

Menerima argumen Boolean, yang menentukan apakah akan memancarkan metrik `StepSuccess` dengan dimensi `CanaryName` untuk canary ini.

Metode untuk mengaktifkan atau menonaktifkan metrik

disable_aggregated_request_metrics()

Menonaktifkan canary dari memancarkan semua metrik permintaan yang dipancarkan tanpa dimensi `CanaryName`.

disable_request_metrics()

Menonaktifkan semua metrik permintaan, termasuk metrik per canary dan metrik yang dikumpulkan di semua canary.

disable_step_metrics()

Menonaktifkan semua metrik langkah, termasuk metrik langkah sukses dan metrik durasi langkah.

enable_aggregated_request_metrics()

Mengaktifkan canary untuk memancarkan semua metrik permintaan yang dipancarkan tanpa dimensi `CanaryName`.

enable_request_metrics()

Mengaktifkan semua metrik permintaan, termasuk metrik per canary dan metrik yang dikumpulkan di semua canary.

enable_step_metrics()

Mengaktifkan semua metrik langkah, termasuk metrik keberhasilan langkah dan metrik durasi langkah.

Penggunaan di canary UI

Pertama, impor dependensi synthetics dan ambil konfigurasinya. Lalu, atur konfigurasi untuk setiap pilihan dengan memanggil metode setConfig menggunakan salah satu pilihan berikut.

```
from aws_synthetics.common import synthetics_configuration

synthetics_configuration.set_config(
     {
        "screenshot_on_step_start": False,
        "screenshot_on_step_success": False,
        "screenshot_on_step_failure": True
     }
)

or
```

Atau

```
synthetics_configuration.with_screenshot_on_step_start(False).with_screenshot_on_step_success(F
```

Untuk menonaktifkan semua tangkapan layar, gunakan fungsi disableStepScreenshots () seperti pada contoh ini.

```
synthetics_configuration.disable_step_screenshots()
```

Anda dapat mengaktifkan dan menonaktifkan tangkapan layar kapan saja di kode tersebut. Misalnya, untuk menonaktifkan tangkapan layar hanya untuk satu langkah, nonaktifkan sebelum menjalankan langkah tersebut dan kemudian aktifkan setelah langkah.

set_config(options) untuk canary UI

Dimulai dengan `syn-python-selenium-1.1`, untuk canary UI, `set_config` dapat mencakup parameter Boolean berikut:

- `continue_on_step_failure` (boolean)— Apakah akan melanjutkan menjalankan skrip canary setelah langkah gagal (ini mengacu pada fungsi executeStep). Jika ada langkah yang gagal, proses canary akan tetap ditandai sebagai gagal. Bawaannya adalah `false`.

SyntheticsLogger kelas

`synthetics_logger` menulis log keluar ke konsol dan ke file log lokal pada tingkat log yang sama. File log ini ditulis ke kedua lokasi hanya jika tingkat log berada pada atau di bawah tingkat log yang diinginkan dari fungsi log yang dipanggil tersebut.

Pernyataan pembuatan log di file log lokal diawali dengan "DEBUG: ", "INFO: ", dan seterusnya untuk mencocokkan tingkat log dari fungsi yang dipanggil tersebut.

Menggunakan `synthetics_logger` tidak diperlukan untuk membuat file log yang diunggah ke lokasi hasil Amazon S3 Anda. Sebagai penggantinya, Anda dapat membuat file log lain di `/tmp` folder. File apa pun yang dibuat di bawah folder `/tmp` diunggah ke lokasi hasil dalam bucket S3 sebagai artefak.

Untuk menggunakan `synthetics_logger`:

```
from aws_synthetics.common import synthetics_logger
```

Definisi fungsi yang berguna:

Dapatkan tingkat log:

```
log_level = synthetics_logger.get_level()
```

Atur tingkat log:

```
synthetics_logger.set_level()
```

Buat log sebuah pesan dengan tingkat tertentu. Tingkat dapat `DEBUG`, `INFO`, `WARN`, atau `ERROR`, seperti dalam contoh sintaks berikut:

```
synthetics_logger.debug(message, *args, **kwargs)
```

```
synthetics_logger.info(message, *args, **kwargs)
```

```
synthetics_logger.log(message, *args, **kwargs)
```

```
synthetics_logger.warn(message, *args, **kwargs)
```

```
synthetics_logger.error(message, *args, **kwargs)
```

Untuk informasi tentang parameter debug, silakan lihat dokumentasi Python standar di logging.debug

Dalam fungsi pembuatan log ini, `message` adalah string format pesan. `args` adalah argumen yang digabung menjadi `msg` menggunakan operator pemformatan string.

Ada tiga argumen kata kunci di `kwargs`:

- `exc_info`– Jika tidak dievaluasi sebagai false, tambahkan informasi pengecualian ke pesan pembuatan log.
- `stack_info`– default ke false. Jika true, tambahkan informasi tumpukan ke pesan pembuatan log, termasuk panggilan pembuatan log yang sebenarnya.
- `extra`– Argumen kata kunci opsional ketiga, yang dapat Anda gunakan untuk meneruskan kamus yang digunakan untuk mengisi `__dict__` dari `LogRecord` yang dibuat untuk peristiwa pembuatan log dengan atribut yang ditentukan pengguna.

Contoh:

Buat log pesan dengan tingkat `DEBUG`:

```
synthetics_logger.debug('Starting step - login.')
```

Buat log pesan dengan tingkat `INFO`. `logger.log` adalah sinonim untuk `logger.info`:

```
synthetics_logger.info('Successfully completed step - login.')
```

atau

```
synthetics_logger.log('Successfully completed step - login.')
```

Buat log pesan dengan tingkat `WARN`:

```
synthetics_logger.warn('Warning encountered trying to publish %s', 'CloudWatch Metric')
```

Buat log pesan dengan tingkat `ERROR`:

```
synthetics_logger.error('Error encountered trying to publish %s', 'CloudWatch Metric')
```

Buat log pengecualian:

```
synthetics_logger.exception(message, *args, **kwargs)
```

Membuat log pesan dengan tingkat `ERROR`. Informasi pengecualian ditambahkan ke pesan pembuatan log. Anda harus memanggil fungsi ini hanya dari handler pengecualian.

Untuk informasi tentang parameter pengecualian, silakan lihat dokumentasi Python standar di logging.exception

`message` adalah string format pesan. `args` adalah argumen, yang digabungkan menjadi `msg` menggunakan operator pemformatan string.

Ada tiga argumen kata kunci di `kwargs`:

- `exc_info`– Jika tidak dievaluasi sebagai false, tambahkan informasi pengecualian ke pesan pembuatan log.
- `stack_info`– default ke false. Jika true, tambahkan informasi tumpukan ke pesan pembuatan log, termasuk panggilan pembuatan log yang sebenarnya.

- `extra`– Argumen kata kunci opsional ketiga, yang dapat Anda gunakan untuk meneruskan kamus yang digunakan untuk mengisi `__dict__` dari `LogRecord` yang dibuat untuk peristiwa pembuatan log dengan atribut yang ditentukan pengguna.

Contoh:

```
synthetics_logger.exception('Error encountered trying to publish %s', 'CloudWatch
 Metric')
```

Kelas dan fungsi pustaka Python dan Selenium yang hanya berlaku untuk canary UI

Fungsi perpustakaan CloudWatch Synthetics Selenium berikut untuk Python hanya berguna untuk kenari UI.

Topik

- SyntheticsBrowser kelas
- SyntheticsWebDriver kelas

SyntheticsBrowser kelas

Ketika Anda membuat instans browser dengan memanggil `synthetics_webdriver.Chrome()`, instans browser yang dikembalikan bertipe `SyntheticsBrowser`. `SyntheticsBrowser`Kelas mengontrol ChromeDriver, dan memungkinkan skrip kenari untuk menggerakkan browser, memungkinkan Selenium WebDriver bekerja dengan Synthetics.

Selain metode Selenium standar, itu juga menyediakan metode berikut.

set_viewport_size(width, height)

Mengatur viewport browser. Contoh:

```
browser.set_viewport_size(1920, 1080)
```

save_screenshot(filename, suffix)

Menyimpan tangkapan layar ke direktori `/tmp`. Tangkapan layar diunggah dari sana ke folder artefak canary di dalam bucket S3.

filename adalah nama file untuk tangkapan layar, dan suffix adalah string opsional untuk digunakan guna menamai tangkapan layar.

Contoh:

```
browser.save_screenshot('loaded.png', 'page1')
```

SyntheticsWebDriver kelas

Untuk menggunakan kelas ini, gunakan yang berikut ini dalam skrip Anda:

```
from aws_synthetics.selenium import synthetics_webdriver
```

add_execution_error(errorMessage, ex);

`errorMessage` menjelaskan kesalahan dan `ex` adalah pengecualian yang ditemui

Anda dapat menggunakan `add_execution_error` untuk mengatur kesalahan eksekusi bagi canary Anda. Itu menggagalkan canary tanpa mengganggu eksekusi skrip. Itu juga tidak memengaruhi metrik `successPercent` Anda.

Anda harus melacak kesalahan sebagai kesalahan eksekusi hanya jika kesalahan itu tidak penting untuk menunjukkan keberhasilan atau kegagalan skrip canary Anda.

Contoh penggunaan `add_execution_error` adalah sebagai berikut. Anda memantau ketersediaan titik akhir Anda dan mengambil tangkapan layar setelah halaman dimuat. Karena kegagalan mengambil tangkapan layar tidak menentukan ketersediaan titik akhir, Anda dapat menangkap kesalahan apa pun yang dihadapi saat mengambil tangkapan layar dan menambahkannya sebagai kesalahan eksekusi. Metrik ketersediaan Anda masih akan menunjukkan bahwa titik akhir telah aktif dan berjalan, namun status canary Anda akan ditandai sebagai gagal. Blok kode sampel berikut menangkap kesalahan tersebut dan menambahkannya sebagai kesalahan eksekusi.

```
try:
    browser.save_screenshot("loaded.png")
except Exception as ex:
   self.add_execution_error("Unable to take screenshot", ex)
```

add_user_agent(user_agent_str)

Menambahkan nilai dari `user_agent_str` ke header agen pengguna browser. Anda harus menetapkan `user_agent_str` sebelum membuat instans browser.

Contoh:

```
synthetics_webdriver.add_user_agent('MyApp-1.0')
```

execute_step(step_name, function_to_execute)

Memproses satu fungsi. Ini juga melakukan hal berikut:

- Membuat log bahwa langkah tersebut dimulai.
- Mengambil tangkapan layar yang bernama `<stepName>-starting`.
- Memulai pengatur waktu.
- Menjalankan fungsi yang disediakan.
- Jika fungsi kembali secara normal, ini dihitung sebagai lulus. Jika fungsi tersebut throw, maka ini dihitung sebagai gagal.
- Mengakhiri pengatur waktu.
- Mencatat log jika langkah berhasil atau gagal
- Mengambil tangkapan layar yang bernama `<stepName>-succeeded` atau `<stepName>-failed`.
- Memancarkan metrik `stepName SuccessPercent`, 100 untuk lulus atau 0 untuk gagal.
- Memancarkan metrik `stepName Duration`, dengan nilai didasarkan pada mulai melangkah dan waktu selesai.
- Akhirnya, mengembalikan apa yang dikembalikan oleh `functionToExecute` atau melempar kembali apa yang dilempar oleh `functionToExecute`.

Contoh:

```
from selenium.webdriver.common.by import By

def custom_actions():
        #verify contains
        browser.find_element(By.XPATH, "//*[@id=\"id_1\"][contains(text(),'login')]")
        #click a button
        browser.find_element(By.XPATH, '//*[@id="submit"]/a').click()

    await synthetics_webdriver.execute_step("verify_click", custom_actions)
```

Chrome()

Meluncurkan sebuah instans dari browser Chromium dan mengembalikan instans yang dibuat dari browser.

Contoh:

```
browser = synthetics_webdriver.Chrome()
browser.get("https://example.com/)
```

Untuk meluncurkan browser dalam mode penyamaran, gunakan yang berikut ini:

```
add_argument('—-incognito')
```

Untuk menambahkan pengaturan proxy, gunakan yang berikut ini:

```
add_argument('--proxy-server=%s' % PROXY)
```

Contoh:

```
from selenium.webdriver.chrome.options import Options
chrome_options = Options()
chrome_options.add_argument("—-incognito")
browser = syn_webdriver.Chrome(chrome_options=chrome_options)
```

Penjadwalan operasi canary yang menggunakan cron

Menggunakan ekspresi cron memberi Anda fleksibilitas saat Anda menjadwalkan canary. Ekspresi cron berisi lima atau enam bidang dalam urutan yang tercantum dalam tabel berikut. Bidang dipisahkan oleh spasi. Sintaksnya berbeda tergantung pada apakah Anda menggunakan CloudWatch konsol untuk membuat kenari, atau atau SDK AWS CLI . AWS Bila Anda menggunakan konsol, Anda menentukan hanya lima bidang pertama. Bila Anda menggunakan AWS CLI atau AWS SDK, Anda menentukan semua enam bidang, dan Anda harus menentukan * untuk `Year` bidang tersebut.

Bidang	Nilai yang diizinkan	Karakter khusus yang diizinkan
Menit	0-59	, - * /

Bidang	Nilai yang diizinkan	Karakter khusus yang diizinkan
Jam	0-23	, - * /
D ay-of-month	1-31	, - * ? / L W
Bulan	1-12 atau JAN-DES	, - * /
D ay-of-week	1-7 atau MGG-SBT	, - * ? L #
Tahun	*	

Karakter-karakter khusus

- , (koma) mencakup beberapa nilai dalam ekspresi untuk suatu bidang. Misalnya, di bidang Bulan, JAN,FEB,MAR akan menyertakan Januari, Februari, dan Maret.
- - (tanda hubung) karakter khusus menentukan rentang. Di bidang Tanggal, 1-15 akan mencakup tanggal 1 hingga 15 pada bulan yang ditentukan.
- * (bintang) mencakup semua nilai di bidang tersebut. Di bidang Jam, * mencakup setiap jam. Anda tidak dapat menggunakan* di ay-of-week bidang D ay-of-month dan D dalam ekspresi yang sama. Jika Anda menggunakannya di satu bidang, Anda harus menggunakan ? di bidang lain.
- / (garis miring) menentukan kenaikan. Di bidang Menit, Anda dapat memasukkan 1/10 untuk menentukan setiap menit kesepuluh, mulai dari menit pertama jam (sebagai contoh, menit kesebelas, dua puluh satu, dan tiga puluh satu, dan seterusnya).
- ? (tanda tanya) menentukan satu atau yang lain. Jika Anda memasukkan 7 di ay-of-month bidang D dan Anda tidak peduli hari apa dalam minggu ketujuh, Anda dapat masuk? di ay-of-week bidang D.
- Wildcard L di ay-of-week bidang D ay-of-month atau D menentukan hari terakhir bulan atau minggu.
- **W**Wildcard di ay-of-month bidang D menentukan hari kerja. Di ay-of-month bidang D, **3W** tentukan hari kerja yang paling dekat dengan hari ketiga bulan itu.
- Wildcard # di ay-of-week bidang D menentukan contoh tertentu dari hari yang ditentukan dalam seminggu dalam sebulan. Sebagai contoh, 3#2 adalah hari Selasa kedua setiap bulan. Angka 3 mengacu pada hari Selasa karena itu adalah hari ketiga setiap minggu, dan angka 2 mengacu pada hari kedua dari jenis tersebut dalam bulan tersebut.

Keterbatasan:

- Anda tidak dapat menentukan ay-of-week bidang D ay-of-month dan D dalam ekspresi cron yang sama. Jika Anda menentukan nilai atau * (bintang) di salah satu bidang, Anda harus menggunakan ? (tanda tanya) di bidang lain.
- Ekspresi cron yang mengarah ke tingkat lebih cepat dari satu menit tidak didukung.
- Anda tidak dapat mengatur canary untuk menunggu lebih dari satu tahun sebelum berjalan, sehingga Anda dapat menentukan hanya * dalam bidang `Year`.

Contoh

Anda dapat merujuk ke string cron sampel berikut ketika Anda membuat canary. Contoh berikut adalah sintaks yang benar untuk menggunakan AWS CLI atau AWS SDK untuk membuat atau memperbarui kenari. Jika Anda menggunakan CloudWatch konsol, hilangkan final * di setiap contoh.

Ekspresi	Arti
`0 10 * * ? *`	Jalankan pada pukul 10:00 (UTC) setiap hari
`15 12 * * ? *`	Jalankan pada pukul 12.15 (UTC) setiap hari
`0 18 ? * MON-FRI *`	Jalankan pada pukul 18.00 (UTC) setiap Senin hingga Jumat
`0 8 1 * ? *`	Jalankan pada pukul 08.00 (UTC) pada hari pertama setiap bulan
`0/10 * ? * MON-SAT *`	Jalankan setiap 10 menit Senin hingga Sabtu setiap minggu
`0/5 8-17 ? * MON-FRI *`	Jalankan setiap lima menit Senin hingga Jumat antara pukul 08.00 dan 17.55 (UTC)

Grup

Anda dapat membuat grup untuk mengasosiasikan canary satu sama lain, termasuk canary lintas wilayah. Menggunakan grup dapat membantu Anda mengelola dan mengotomatiskan canary Anda, dan Anda juga dapat melihat hasil dan statistik run gabungan untuk semua canary dalam sebuah grup.

Grup adalah sumber daya global. Saat Anda membuat grup, grup tersebut direplikasi di semua AWS Wilayah yang mendukung grup, dan Anda dapat menambahkan kenari dari salah satu Wilayah ini ke dalamnya, dan melihatnya di salah satu Wilayah ini. Meskipun format ARN grup mencerminkan nama Wilayah tempat itu dibuat, grup tidak dibatasi ke Wilayah mana pun. Ini berarti Anda dapat menempatkan canary dari beberapa Wilayah ke dalam grup yang sama, dan kemudian menggunakan grup itu untuk melihat dan mengelola semua canary tersebut dalam satu tampilan.

Grup didukung di semua Wilayah kecuali Wilayah yang dinonaktifkan secara default. Untuk informasi selengkapnya tentang Wilayah ini, silakan lihat Mengaktifkan sebuah Wilayah.

Setiap kelompok dapat berisi sebanyak 10 canary. Anda dapat memiliki sebanyak 20 grup di akun Anda. Setiap canary tunggal dapat menjadi anggota hingga 10 grup.

Untuk membuat grup

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Application Signals, Synthetics Canaries.
3. Pilih Buat group.
4. Di bawah Nama Grup, masukkan nama untuk grup tersebut.
5. Pilih canary untuk dikaitkan dengan grup ini. Untuk memilih canary, ketik nama lengkapnya di Nama canary yang tepat dan pilih Cari. Kemudian pilih kotak centang di samping nama canary. Jika ada beberapa canary dengan nama yang sama di Wilayah yang berbeda, pastikan untuk memilih canary yang Anda inginkan.

   Anda dapat mengulangi langkah ini untuk mengasosiasikan sebanyak 10 canary dengan grup tersebut.
6. (Opsional) Di bawah Tag, tambahkan satu atau beberapa pasangan nilai kunci sebagai tanda untuk grup ini. Tag dapat membantu Anda mengidentifikasi dan mengatur AWS sumber daya Anda dan melacak AWS biaya Anda. Untuk informasi selengkapnya, lihat Pemberian tag pada sumber daya Amazon CloudWatch Anda.
7. Pilih Buat group.

Memecahkan masalah canary yang gagal

Jika canary Anda gagal, periksa hal berikut ini untuk pemecahan masalah.

Pemecahan masalah umum

- Gunakan halaman detail canary untuk menemukan informasi lebih lanjut. Di CloudWatch konsol, pilih Canary di panel navigasi dan kemudian pilih nama kenari untuk membuka halaman detail kenari. Di tab Ketersediaan, periksa SuccessPercentmetrik untuk melihat apakah masalahnya konstan atau terputus-putus.

  Sementara masih di tab Ketersediaan, pilih titik data yang gagal untuk melihat tangkapan layar, log, dan laporan langkah (jika tersedia) untuk proses yang gagal tersebut.

  Jika laporan langkah tersedia karena langkah-langkah adalah bagian dari skrip Anda, periksa untuk melihat langkah mana yang telah gagal dan lihat tangkapan layar terkait untuk melihat masalah yang dilihat pelanggan Anda.

  Anda juga dapat memeriksa file HAR untuk melihat apakah satu atau beberapa permintaan gagal. Anda dapat menggali lebih dalam dengan menggunakan log untuk menelusuri permintaan dan kesalahan yang gagal. Akhirnya, Anda dapat membandingkan artefak ini dengan artefak dari operasi canary yang berhasil untuk menunjukkan masalah.

  Secara default, CloudWatch Synthetics menangkap tangkapan layar untuk setiap langkah dalam kenari UI. Namun demikian, skrip Anda mungkin dikonfigurasi untuk menonaktifkan tangkapan layar. Selama proses debug, Anda mungkin ingin mengaktifkan tangkapan layar lagi. Demikian pula, untuk canary API Anda mungkin ingin melihat header dan isi permintaan dan respons HTTP selama proses debug. Untuk informasi tentang cara menyertakan data ini dalam laporan, silakan lihat executeHttpStep(StepName, requestOptions, [callback], [stepConfig]).
- Jika Anda memiliki deployment terbaru untuk aplikasi Anda, kembalikan ke keadaan semula dan kemudian lakukan debug nanti.
- Hubungkan ke titik akhir Anda secara manual untuk melihat apakah Anda dapat mereproduksi masalah yang sama.

Topik

- Canary gagal setelah pembaruan lingkungan Lambda
- Kenari saya diblokir oleh AWS WAF
- Menunggu elemen untuk muncul
- Simpul baik tidak terlihat atau bukan merupakan HTMLElement untuk page.click()
- Tidak dapat mengunggah artefak ke S3, Pengecualian: Tidak dapat mengambil lokasi bucket S3: Akses Ditolak
- Kesalahan: Kesalahan protokol (Runtime. callFunctionOn): Target ditutup.

- Canary Gagal. Kesalahan: Tidak ada datapoint - Canary Menunjukkan kesalahan batas waktu
- Coba untuk mengakses titik akhir internal
- Masalah peningkatan dan penurunan versi runtime Canary
- Masalah berbagi permintaan lintas asal (CORS)
- Pemecahan masalah canary di VPC

Canary gagal setelah pembaruan lingkungan Lambda

CloudWatch Canary Synthetics diimplementasikan sebagai fungsi Lambda di akun Anda. Fungsi Lambda ini tunduk pada pembaruan runtime Lambda reguler yang berisi pembaruan keamanan, perbaikan bug, dan peningkatan lainnya. Lambda berusaha untuk menyediakan pembaruan runtime yang kompatibel dengan fungsi yang ada. Namun, seperti halnya patching perangkat lunak, ada kasus yang jarang terjadi di mana pembaruan runtime dapat berdampak negatif pada fungsi yang ada. Jika Anda yakin kenari Anda telah terpengaruh oleh pembaruan runtime Lambda, Anda dapat menggunakan mode manual manajemen runtime Lambda (di Wilayah yang didukung) untuk memutar kembali versi runtime Lambda sementara. Ini membuat fungsi kenari Anda tetap berfungsi dan meminimalkan gangguan, memberikan waktu untuk memperbaiki ketidakcocokan sebelum kembali ke versi runtime terbaru.

Jika kenari Anda gagal setelah pembaruan runtime Lambda, solusi terbaik adalah meningkatkan ke salah satu runtime Synthetics terbaru. Untuk informasi selengkapnya tentang runtime terbaru, lihatVersi runtime Synthetics.

Sebagai solusi alternatif, di Wilayah di mana kontrol manajemen runtime Lambda tersedia, Anda dapat mengembalikan kenari kembali ke runtime terkelola Lambda yang lebih lama, menggunakan mode manual untuk kontrol manajemen runtime. Anda dapat mengatur mode manual menggunakan AWS CLI atau dengan menggunakan konsol Lambda, menggunakan langkah-langkah di bawah ini di bagian berikut.

> **Warning**
>
> Saat Anda mengubah pengaturan runtime ke mode manual, fungsi Lambda Anda tidak akan menerima pembaruan keamanan otomatis hingga dikembalikan ke mode Otomatis. Selama periode ini, fungsi Lambda Anda mungkin rentan terhadap kerentanan keamanan.

Prasyarat

- Instal jq
- Instal versi terbaru dari file AWS CLI. Untuk informasi selengkapnya, lihat petunjuk AWS CLI pemasangan dan perbarui.

Langkah 1: Dapatkan ARN fungsi Lambda

Jalankan perintah berikut untuk mengambil `EngineArn` bidang dari respons. Ini `EngineArn` adalah ARN dari fungsi Lambda yang dikaitkan dengan kenari. Anda akan menggunakan ARN ini dalam langkah-langkah berikut.

```
aws synthetics get-canary --name my-canary | jq '.Canary.EngineArn'
```

Contoh keluaran dari`EngingArn`:

```
"arn:aws:lambda:us-west-2:123456789012:function:cwsyn-my-canary-dc5015c2-db17-4cb5-
afb1-EXAMPLE991:8"
```

Langkah 2: Dapatkan ARN versi runtime Lambda terakhir yang bagus

Untuk membantu memahami apakah kenari Anda terkena dampak pembaruan runtime Lambda, periksa apakah tanggal dan waktu perubahan ARN versi runtime Lambda di log Anda muncul pada tanggal dan waktu ketika Anda melihat dampak pada kenari Anda. Jika tidak cocok, itu mungkin bukan pembaruan runtime Lambda yang menyebabkan masalah Anda.

Jika kenari Anda dipengaruhi oleh pembaruan runtime Lambda, Anda harus mengidentifikasi ARN dari versi runtime Lambda yang berfungsi yang sebelumnya Anda gunakan. Ikuti petunjuk dalam Mengidentifikasi perubahan versi runtime untuk menemukan ARN dari runtime sebelumnya. Rekam ARN versi runtime, dan lanjutkan ke Langkah 3. untuk mengatur konfigurasi manajemen runtime.

Jika kenari Anda belum terpengaruh oleh pembaruan lingkungan Lambda, maka Anda dapat menemukan ARN dari versi runtime Lambda yang saat ini Anda gunakan. Jalankan perintah berikut untuk mengambil fungsi Lambda dari respons. `RuntimeVersionArn`

```
aws lambda get-function-configuration \
--function-name "arn:aws:lambda:us-west-2:123456789012:function:cwsyn-my-canary-
dc5015c2-db17-4cb5-afb1-EXAMPLE991:8" | jq '.RuntimeVersionConfig.RuntimeVersionArn'
```

Contoh keluaran dari`RuntimeVersionArn`:

```
"arn:aws:lambda:us-
west-2::runtime:EXAMPLE647b82f490a45d7ddd96b557b916a30128d9dcab5f4972911ec0f"
```

Langkah 3: Memperbarui konfigurasi manajemen runtime Lambda

Anda dapat menggunakan konsol Lambda AWS CLI atau Lambda untuk memperbarui konfigurasi manajemen runtime.

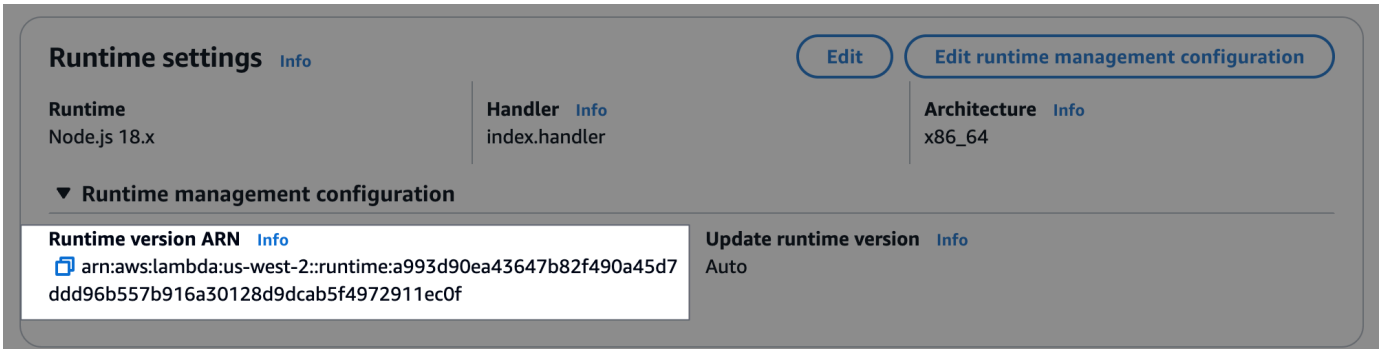
Untuk mengatur mode manual konfigurasi manajemen runtime Lambda menggunakan AWS CLI

Masukkan perintah berikut untuk mengubah manajemen runtime fungsi Lambda ke mode manual. Pastikan untuk mengganti *fungsi-nama* dan *qualifier dengan* fungsi Lambda ARN dan nomor versi fungsi Lambda masing-masing, menggunakan nilai yang Anda temukan di Langkah 1. Juga ganti *runtime-version-arn*dengan versi ARN yang Anda temukan di Langkah 2.

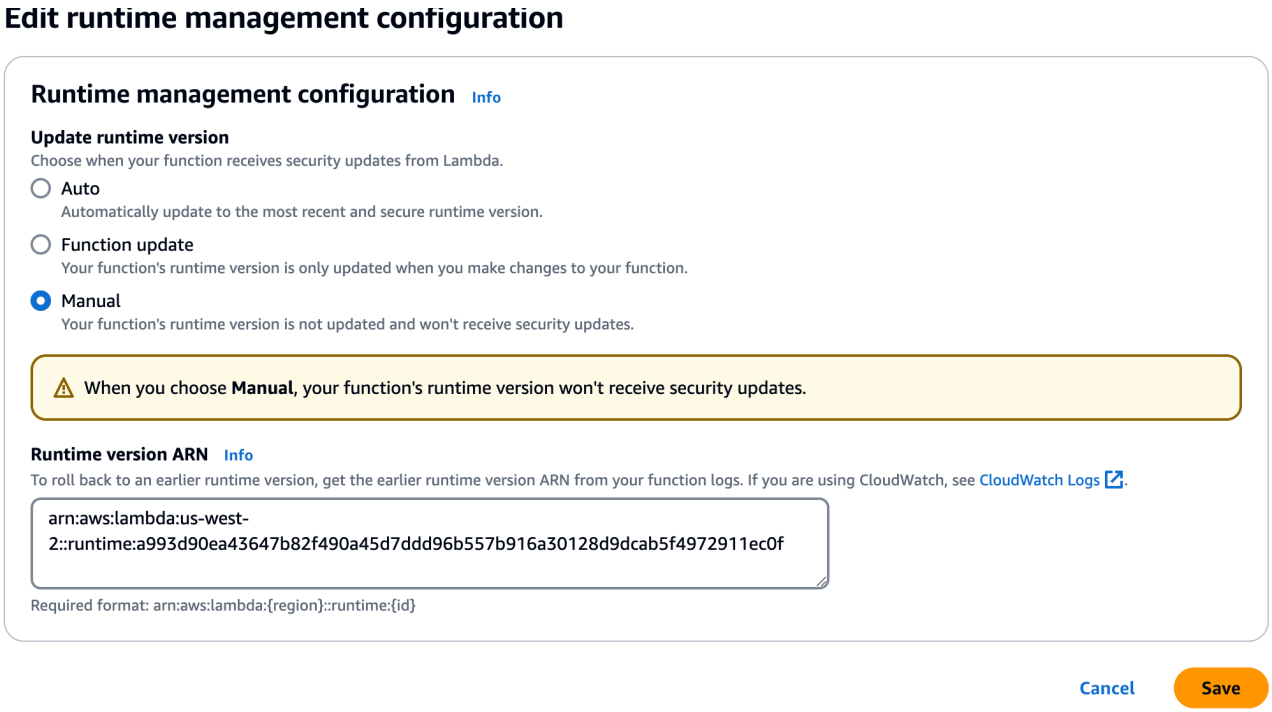
```
aws lambda put-runtime-management-config \
    --function-name "arn:aws:lambda:us-west-2:123456789012:function:cwsyn-my-canary-
dc5015c2-db17-4cb5-afb1-EXAMPLE991" \
    --qualifier 8 \
    --update-runtime-on "Manual" \
    --runtime-version-arn "arn:aws:lambda:us-
west-2::runtime:a993d90ea43647b82f490a45d7ddd96b557b916a30128d9dcab5f4972911ec0f"
```

Untuk mengubah kenari ke mode manual menggunakan konsol Lambda

1. Buka AWS Lambda konsol di https://console.aws.amazon.com/lambda/.
2. Pilih tab Versi, pilih tautan nomor versi yang sesuai dengan ARN Anda, dan pilih tab Kode.
3. Gulir ke bawah ke pengaturan Runtime, perluas konfigurasi manajemen Runtime, dan salin ARN versi Runtime.



4. Pilih Edit konfigurasi manajemen runtime, pilih Manual, tempel ARN versi runtime yang Anda salin sebelumnya ke bidang ARN versi Runtime. Lalu, pilih Simpan.



Kenari saya diblokir oleh AWS WAF

Untuk AWS WAF mencegah pemblokiran kenari Anda, siapkan kondisi kecocokan AWS WAF string yang memungkinkan string`CloudWatchSynthetics`. Untuk informasi selengkapnya, lihat Bekerja dengan kondisi pencocokan string dalam AWS WAF dokumentasi.

Menunggu elemen untuk muncul

Setelah menganalisis log dan tangkapan layar Anda, jika Anda melihat bahwa skrip Anda menunggu elemen muncul di layar dan waktu habis, periksa tangkapan layar yang relevan untuk melihat apakah elemen tersebut muncul di halaman. Verifikasi `xpath` Anda untuk memastikan bahwa itu benar.

Untuk masalah terkait Dalang, periksa halaman Dalang atau forum internet. GitHub

Simpul baik tidak terlihat atau bukan merupakan HTMLElement untuk page.click()

Jika simpul tidak terlihat atau bukan merupakan `HTMLElement` untuk `page.click()`, pertama verifikasi `xpath` yang Anda gunakan untuk mengklik elemen tersebut. Juga, jika elemen Anda berada di bagian bawah layar, sesuaikan viewport Anda. CloudWatch Synthetics secara default menggunakan viewport 1920 * 1080. Anda dapat mengatur viewport yang berbeda ketika Anda meluncurkan browser atau dengan menggunakan fungsi Puppeteer `page.setViewport`.

Tidak dapat mengunggah artefak ke S3, Pengecualian: Tidak dapat mengambil lokasi bucket S3: Akses Ditolak

Jika kenari Anda gagal karena kesalahan Amazon S3, CloudWatch Synthetics tidak dapat mengunggah tangkapan layar, log, atau laporan yang dibuat untuk kenari karena masalah izin. Periksa hal-hal berikut:

- Periksa apakah peran IAM canary memiliki izin `s3:ListAllMyBuckets`, izin `s3:GetBucketLocation` untuk bucker Amazon S3 yang benar, dan izin `s3:PutObject` untuk bucket tempat canary menyimpan artefaknya. Jika canary melakukan pemantauan visual, peran tersebut juga memerlukan izin `s3:GetObject` untuk bucket. Izin yang sama ini juga diperlukan dalam Kebijakan Titik Akhir Gateway Amazon VPC S3, jika canary di-deploy di VPC dengan titik akhir VPC.
- Jika kenari menggunakan kunci yang dikelola AWS KMS pelanggan untuk enkripsi alih-alih kunci AWS terkelola standar (default), peran IAM kenari mungkin tidak memiliki izin untuk mengenkripsi atau mendekripsi menggunakan kunci itu. Untuk informasi selengkapnya, lihat Mengenkripsi artefak canary.
- Kebijakan bucket Anda mungkin tidak mengizinkan mekanisme enkripsi yang digunakan canary. Sebagai contoh, jika kebijakan bucket Anda mengamanatkan untuk menggunakan mekanisme enkripsi tertentu atau kunci KMS, maka Anda harus memilih mode enkripsi yang sama untuk canary Anda.

Jika canary melakukan pemantauan visual, silakan lihat Memperbarui lokasi artefak dan enkripsi saat menggunakan pemantauan visual untuk informasi lebih lanjut.

Kesalahan: Kesalahan protokol (Runtime. callFunctionOn): Target ditutup.

Kesalahan ini muncul jika ada beberapa permintaan jaringan setelah halaman atau browser ditutup. Anda mungkin lupa untuk menunggu operasi asynchronous. Setelah menjalankan skrip Anda, CloudWatch Synthetics menutup browser. Eksekusi setiap operasi asynchronous setelah browser ditutup dapat menyebabkan `target closed error`.

Canary Gagal. Kesalahan: Tidak ada datapoint - Canary Menunjukkan kesalahan batas waktu

Ini berarti bahwa canary Anda berjalan melebihi batas waktu. Eksekusi kenari berhenti sebelum CloudWatch Synthetics dapat mempublikasikan metrik CloudWatch persen keberhasilan atau

memperbarui artefak seperti file HAR, log, dan tangkapan layar. Jika batas waktu Anda terlalu rendah, Anda dapat meningkatkannya.

Secara bawaan, nilai batas waktu canary sama dengan frekuensinya. Anda dapat secara manual menyesuaikan nilai batas waktu menjadi kurang dari atau sama dengan frekuensi canary. Jika frekuensi canary Anda rendah, Anda harus meningkatkan frekuensi untuk meningkatkan batas waktu. Anda dapat menyesuaikan frekuensi dan nilai batas waktu di bawah Jadwal saat Anda membuat atau memperbarui kenari dengan menggunakan konsol Synthetics CloudWatch .

Pastikan nilai batas waktu canary Anda tidak lebih singkat dari 15 detik untuk memungkinkan Lambda melakukan cold start dan waktu yang diperlukan untuk melakukan boot up instrumentasi canary.

Artefak kenari tidak tersedia untuk dilihat di konsol CloudWatch Synthetics saat kesalahan ini terjadi. Anda dapat menggunakan CloudWatch Log untuk melihat log kenari.

Untuk menggunakan CloudWatch Log untuk melihat log untuk kenari

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi kiri, pilih Grup log.
3. Temukan grup log dengan mengetik nama canary di kotak filter. Grup log untuk canary memiliki nama /aws/lambda/cwsyn-***canaryName***-randomId.

Coba untuk mengakses titik akhir internal

Jika Anda ingin kenari Anda mengakses titik akhir di jaringan internal Anda, kami sarankan Anda mengatur CloudWatch Synthetics untuk menggunakan VPC. Untuk informasi selengkapnya, lihat Menjalankan canary di VPC.

Masalah peningkatan dan penurunan versi runtime Canary

Jika Anda baru saja meningkatkan canary dari versi runtime `syn-1.0` ke versi yang lebih baru, itu mungkin merupakan masalah berbagi permintaan lintas asal (CORS). Untuk informasi selengkapnya, lihat Masalah berbagi permintaan lintas asal (CORS).

Jika Anda baru-baru ini menurunkan versi kenari ke versi runtime yang lebih lama, periksa untuk memastikan bahwa fungsi CloudWatch Synthetics yang Anda gunakan tersedia dalam versi runtime yang lebih lama yang Anda turunkan. Misalnya, fungsi `executeHttpStep` tersedia untuk versi runtime `syn-nodejs-2.2` dan yang lebih baru. Untuk memeriksa ketersediaan fungsi, silakan lihat Menulis skrip canary.

> **Note**
>
> Saat Anda berencana untuk memutakhirkan atau menurunkan versi runtime untuk kenari, sebaiknya Anda mengkloning kenari terlebih dahulu dan memperbarui versi runtime di kenari kloning. Setelah Anda telah memverifikasi bahwa klon dengan versi runtime baru bekerja, Anda dapat memperbarui versi runtime canary asli Anda dan menghapus klon tersebut.

Masalah berbagi permintaan lintas asal (CORS)

Dalam UI canary, jika beberapa permintaan jaringan gagal dengan `403` atau `net::ERR_FAILED`, periksa apakah canary memiliki pelacakan aktif yang diaktifkan dan juga gunakan fungsi Puppeteer `page.setExtraHTTPHeaders` untuk menambahkan header. Jika demikian, permintaan jaringan yang gagal mungkin disebabkan oleh pembatasan berbagi permintaan lintas asal (CORS). Anda dapat mengonfirmasi apakah ini terjadi dengan menonaktifkan pelacakan aktif atau menghapus header HTTP tambahan.

Mengapa hal ini terjadi?

Ketika pelacakan aktif digunakan, header tambahan ditambahkan ke semua permintaan keluar untuk melacak panggilan. Memodifikasi header permintaan dengan menambahkan header jejak atau menambahkan header tambahan menggunakan Puppeteer `page.setExtraHTTPHeaders` menyebabkan pemeriksaan CORS untuk permintaan XHTML (XHR). HttpRequest

Jika Anda tidak ingin menonaktifkan pelacakan aktif atau menghapus header tambahan, Anda dapat memperbarui aplikasi web Anda untuk mengizinkan akses lintas asal atau Anda dapat menonaktifkan keamanan web dengan menggunakan bendera `disable-web-security` saat Anda meluncurkan browser Chrome di skrip Anda.

Anda dapat mengganti parameter peluncuran yang digunakan oleh CloudWatch Synthetics dan meneruskan parameter flag `disable-web-security` tambahan dengan menggunakan fungsi peluncuran CloudWatch Synthetics. Untuk informasi selengkapnya, lihat Fungsi pustaka tersedia untuk skrip canary Node.js.

> **Note**
>
> Anda dapat mengganti parameter peluncuran yang digunakan oleh CloudWatch Synthetics saat Anda menggunakan versi runtime `syn-nodejs-2.1` atau yang lebih baru.

Pemecahan masalah canary di VPC

Jika Anda memiliki masalah setelah membuat atau memperbarui canary di VPC, salah satu bagian berikut dapat membantu Anda memecahkan masalah.

Canary baru dalam status kesalahan atau canary tidak dapat diperbarui

Jika Anda membuat canary untuk beroperasi di VPC dan segera masuk ke status error, atau Anda tidak dapat memperbarui canary untuk berjalan di VPC, peran canary tersebut mungkin tidak memiliki izin yang tepat. Untuk berjalan di VPC, canary harus memiliki izin `ec2:CreateNetworkInterface`, `ec2:DescribeNetworkInterfaces`, dan `ec2:DeleteNetworkInterface`. Semua izin ini disertakan dalam kebijakan terkelola `AWSLambdaVPCAccessExecutionRole`. Untuk informasi selengkapnya, silakan lihat Peran Eksekusi dan Izin Pengguna.

Jika masalah ini terjadi ketika Anda membuat canary, Anda harus menghapus canary tersebut, dan membuat canary yang baru. Jika Anda menggunakan CloudWatch konsol untuk membuat kenari baru, di bawah Izin Akses, pilih Buat peran baru. Peran baru yang mencakup semua izin yang diperlukan untuk menjalankan canary akan dibuat.

Jika masalah ini terjadi ketika Anda memperbarui canary, Anda dapat memperbarui canary lagi dan memberikan peran baru yang memiliki izin yang diperlukan.

Kesalahan "Tidak ada hasil uji yang dikembalikan"

Jika canary menampilkan kesalahan "tidak ada hasil uji yang dikembalikan", salah satu masalah berikut mungkin menjadi penyebabnya:

- Jika VPC Anda tidak memiliki akses internet, Anda harus menggunakan titik akhir VPC untuk memberikan akses kenari ke dan Amazon S3. CloudWatch Anda harus mengaktifkan Resolusi DNS dan opsi nama host DNS di VPC untuk alamat titik akhir ini agar dapat diselesaikan dengan benar. Untuk informasi selengkapnya, lihat Menggunakan DNS dengan VPC Anda dan Menggunakan CloudWatch dan CloudWatch Synthetics dengan titik akhir VPC antarmuka.
- Canary harus berjalan di subnet privat di dalam VPC. Untuk memeriksa hal ini, buka halaman Subnet di konsol VPC. Periksa subnet yang Anda pilih ketika mengonfigurasikan canary. Jika subnet tersebut memiliki jalur menuju gateway internet (igw-), mereka bukan subnet privat.

Untuk membantu Anda memecahkan masalah ini, silakan lihat log untuk canary.

Untuk melihat peristiwa log dari canary

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Grup log.
3. Pilih nama grup log canary. Nama grup log dimulai dengan `/aws/lambda/cwsyn-`*canary-name*.

Kode sampel untuk skrip canary

Bagian ini berisi contoh kode yang menggambarkan beberapa kemungkinan fungsi untuk skrip kenari CloudWatch Synthetics.

Sampel untuk Node.js dan Puppeteer

Mengatur cookie

Situs web mengandalkan cookie untuk menyediakan fungsionalitas kustom atau melacak para pengguna. Dengan menyetel cookie dalam skrip CloudWatch Synthetics, Anda dapat meniru perilaku kustom ini dan memvalidasinya.

Misalnya, situs web dapat menampilkan tautan Login untuk pengguna yang mengunjungi kembali, bukan tautan Daftar.

```
var synthetics = require('Synthetics');
const log = require('SyntheticsLogger');

const pageLoadBlueprint = async function () {

    let url = "http://smile.amazon.com/";

    let page = await synthetics.getPage();

    // Set cookies.  I found that name, value, and either url or domain are required
 fields.
    const cookies = [{
      'name': 'cookie1',
      'value': 'val1',
      'url': url
    },{
      'name': 'cookie2',
      'value': 'val2',
```

```
      'url': url
    },{
      'name': 'cookie3',
      'value': 'val3',
      'url': url
    }];

    await page.setCookie(...cookies);

    // Navigate to the url
    await synthetics.executeStep('pageLoaded_home', async function (timeoutInMillis =
 30000) {

        var response = await page.goto(url, {waitUntil: ['load', 'networkidle0'],
 timeout: timeoutInMillis});

        // Log cookies for this page and this url
        const cookiesSet = await page.cookies(url);
        log.info("Cookies for url: " + url + " are set to: " +
 JSON.stringify(cookiesSet));
    });

};

exports.handler = async () => {
    return await pageLoadBlueprint();
};
```

Emulasi perangkat

Anda dapat menulis skrip yang mengemulasi berbagai perangkat sehingga Anda dapat memperkirakan cara halaman terlihat dan berfungsi pada perangkat tersebut.

Sampel berikut mengemulasi perangkat iPhone 6. Untuk informasi selengkapnya tentang emulasi, silakan lihat page.emulate(options) dalam dokumentasi Puppeteer.

```
var synthetics = require('Synthetics');
const log = require('SyntheticsLogger');
const puppeteer = require('puppeteer-core');

const pageLoadBlueprint = async function () {

    const iPhone = puppeteer.devices['iPhone 6'];
```

```
    // INSERT URL here
    const URL = "https://amazon.com";

    let page = await synthetics.getPage();
    await page.emulate(iPhone);

    //You can customize the wait condition here. For instance,
    //using 'networkidle2' may be less restrictive.
    const response = await page.goto(URL, {waitUntil: 'domcontentloaded', timeout:
 30000});
    if (!response) {
        throw "Failed to load page!";
    }

    await page.waitFor(15000);

    await synthetics.takeScreenshot('loaded', 'loaded');

    //If the response status code is not a 2xx success code
    if (response.status() < 200 || response.status() > 299) {
        throw "Failed to load page!";
    }
};

exports.handler = async () => {
    return await pageLoadBlueprint();
};
```

Canary API multi-langkah

Kode sampel ini menunjukkan API canary dengan dua langkah HTTP: menguji API yang sama untuk kasus uji positif dan negatif. Konfigurasi langkah diteruskan untuk mengaktifkan pelaporan header permintaan/respons. Selain itu, ini menyembunyikan header Otorisasi dan X-Amz-Security-Token, karena memuat kredensial pengguna.

Ketika skrip ini digunakan sebagai canary, Anda dapat melihat detail tentang setiap langkah dan permintaan HTTP terkait seperti langkah lulus/gagal, durasi, dan metrik performa seperti waktu pencarian DNS dan waktu byte pertama. Anda dapat melihat jumlah 2xx, 4xx dan 5xx untuk berjalannya canary Anda.

```
var synthetics = require('Synthetics');
```

```
const log = require('SyntheticsLogger');

const apiCanaryBlueprint = async function () {

    // Handle validation for positive scenario
    const validatePositiveCase = async function(res) {
        return new Promise((resolve, reject) => {
            if (res.statusCode < 200 || res.statusCode > 299) {
                throw res.statusCode + ' ' + res.statusMessage;
            }

            let responseBody = '';
            res.on('data', (d) => {
                responseBody += d;
            });

            res.on('end', () => {
                // Add validation on 'responseBody' here if required. For ex, your
 status code is 200 but data might be empty
                resolve();
            });
        });
    };

    // Handle validation for negative scenario
    const validateNegativeCase = async function(res) {
        return new Promise((resolve, reject) => {
            if (res.statusCode < 400) {
                throw res.statusCode + ' ' + res.statusMessage;
            }

            resolve();
        });
    };

    let requestOptionsStep1 = {
        'hostname': 'myproductsEndpoint.com',
        'method': 'GET',
        'path': '/test/product/validProductName',
        'port': 443,
        'protocol': 'https:'
    };

    let headers = {};
```

```
    headers['User-Agent'] = [synthetics.getCanaryUserAgentString(), headers['User-
Agent']].join(' ');

    requestOptionsStep1['headers'] = headers;

    // By default headers, post data and response body are not included in the report
 for security reasons.
    // Change the configuration at global level or add as step configuration for
 individual steps
    let stepConfig = {
        includeRequestHeaders: true,
        includeResponseHeaders: true,
        restrictedHeaders: ['X-Amz-Security-Token', 'Authorization'], // Restricted
 header values do not appear in report generated.
        includeRequestBody: true,
        includeResponseBody: true
    };


    await synthetics.executeHttpStep('Verify GET products API with valid name',
 requestOptionsStep1, validatePositiveCase, stepConfig);

    let requestOptionsStep2 = {
        'hostname': 'myproductsEndpoint.com',
        'method': 'GET',
        'path': '/test/canary/InvalidName(',
        'port': 443,
        'protocol': 'https:'
    };

    headers = {};
    headers['User-Agent'] = [synthetics.getCanaryUserAgentString(), headers['User-
Agent']].join(' ');

    requestOptionsStep2['headers'] = headers;

    // By default headers, post data and response body are not included in the report
 for security reasons.
    // Change the configuration at global level or add as step configuration for
 individual steps
    stepConfig = {
        includeRequestHeaders: true,
        includeResponseHeaders: true,
```

```
        restrictedHeaders: ['X-Amz-Security-Token', 'Authorization'], // Restricted
 header values do not appear in report generated.
        includeRequestBody: true,
        includeResponseBody: true
    };

    await synthetics.executeHttpStep('Verify GET products API with invalid name',
 requestOptionsStep2, validateNegativeCase, stepConfig);

};

exports.handler = async () => {
    return await apiCanaryBlueprint();
};
```

Sampel untuk Python dan Selenium

Sampel kode Selenium berikut adalah sebuah canary yang gagal dengan sebuah pesan kesalahan kustom saat sebuah elemen target tidak dimuat.

```
from aws_synthetics.selenium import synthetics_webdriver as webdriver
from aws_synthetics.common import synthetics_logger as logger
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By

def custom_selenium_script():
    # create a browser instance
    browser = webdriver.Chrome()
    browser.get('https://www.example.com/')
    logger.info('navigated to home page')
    # set cookie
    browser.add_cookie({'name': 'foo', 'value': 'bar'})
    browser.get('https://www.example.com/')
    # save screenshot
    browser.save_screenshot('signed.png')
    # expected status of an element
    button_condition = EC.element_to_be_clickable((By.CSS_SELECTOR, '.submit-button'))
    # add custom error message on failure
    WebDriverWait(browser, 5).until(button_condition, message='Submit button failed to
 load').click()
    logger.info('Submit button loaded successfully')
    # browser will be quit automatically at the end of canary run,
```

```
    # quit action is not necessary in the canary script
    browser.quit()

# entry point for the canary
def handler(event, context):
    return custom_selenium_script()
```

Penelusuran Canary dan X-Ray

Anda dapat memilih untuk mengaktifkan AWS X-Ray penelusuran aktif pada kenari yang menggunakan runtime `syn-nodejs-2.0` atau yang lebih baru. Dengan penelusuran diaktifkan, jejak dikirim untuk semua panggilan yang dilakukan oleh kenari yang menggunakan browser, AWS SDK, atau modul HTTP atau HTTPS. Canary dengan pelacakan diaktifkan muncul di X-Ray Trace Map, dan di dalam Sinyal Aplikasi setelah Anda mengaktifkannya untuk aplikasi Anda.

> **Note**
>
> Mengaktifkan pelacakan X-Ray pada canary belum didukung di Asia Pasifik (Jakarta).

Ketika canary muncul di X-Ray Trace Map, itu muncul sebagai jenis simpul klien yang baru. Anda dapat mengarahkan kursor ke simpul canary untuk melihat data tentang latensi, permintaan, dan kesalahan. Anda juga dapat memilih simpul canary untuk melihat lebih banyak data di bagian bawah halaman. Dari area halaman ini, Anda dapat memilih Lihat di Synthetics untuk melompat ke konsol CloudWatch Synthetics untuk detail selengkapnya tentang kenari, atau pilih Lihat Jejak untuk melihat detail selengkapnya tentang jejak dari jalan kenari ini.

Canary dengan penelusuran aktif juga memiliki tab Pelacakan di halaman detailnya, dengan detail tentang jejak dan segmen dari operasi canary.

Mengaktifkan pelacakan meningkatkan runtime canary sebesar 2,5% hingga 7%.

Canary dengan penelusuran yang diaktifkan harus menggunakan peran dengan izin berikut. Jika Anda menggunakan konsol untuk membuat peran ketika membuat canary, maka akan diberikan izin ini.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "Sid230934",
            "Effect": "Allow",
            "Action": [
                "xray:PutTraceSegments"
            ],
            "Resource": "*"
        }
    ]
}
```

Penelusuran yang dihasilkan akan dikenakan biaya canary. Untuk informasi selengkapnya tentang penetapan harga X-Ray, silakan lihat Penetapan harga AWS X-Ray.

Menjalankan canary di VPC

Anda dapat menjalankan canary pada titik akhir di VPC dan titik akhir internal publik. Untuk menjalankan canary di VPC, Anda harus memiliki Resolusi DNS dan opsi nama host DNS yang diaktifkan pada VPC. Untuk informasi selengkapnya, silakan lihat Menggunakan DNS dengan VPC Anda.

Saat Anda menjalankan kenari di titik akhir VPC, Anda harus menyediakan cara untuk mengirim metriknya ke dan artefaknya CloudWatch ke Amazon S3. Jika VPC sudah diaktifkan untuk akses internet, tidak ada lagi yang bisa Anda lakukan. Canary mengeksekusi di VPC Anda, tetapi dapat mengakses internet untuk mengunggah metrik dan artefaknya.

Jika VPC belum diaktifkan untuk akses internet, Anda memiliki dua opsi:

- Aktifkan untuk akses internet. Untuk informasi selengkapnya, silakan lihat bagian Memberikan akses internet ke canary Anda di VPC berikut ini.
- Jika Anda ingin menjaga VPC Anda tetap pribadi, Anda dapat mengonfigurasi kenari untuk mengirim datanya ke dan Amazon CloudWatch S3 melalui titik akhir VPC pribadi. Jika Anda belum melakukannya, Anda harus membuat titik akhir VPC untuk CloudWatch (com.amazonaws. *region* .monitoring) dan titik akhir gateway untuk Amazon S3. Untuk informasi selengkapnya, silakan lihat Menggunakan CloudWatch dan CloudWatch Synthetics dengan titik akhir VPC antarmuka dan Amazon titik akhir VPC untuk Amazon S3.

Memberikan akses internet ke canary Anda di VPC

Ikuti langkah-langkah ini untuk memberikan akses internet ke kenari VPC Anda, atau untuk menetapkan kenari Anda alamat IP statis

Untuk memberikan akses internet ke canary di VPC

1. Buat gateway NAT di subnet publik di VPC. Untuk petunjuk, silakan lihat Membuat gateway NAT.
2. Tambahkan rute baru ke tabel rute di subnet privat tempat canary diluncurkan. Tentukan hal berikut:

   - Untuk Tujuan, masukkan **`0.0.0.0/0`**
   - Untuk Target, pilih Gateway NAT, lalu pilih ID gateway NAT yang Anda buat.
   - Pilih Simpan rute.

   Untuk informasi selengkapnya tentang menambahkan rute ke tabel rute, silakan lihat Menambahkan dan menghapus rute dari tabel rute.

> **Note**
>
> Pastikan rute ke gateway NAT Anda dalam status aktif. Jika gateway NAT dihapus dan Anda belum memperbarui rute, mereka berada dalam status lubang hitam. Untuk informasi selengkapnya, silakan lihat Bekerja dengan gateway NAT.

Mengenkripsi artefak canary

CloudWatch Synthetics menyimpan artefak kenari seperti tangkapan layar, file HAR, dan laporan di bucket Amazon S3 Anda. Secara default, artefak ini dienkripsi saat istirahat menggunakan kunci terkelola AWS . Untuk informasi selengkapnya, lihat Kunci dan AWS kunci pelanggan.

Anda dapat memilih untuk menggunakan opsi enkripsi yang berbeda. CloudWatch Synthetics mendukung hal-hal berikut:

- SSE–S3 – Enkripsi di sisi server (SSE) dengan kunci yang dikelola Amazon S3.
- SSE–KMS – Enkripsi di sisi server (SSE) dengan Kunci yang dikelola pelanggan AWS KMS .

Jika Anda ingin menggunakan opsi enkripsi default dengan kunci AWS terkelola, Anda tidak memerlukan izin tambahan.

Untuk menggunakan enkripsi SSE-S3, Anda menentukan SSE_S3 sebagai mode enkripsi saat Anda membuat atau memperbarui canary Anda. Anda tidak memerlukan izin tambahan untuk

menggunakan mode enkripsi ini. Untuk informasi selengkapnya, silakan lihat Melindungi Data Menggunakan Enkripsi di Sisi Server dengan Kunci Enkripsi yang terkelola oleh Amazon S3 (SSE-S3).

Untuk menggunakan kunci terkelola AWS KMS pelanggan, Anda menentukan SSE-KMS sebagai mode enkripsi saat membuat atau memperbarui kenari, dan Anda juga memberikan Nama Sumber Daya Amazon (ARN) kunci Anda. Anda juga dapat menggunakan kunci KMS lintas akun.

Untuk menggunakan kunci yang dikelola pelanggan, Anda memerlukan pengaturan berikut ini:

- Peran IAM untuk canary Anda harus memiliki izin untuk mengenkripsi artefak Anda menggunakan kunci Anda. Jika Anda menggunakan pemantauan visual, Anda juga harus memberikan izin untuk mendekripsi artefak.

```
{
    "Version": "2012-10-17",
    "Statement": {"Effect": "Allow",
        "Action": [
            "kms:GenerateDataKey",
            "kms:Decrypt"
        ],
        "Resource": "Your KMS key ARN"
    }
}
```

- Alih-alih menambahkan izin ke peran IAM, Anda dapat menambahkan peran IAM ke kebijakan kunci Anda. Jika Anda menggunakan peran yang sama untuk beberapa canary, Anda harus mempertimbangkan pendekatan ini.

```
{
    "Sid": "Enable IAM User Permissions",
    "Effect": "Allow",
    "Principal": {
        "AWS": "Your synthetics IAM role ARN"
    },
    "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
    ],
    "Resource": "*"
}
```

- Jika Anda menggunakan kunci KMS lintas akun, silakan lihat Mengizinkan pengguna di akun lain untuk menggunakan kunci KMS.

Melihat artefak canary terenkripsi saat menggunakan kunci yang dikelola pelanggan

Untuk melihat artefak kenari, perbarui kunci terkelola pelanggan Anda untuk memberikan AWS KMS izin dekripsi kepada pengguna yang melihat artefak. Atau, tambahkan izin dekripsi ke pengguna atau peran IAM yang melihat artefak tersebut.

AWS KMS Kebijakan default memungkinkan kebijakan IAM di akun untuk mengizinkan akses ke kunci KMS. Jika Anda menggunakan kunci KMS lintas akun, lihat Mengapa pengguna lintas akun mendapatkan kesalahan Akses Ditolak saat mereka mencoba mengakses objek Amazon S3 yang dienkripsi oleh kunci khusus? AWS KMS .

Untuk informasi selengkapnya tentang pemecahan masalah akses yang ditolak karena kunci KMS, silakan lihat Memecahkan masalah akses kunci.

Memperbarui lokasi artefak dan enkripsi saat menggunakan pemantauan visual

Untuk melakukan pemantauan visual, CloudWatch Synthetics membandingkan tangkapan layar Anda dengan tangkapan layar dasar yang diperoleh dalam proses yang dipilih sebagai baseline. Jika Anda memperbarui lokasi artefak atau opsi enkripsi, Anda harus melakukan salah satu hal berikut:

- Pastikan bahwa peran IAM Anda memiliki izin yang cukup untuk lokasi Amazon S3 sebelumnya dan lokasi Amazon S3 baru untuk artefak. Juga pastikan bahwa ia memiliki izin untuk metode enkripsi sebelumnya dan metode enkripsi baru dan kunci KMS.
- Buat baseline baru dengan memilih run canary berikutnya sebagai baseline baru. Jika Anda menggunakan opsi ini, Anda hanya perlu memastikan bahwa peran IAM Anda memiliki izin yang cukup untuk lokasi artefak dan opsi enkripsi baru.

Kami merekomendasikan opsi kedua untuk memilih run berikutnya sebagai baseline baru. Ini menghindari dependensi pada lokasi artefak atau opsi enkripsi yang tidak Anda gunakan lagi untuk canary.

Misalnya, canary Anda menggunakan lokasi artefak A dan kunci KMS K untuk mengunggah artefak. Jika Anda memperbarui canary ke lokasi artefak B dan kunci KMS L, Anda dapat memastikan bahwa peran IAM Anda memiliki izin untuk kedua lokasi artefak (A dan B) dan kedua kunci KMS (K dan L). Atau, Anda dapat memilih proses berikutnya sebagai baseline baru dan memastikan bahwa peran IAM canary Anda memiliki izin untuk artefak lokasi B dan kunci KMS L.

Melihat statistik dan detail canary

Anda dapat melihat detail tentang canary dan melihat statistik tentang operasinya.

Untuk dapat melihat semua detail tentang hasil operasi canary Anda, Anda harus log on ke akun yang memiliki izin yang memadai. Untuk informasi selengkapnya, lihat Peran dan izin yang diperlukan untuk CloudWatch canary.

Untuk melihat statistik dan detail canary

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Application Signals, Synthetics Canaries.

   Dalam detail tentang canary yang telah Anda buat:

   - Status secara visual menunjukkan seberapa banyak canary Anda yang telah lulus dari operasi terbaru.
   - Grup menampilkan grup yang telah Anda buat, dan menampilkan berapa banyak dari mereka yang memiliki canary yang gagal atau mengkhawatirkan.
   - Penampil paling lambat menampilkan grup dan Wilayah dengan canary berperforma paling lambat. Ini dihitung dengan menjumlahkan durasi rata-rata semua canary (di seluruh rentang waktu yang dipilih) dalam kelompok atau Wilayah dan membaginya dengan jumlah canary dalam grup atau Wilayah. Jika Anda memilih metrik untuk grup Paling Lambat, tabel difilter untuk menampilkan hanya grup paling lambat dan canary mereka. Tabel diurutkan berdasarkan Durasi Rata-rata.
   - Dekat bagian bawah halaman adalah tabel yang menampilkan semua canary. Satu kolom menampilkan alarm yang dibuat untuk setiap canary. Hanya alarm yang sesuai dengan standar penamaan untuk alarm canary yang ditampilkan. Standar ini adalah `Synthetics-Alarm-`*`canaryName`*`-`*`index`* . Alarm Canary yang Anda buat di bagian Synthetics CloudWatch konsol secara otomatis menggunakan konvensi penamaan ini. Jika Anda membuat alarm kenari di bagian Alarm CloudWatch konsol atau dengan menggunakan AWS CloudFormation, dan Anda tidak menggunakan konvensi penamaan ini, alarm berfungsi tetapi alarm tersebut tidak muncul dalam daftar ini.
3. Untuk melihat detail lebih lanjut tentang satu canary, pilih nama canary di tabel Canary.

   Dalam detail tentang canary tersebut:

   - Tab Ketersediaan menampilkan informasi tentang operasi terbaru dari canary ini.

Pada Jalankan canary, Anda dapat memilih salah satu baris untuk melihat detail operasi tersebut.

Di bawah grafik, Anda dapat memilih Langkah, Tangkapan Layar, Log, atau file HAR untuk melihat jenis detail ini. Jika canary mengaktifkan penelusuran aktif, Anda juga dapat memilih Penelusuran untuk melihat informasi penelusuran dari operasi canary.

Log untuk proses kenari disimpan dalam bucket S3 dan di Log. CloudWatch

Tangkapan layar menunjukkan cara para pelanggan Anda melihat halaman web. Anda dapat menggunakan file HAR (file Arsip HTTP) untuk melihat data performa terperinci tentang halaman web. Anda dapat menganalisis daftar permintaan web dan menangkap masalah performa seperti waktu untuk memuat sebuah item. File log menunjukkan catatan interaksi antara operasi canary dan halaman web dan dapat digunakan untuk mengidentifikasi detail kesalahan.

Jika canary menggunakan runtime `syn-nodejs-2.0-beta` atau yang lebih baru, Anda dapat menyortir file HAR berdasarkan kode status, ukuran permintaan, atau durasi.

Tab Langkah menampilkan daftar langkah-langkah canary, status setiap langkah, alasan kegagalan, URL setelah eksekusi langkah, tangkapan layar, dan durasi eksekusi langkah. Untuk canary API dengan langkah-langkah HTTP, Anda dapat melihat langkah-langkah dan permintaan HTTP yang sesuai jika Anda menggunakan runtime `syn-nodejs-2.2` atau yang lebih baru.

Pilih tab Permintaan HTTP untuk melihat log dari setiap permintaan HTTP yang dibuat oleh canary. Anda dapat melihat header permintaan/tanggapan, isi respons, kode status, kesalahan dan pengaturan waktu performa (total durasi, waktu koneksi TCP, waktu handshake TLS, waktu byte pertama, dan waktu transfer konten). Semua permintaan HTTP yang menggunakan modul HTTP/HTTPS di bawah hood ditangkap di sini.

Secara default di canary API, header permintaan, header respon, isi permintaan, dan isi respons tidak termasuk dalam laporan untuk alasan keamanan. Jika Anda memilih untuk memasukkannya, data tersebut disimpan hanya dalam bucket S3 Anda. Untuk informasi tentang cara menyertakan data ini dalam laporan, silakan lihat executeHttpStep(StepName, requestOptions, [callback], [stepConfig]).

Jenis konten isi respons teks, HTML, dan JSON didukung. Jenis konten seperti teks/HTML, teks/polos, aplikasi/JSON dan aplikasi/ -1.0 didukung. x-amz-json Respons terkompresi tidak didukung.

- Tab Monitoring menampilkan grafik CloudWatch metrik yang diterbitkan oleh kenari ini. Untuk informasi selengkapnya tentang metrik ini, lihat CloudWatch metrik yang diterbitkan oleh kenari.

  Di bawah CloudWatch grafik yang diterbitkan oleh kenari adalah grafik metrik Lambda yang terkait dengan kode Lambda kenari.
- Tab Konfigurasi menampilkan konfigurasi dan informasi jadwal tentang canary.
- Tab Grup menampilkan grup yang dikaitkan dengan canary ini, jika ada.
- Tab Tanda menampilkan tanda yang terkait dengan canary.

CloudWatch metrik yang diterbitkan oleh kenari

Canaries mempublikasikan metrik berikut ke CloudWatch dalam namespace. `CloudWatchSynthetics` Untuk informasi selengkapnya tentang melihat CloudWatch metrik, lihatLihat metrik yang tersedia.

Metrik	Deskripsi
`SuccessPercent`	Persentase operasi canary yang berhasil dan tidak ditemukan kegagalan . Dimensi yang valid: CanaryName Statistik yang Valid: Average Satuan: Persen
`Duration`	Durasi dalam milidetik run canary. Dimensi yang valid: CanaryName Statistik yang Valid: Average Satuan: Milidetik

Metrik	Deskripsi
`Errors`	Berapa kali kenari gagal menjalankan skrip lengkapnya. Dimensi yang valid: CanaryName Statistik Valid: Sum
`2xx`	Jumlah permintaan jaringan yang dilakukan oleh canary yang mengembalikan respons OK, dengan kode tanggapan antara 200 dan 299. Metrik ini dilaporkan untuk canary UI yang menggunakan versi runtime `syn-nodejs-2.0` atau yang lebih baru, dan dilaporkan untuk canary API yang menggunakan versi runtime `syn-nodejs-2.2` atau yang lebih baru. Dimensi yang valid: CanaryName Statistik Valid: Sum Satuan: Hitungan
`4xx`	Jumlah permintaan jaringan yang dilakukan oleh canary yang mengembalikan respons Kesalahan, dengan kode tanggapan antara 400 dan 499. Metrik ini dilaporkan untuk canary UI yang menggunakan versi runtime `syn-nodejs-2.0` atau yang lebih baru, dan dilaporkan untuk canary API yang menggunakan versi runtime `syn-nodejs-2.2` atau yang lebih baru. Dimensi yang valid: CanaryName Statistik Valid: Sum Satuan: Hitungan

Metrik	Deskripsi
`5xx`	Jumlah permintaan jaringan yang dilakukan oleh canary yang mengembalikan respons Kesalahan, dengan kode respons antara 500 dan 599. Metrik ini dilaporkan untuk canary UI yang menggunakan versi runtime `syn-nodejs-2.0` atau yang lebih baru, dan dilaporkan untuk canary API yang menggunakan versi runtime `syn-nodejs-2.2` atau yang lebih baru. Dimensi yang valid: CanaryName Statistik Valid: Sum Satuan: Hitungan
`Failed`	Jumlah operasi canary yang gagal untuk dieksekusi. Kegagalan ini berkaitan dengan canary itu sendiri. Dimensi yang valid: CanaryName Statistik Valid: Sum Satuan: Hitungan
`Failed requests`	Jumlah permintaan HTTP yang dijalankan oleh canary pada situs web target yang gagal tanpa respons. Dimensi yang valid: CanaryName Statistik Valid: Sum Satuan: Hitungan

Metrik	Deskripsi
`VisualMonitoringSuccessPercent`	Persentase perbandingan visual yang berhasil mencocokkan tangkapan layar dasar selama berjalannya canary. Dimensi yang valid: CanaryName Statistik yang Valid: Average Satuan: Persen
`VisualMonitoringTotalComparisons`	Jumlah total perbandingan visual yang terjadi selama berjalannya canary. Dimensi yang valid: CanaryName Satuan: Hitungan

> **Note**
>
> Canary yang menggunakan metode baik `executeStep()` atau `executeHttpStep()` dari pustaka Synthetics juga menerbitka metrik `SuccessPercent` dan `Duration` dengan dimensi `CanaryName` dan `StepName` untuk setiap langkah.

Mengedit atau menghapus canary

Anda dapat menyunting atau menghapus canary yang ada.

Sunting canary

Ketika Anda menyunting canary, bahkan jika Anda tidak mengubah jadwalnya, jadwal akan diatur ulang sesuai dengan ketika Anda menyunting canary. Sebagai contoh, jika Anda memiliki canary yang berjalan setiap jam, dan Anda menyuntingnya, canary akan berjalan segera setelah penyuntingan selesai dan kemudian setiap jam setelah itu.

Untuk menyunting atau memperbarui canary

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.

2. Di panel navigasi, pilih Application Signals, Synthetics Canaries.
3. Pilih tombol di samping nama canary, dan pilih Tindakan, Sunting.
4. (Opsional) Jika canary ini melakukan pemantauan visual tangkapan layar dan Anda ingin mengatur proses canary berikutnya sebagai baseline, pilih Tetapkan run berikutnya sebagai baseline baru.
5. (Opsional) Jika canary ini melakukan pemantauan visual tangkapan layar dan Anda ingin menghapus tangkapan layar dari pemantauan visual atau Anda ingin menunjuk bagian tangkapan layar yang akan diabaikan selama perbandingan visual, di bawah Pemantauan Visual pilih Sunting Baseline.

   Tangkapan layar muncul, dan Anda dapat melakukan salah satu dari yang berikut ini:

   - Untuk menghapus tangkapan layar agar tidak digunakan untuk pemantauan visual, pilih Hapus tangkapan layar dari baseline tes visual.
   - Untuk menunjuk bagian tangkapan layar yang akan diabaikan selama perbandingan visual, klik dan seret untuk menggambar area layar untuk diabaikan. Setelah Anda melakukan ini untuk semua area yang ingin Anda abaikan selama perbandingan, pilih Simpan.
6. Buat perubahan lain pada canary yang Anda inginkan, dan pilih Simpan.

Hapus canary

Ketika Anda menghapus canary, Anda dapat memilih apakah akan menghapus sumber daya lain yang digunakan dan dibuat oleh canary tersebut. Saat Anda menghapus canary, Anda juga harus menghapus hal-hal berikut ini:

- Fungsi dan lapisan Lambda yang digunakan oleh canary ini. Awalan mereka adalah `cwsyn-`*`MyCanaryName`*.
- CloudWatch alarm dibuat untuk kenari ini. Alarm ini memiliki nama yang dimulai dengan `Synthetics-Alarm-`*`MyCanaryName`*. Untuk informasi selengkapnya tentang penghapusan alarm, silakan lihat Mengedit atau menghapus CloudWatch alarm.
- Objek dan bucket Amazon S3, seperti lokasi hasil canary dan lokasi artefak.
- Peran IAM yang dibuat untuk canary tersebut. Peran ini memiliki nama `role/service-role/CloudWatchSyntheticsRole-`*`MyCanaryName`*.
- Grup log di CloudWatch Log yang dibuat untuk kenari. Grup log ini memiliki nama berikut: `/aws/lambda/cwsyn-`*`MyCanaryName-randomId`*.

Sebelum menghapus canary, Anda mungkin ingin melihat detail canary dan mencatat informasi ini. Dengan demikian, Anda dapat menghapus sumber daya yang benar setelah Anda menghapus canary tersebut.

Untuk menghapus canary

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Application Signals, Synthetics Canaries.
3. Jika canary saat ini berada di `RUNNING` negara bagian, Anda harus menghentikannya. Hanya canary di`STOPPED`, `READY(NOT_STARTED)`, atau `ERROR` negara bagian yang dapat dihapus.

   Untuk menghentikan canary, pilih tombol di samping nama canary, dan pilih Tindakan, Berhenti.
4. Pilih tombol di samping nama canary, dan pilih Tindakan, Hapus.
5. Pilih apakah akan menghapus sumber daya lain yang dibuat untuk dan digunakan oleh canary tersebut. Ini termasuk fungsi Lambda dan lapisan, dan peran IAM canary dan kebijakan IAM.

   Untuk menghapus peran IAM dan kebijakan IAM canary, Anda harus memiliki izin yang memadai. Untuk informasi selengkapnya, lihat AWS kebijakan terkelola (standar) untuk CloudWatch Synthetics.
6. Masukkan **`Delete`** ke dalam kotak dan pilih Hapus.
7. Hapus sumber daya lain yang digunakan dan dibuat untuk canary, seperti yang tercantum sebelumnya di bagian ini.

Mulai, hentikan, hapus, atau perbarui runtime untuk banyak canary

Anda dapat menghentikan, memulai, menghapus, atau memperbarui runtime sebanyak lima canary dengan satu tindakan. Jika Anda memperbarui runtime canary, itu diperbarui ke runtime terbaru yang tersedia untuk bahasa dan kerangka kerja yang digunakan oleh canary tersebut.

Jika Anda memilih banyak canary dan hanya beberapa dari mereka yang berada dalam keadaan yang valid untuk tindakan yang Anda pilih, maka tindakan itu hanya dilakukan pada canary di mana tindakan itu valid. Sebagai contoh, jika Anda memilih beberapa canary yang sedang berjalan dan beberapa yang tidak, dan Anda memilih untuk memulai canary, maka canary yang belum berjalan akan mulai, dan canary yang sudah berjalan tidak terpengaruh.

Jika tidak ada canary yang Anda pilih valid untuk suatu tindakan, tindakan itu tidak akan tersedia di menu.

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Application Signals, Synthetics Canaries.
3. Centang kotak di samping canary yang ingin Anda hentikan, mulai, atau hapus.
4. Pilih Tindakan dan kemudian pilih Mulai, Berhenti, Hapus, atau Perbarui Runtime.

Memantau peristiwa kenari dengan Amazon EventBridge

Aturan EventBridge acara Amazon dapat memberi tahu Anda saat kenari mengubah status atau menyelesaikan proses. EventBridgememberikan near-real-time aliran peristiwa sistem yang menggambarkan perubahan AWS sumber daya. CloudWatch Synthetics mengirimkan acara ini ke EventBridge atas dasar upaya terbaik. Penyampaian upaya terbaik berarti bahwa CloudWatch Synthetics mencoba mengirim semua acara ke EventBridge, tetapi dalam beberapa kasus yang jarang terjadi suatu peristiwa mungkin tidak terkirim. EventBridge memproses semua acara yang diterima setidaknya sekali. Selain itu, pendengar peristiwa Anda mungkin tidak menerima peristiwa dalam urutan peristiwa yang terjadi.

> **Note**
>
> Amazon EventBridge adalah layanan bus acara yang dapat Anda gunakan untuk menghubungkan aplikasi Anda dengan data dari berbagai sumber. Untuk informasi selengkapnya, lihat Apa itu Amazon EventBridge? di Panduan EventBridge Pengguna Amazon.

CloudWatch Synthetics memancarkan peristiwa ketika kenari mengubah status atau menyelesaikan proses. Anda dapat membuat EventBridge aturan yang menyertakan pola acara agar sesuai dengan semua jenis acara yang dikirim dari CloudWatch Synthetics, atau yang hanya cocok dengan jenis peristiwa tertentu. Ketika kenari memicu aturan, EventBridge memanggil tindakan target yang ditentukan dalam aturan. Hal ini memungkinkan Anda untuk mengirim notifikasi, menangkap informasi peristiwa, dan mengambil tindakan korektif, dalam menanggapi perubahan status canary atau selesainya sebuah run canary. Sebagai contoh, Anda dapat membuat aturan untuk kasus penggunaan berikut:

- Investigasi ketika operasi canary gagal
- Investigasi ketika canary telah masuk ke status `ERROR`
- Menelusuri siklus hidup canary

- Memantau keberhasilan atau kegagalan menjalankan canary sebagai bagian dari alur kerja

Contoh peristiwa dari CloudWatch Synthetics

Bagian ini mencantumkan contoh peristiwa dari CloudWatch Synthetics. Untuk informasi selengkapnya tentang format acara, lihat Peristiwa dan Pola Peristiwa di EventBridge.

Perubahan status canary

Dalam jenis peristiwa ini, nilai-nilai dari `current-state` dan `previous-state` dapat berupa berikut:

`CREATING` | `READY` | `STARTING` | `RUNNING` | `UPDATING` | `STOPPING` | `STOPPED` | `ERROR`

```
{
                "version": "0",
                "id": "8a99ca10-1e97-2302-2d64-316c5dedfd61",
                "detail-type": "Synthetics Canary Status Change",
                "source": "aws.synthetics",
                "account": "123456789012",
                "time": "2021-02-09T22:19:43Z",
                "region": "us-east-1",
                "resources": [],
                "detail": {
                                "account-id": "123456789012",
                                "canary-id": "EXAMPLE-dc5a-4f5f-96d1-989b75a94226",
                                "canary-name": "events-bb-1",
                                "current-state": "STOPPED",
                                "previous-state": "UPDATING",
                                "source-location": "NULL",
                                "updated-on": 1612909161.767,
                                "changed-config": {
                                                "executionArn": {
                                                                "previous-value":
 "arn:aws:lambda:us-east-1:123456789012:function:cwsyn-events-bb-1-af3e3a05-
dc5a-4f5f-96d1-989EXAMPLE:1",
                                                                "current-value":
 "arn:aws:lambda:us-east-1:123456789012:function:cwsyn-events-bb-1-af3e3a05-
dc5a-4f5f-96d1-989EXAMPLE:2"
                                                },
                                                "vpcId": {
                                                                "current-value": "NULL"
```

```
                                                },
                                                "testCodeLayerVersionArn": {
                                                                "previous-
value": "arn:aws:lambda:us-east-1:123456789012:layer:cwsyn-events-bb-1-af3e3a05-
dc5a-4f5f-96d1-989EXAMPLE:1",
                                                                "current-value":
 "arn:aws:lambda:us-east-1:123456789012:layer:cwsyn-events-bb-1-af3e3a05-
dc5a-4f5f-96d1-989EXAMPLE:2"
                                                }
                                },
                                "message": "Canary status has changed"
                }
}
```

Eksekusi canary yang berhasil telah selesai

```
{
                "version": "0",
                "id": "989EXAMPLE-f4a5-57a7-1a8f-d9cc768a1375",
                "detail-type": "Synthetics Canary TestRun Successful",
                "source": "aws.synthetics",
                "account": "123456789012",
                "time": "2021-02-09T22:24:01Z",
                "region": "us-east-1",
                "resources": [],
                "detail": {
                                "account-id": "123456789012",
                                "canary-id": "989EXAMPLE-dc5a-4f5f-96d1-989b75a94226",
                                "canary-name": "events-bb-1",
                                "canary-run-id": "c6c39152-8f4a-471c-9810-989EXAMPLE",
                                "artifact-location": "cw-syn-results-123456789012-us-
east-1/canary/us-east-1/events-bb-1-ec3-28ddbe266797/2021/02/09/22/23-41-200",
                                "test-run-status": "PASSED",
                                "state-reason": "null",
                                "canary-run-timeline": {
                                                "started": 1612909421,
                                                "completed": 1612909441
                                },
                                "message": "Test run result is generated successfully"
                }
}
```

Eksekusi canary yang gagal telah selesai

```
{
                "version": "0",
                "id": "2644b18f-3e67-5ebf-cdfd-bf9f91392f41",
                "detail-type": "Synthetics Canary TestRun Failure",
                "source": "aws.synthetics",
                "account": "123456789012",
                "time": "2021-02-09T22:24:27Z",
                "region": "us-east-1",
                "resources": [],
                "detail": {
                                "account-id": "123456789012",
                                "canary-id": "af3e3a05-dc5a-4f5f-96d1-9989EXAMPLE",
                                "canary-name": "events-bb-1",
                                "canary-run-id": "0df3823e-7e33-4da1-8194-
b04e4d4a2bf6",
                                "artifact-location": "cw-syn-results-123456789012-us-
east-1/canary/us-east-1/events-bb-1-ec3-989EXAMPLE/2021/02/09/22/24-21-275",
                                "test-run-status": "FAILED",
                                "state-reason": "\"Error: net::ERR_NAME_NOT_RESOLVED
 \""
                                "canary-run-timeline": {
                                                "started": 1612909461,
                                                "completed": 1612909467
                                },
                                "message": "Test run result is generated successfully"
                }
}
```

Ada kemungkinan bahwa peristiwa dapat diduplikasi atau rusak. Untuk menentukan urutan peristiwa, gunakan properti `time` ini.

Prasyarat untuk membuat aturan EventBridge

Sebelum Anda membuat EventBridge aturan untuk CloudWatch Synthetics, Anda harus melakukan hal berikut:

- Biasakan diri Anda dengan acara, aturan, dan target di EventBridge.
- Buat dan konfigurasikan target yang dipanggil oleh EventBridge aturan Anda. Aturan dapat menginvokasi berbagai jenis target, termasuk:
  - Topik Amazon SNS
  - AWS Lambda fungsi

- Aliran Kinesis
- Antrean Amazon SQS

Untuk informasi selengkapnya, lihat Apa itu Amazon EventBridge? dan Memulai dengan Amazon EventBridge di Panduan EventBridge Pengguna Amazon.

Buat EventBridge aturan (CLI)

Langkah-langkah dalam contoh berikut membuat EventBridge aturan yang menerbitkan topik Amazon SNS saat kenari `my-canary-name` bernama `us-east-1` dalam menyelesaikan status run atau perubahan.

1. Buat aturan.

```
aws events put-rule \
  --name TestRule \
  --region us-east-1 \
  --event-pattern "{\"source\": [\"aws.synthetics\"], \"detail\": {\"canary-name\":
 [\"my-canary-name\"]}}"
```

   Properti apa pun yang Anda hilangkan dari pola akan diabaikan.

2. Tambahkan topik sebagai target aturan.

   - Ganti *topic-arn* dengan Amazon Resource Name (ARN) dari topik Amazon SNS Anda.

```
aws events put-targets \
  --rule TestRule \
  --targets "Id"="1","Arn"="topic-arn"
```

> **Note**
>
> Untuk mengizinkan Amazon EventBridge memanggil topik target Anda, Anda harus menambahkan kebijakan berbasis sumber daya ke topik Anda. Untuk informasi selengkapnya, lihat izin Amazon SNS di Panduan Pengguna Amazon EventBridge .

Untuk informasi selengkapnya, lihat Peristiwa dan pola acara EventBridge di Panduan EventBridge Pengguna Amazon.

Lakukan peluncuran dan eksperimen A/B dengan Evidently CloudWatch

Anda dapat menggunakan Amazon CloudWatch Terbukti untuk memvalidasi fitur baru dengan aman dengan menyajikannya ke persentase tertentu dari pengguna Anda saat Anda meluncurkan fitur tersebut. Anda dapat memantau performa fitur baru untuk membantu Anda memutuskan kapan harus menaikkan lalu lintas ke para pengguna Anda. Hal ini akan membantu Anda mengurangi risiko dan mengidentifikasi konsekuensi yang tidak Anda inginkan sebelum Anda meluncurkan fitur tersebut sepenuhnya.

Anda juga dapat melakukan percobaan-percobaan A/B untuk membuat keputusan desain fitur berdasarkan bukti dan data. Sebuah percobaan dapat menguji sebanyak lima variasi sekaligus. Evidently mengumpulkan data percobaan dan menganalisisnya dengan menggunakan metode statistik. Evidently juga memberikan rekomendasi-rekomendasi yang jelas tentang variasi mana yang memiliki performa yang lebih baik. Anda dapat menguji fitur yang dihadapi pengguna dan fitur-fitur backend.

Penentuan harga Evidently

Evidently mengenakan biaya pada akun Anda berdasarkan peristiwa-peristiwa Evidently dan unit analisis Evidently. Peristiwa-peristiwa Evidently mencakup peristiwa-peristiwa data seperti klik dan tampilan halaman, dan peristiwa-peristiwa penetapan yang menentukan variasi fitur yang akan disajikan kepada pengguna.

Unit analisis Evidently dihasilkan dari peristiwa-peristiwa Evidently, berdasarkan aturan yang telah Anda buat di Evidently. Unit analisis adalah jumlah kecocokan aturan pada peristiwa. Sebagai contoh, sebuah peristiwa klik pengguna bisa menghasilkan satu unit analisis Evidently, satu hitungan klik. Contoh lainnya adalah peristiwa checkout pengguna yang mungkin menghasilkan dua unit analisis Evidently, nilai checkout dan jumlah item yang ada di keranjang. Untuk informasi selengkapnya tentang harga, lihat CloudWatch Harga Amazon.

CloudWatch Terbukti saat ini tersedia di Wilayah berikut:

- AS Timur (Ohio)
- AS Timur (Virginia Utara)

- AS Barat (Oregon)
- Asia Pasifik (Singapura)
- Asia Pasifik (Sydney)
- Asia Pasifik (Tokyo)
- Eropa (Frankfurt)
- Eropa (Irlandia)
- Eropa (Stockholm)

Topik

Kebijakan IAM untuk menggunakan Evidently

Untuk mengelola sepenuhnya CloudWatch Jelas, Anda harus masuk sebagai pengguna IAM atau peran yang memiliki izin berikut:

- Kebijakan AmazonCloudWatchEvidentlyFullAccess
- Kebijakan ResourceGroupsandTagEditorReadOnlyAccess

Selain itu, untuk dapat membuat proyek yang menyimpan peristiwa evaluasi di Amazon S3 atau CloudWatch Log, Anda memerlukan izin berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketPolicy",
                "s3:PutBucketPolicy",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:DescribeResourcePolicies",
                "logs:PutResourcePolicy"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Izin tambahan untuk integrasi CloudWatch RUM

Selain itu, jika Anda bermaksud mengelola peluncuran atau eksperimen yang terbukti yang terintegrasi dengan Amazon CloudWatch RUM dan menggunakan metrik CloudWatch RUM untuk pemantauan, Anda memerlukan kebijakan RUM. AmazonCloudWatch FullAccess Untuk membuat peran IAM untuk memberikan izin klien web CloudWatch RUM untuk mengirim data ke CloudWatch RUM, Anda memerlukan izin berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "iam:CreateRole",
                "iam:CreatePolicy",
                "iam:AttachRolePolicy"
            ],
            "Resource": [
                "arn:aws:iam::*:role/service-role/CloudWatchRUMEvidentlyRole-*",
                "arn:aws:iam::*:policy/service-role/CloudWatchRUMEvidentlyPolicy-*"
            ]
        }
    ]
}
```

Izin untuk akses baca saja ke Evidently

Untuk pengguna lain yang perlu melihat data yang terbukti tetapi tidak perlu membuat sumber daya yang terbukti, Anda dapat memberikan kebijakan tersebut AmazonCloudWatchEvidentlyReadOnlyAccess.

Membuat proyek, fitur, peluncuran, dan percobaan

Untuk memulai dengan CloudWatch Evidently, baik untuk peluncuran fitur atau percobaan A/B, Anda pertama kali membuat proyek. Sebuah proyek adalah pembuatan grup logis dari sumber daya. Dalam proyek tersebut, Anda membuat fitur yang memiliki variasi yang ingin Anda uji atau luncurkan. Anda dapat membuat sebuah fitur sebelum Anda membuat peluncuran atau percobaan, atau secara bersamaan.

Topik

- Membuat sebuah proyek baru
- Menggunakan evaluasi sisi klien - didukung oleh AWS AppConfig
- Menambahkan sebuah fitur ke proyek
- Gunakan segmen-segmen untuk memfokuskan pada audiens Anda
- Membuat sebuah peluncuran
- Membuat sebuah percobaan

Membuat sebuah proyek baru

Gunakan langkah-langkah ini untuk menyiapkan proyek CloudWatch Evidently baru.

Untuk membuat proyek CloudWatch Evidently baru

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Sinyal Aplikasi, Terbukti.
3. Pilih Buat proyek.
4. Untuk nama Project, masukkan nama yang akan digunakan untuk mengidentifikasi proyek ini dalam konsol CloudWatch Evidently.

   Anda dapat menambahkan sebuah deskripsi proyek.
5. Untuk Penyimpanan peristiwa evaluasi, pilih apakah Anda ingin menyimpan peristiwa evaluasi yang Anda kumpulkan dengan Evidently. Bahkan jika Anda tidak menyimpan peristiwa ini, Evidently menggabungkannya untuk membuat metrik-metrik dan data percobaan lain yang dapat Anda lihat di dasbor Evidently. Untuk informasi selengkapnya, lihat Penyimpanan data proyek.
6. Untuk Gunakan evaluasi sisi klien, pilih apakah Anda ingin mengaktifkan evaluasi sisi klien untuk proyek ini, atau tidak. Dengan evaluasi sisi klien, aplikasi Anda dapat menetapkan variasi ke sesi pengguna secara lokal, bukan dengan memanggil operasi. EvaluateFeature Hal ini akan mengurangi risiko latensi dan ketersediaan yang menyertai sebuah panggilan API. Untuk informasi selengkapnya, lihat Menggunakan evaluasi sisi klien - didukung oleh AWS AppConfig.

   Untuk membuat sebuah proyek dengan evaluasi sisi klien, Anda harus memiliki izin `evidently:ExportProjectAsConfiguration`.

   Jika Anda mengaktifkan evaluasi sisi klien, Anda juga perlu melakukan hal berikut:

   a. Pilih apakah akan menggunakan AWS AppConfig aplikasi yang sudah ada atau membuat yang baru.
   b. Pilih apakah akan menggunakan AWS AppConfig lingkungan yang ada atau membuat yang baru.

      Untuk informasi selengkapnya tentang aplikasi dan lingkungan di AWS AppConfig, lihat Cara AWS AppConfig kerja.
7. (Opsional) Untuk menambahkan tag ke proyek ini, silakan pilih Tag, Tambahkan tag baru.

   Kemudian, untuk Kunci, masukkan nama untuk tag tersebut. Anda dapat menambahkan sebuah nilai opsional untuk tanda di Nilai.

   Untuk menambahkan tanda lainnya, silakan pilih Tambahkan tanda baru lagi.

Untuk informasi selengkapnya, lihat Menandai AWS Sumber Daya.

8. Pilih Buat proyek.

Menggunakan evaluasi sisi klien - didukung oleh AWS AppConfig

Anda dapat menggunakan evaluasi sisi klien - didukung oleh AWS AppConfig (evaluasi sisi klien) dalam proyek, yang memungkinkan aplikasi Anda menetapkan variasi ke sesi pengguna secara lokal alih-alih menetapkan variasi dengan memanggil operasi. EvaluateFeature Hal ini akan mengurangi risiko latensi dan ketersediaan yang menyertai sebuah panggilan API.

Untuk menggunakan evaluasi sisi klien, lampirkan ekstensi AWS AppConfig Lambda sebagai lapisan ke fungsi Lambda Anda dan konfigurasikan variabel lingkungan. Evaluasi sisi klien ini akan berjalan sebagai proses sampingan pada host lokal. Kemudian, Anda dapat memanggil EvaluationFeaturedan PutProjectEventoperasi melawan`localhost`. Proses evaluasi sisi klien tersebut akan menangani penugasan variasi, caching, dan sinkronisasi data. Untuk informasi selengkapnya AWS AppConfig, lihat Cara AWS AppConfig kerja.

Saat Anda berintegrasi dengan AWS AppConfig, Anda menentukan ID AWS AppConfig aplikasi dan ID AWS AppConfig lingkungan ke Evidently. Anda dapat menggunakan ID aplikasi dan ID lingkungan yang sama di seluruh proyek-proyek Evidently.

Saat Anda membuat proyek dengan evaluasi sisi klien diaktifkan, Terbukti membuat profil AWS AppConfig konfigurasi untuk proyek itu. Profil konfigurasi untuk masing-masing proyek akan berbeda.

Kontrol akses evaluasi sisi klien

Evaluasi sisi klien Evidently menggunakan mekanisme kontrol akses yang berbeda dari yang dilakukan untuk Evidently lainnya. Kami sangat menyarankan agar Anda memahami hal ini sehingga Anda dapat menerapkan langkah-langkah keamanan yang semestinya.

Dengan Evidently, Anda dapat membuat kebijakan IAM yang membatasi tindakan-tindakan yang dapat dilakukan pengguna pada sumber daya individu. Misalnya, Anda dapat membuat peran pengguna yang melarang pengguna melakukan EvaluateFeaturetindakan. Untuk informasi selengkapnya tentang tindakan Terbukti yang dapat dikontrol dengan kebijakan IAM, lihat Tindakan yang didefinisikan oleh Amazon CloudWatch dengan jelas.

Model evaluasi sisi klien ini akan memungkinkan evaluasi secara lokal atas fitur-fitur yang dimiliki Evidently yang menggunakan metadata proyek. Pengguna proyek dengan evaluasi sisi klien

diaktifkan dapat memanggil EvaluateFeatureAPI terhadap titik akhir host lokal, dan panggilan API ini tidak mencapai Terbukti dan tidak diautentikasi oleh kebijakan IAM layanan Evidently. Panggilan ini berhasil bahkan jika pengguna tidak memiliki izin IAM untuk menggunakan EvaluateFeaturetindakan. Namun, pengguna masih memerlukan PutProjectEventsizin bagi agen untuk menyangga peristiwa evaluasi atau peristiwa khusus dan untuk membongkar data ke Terbukti secara asinkron.

Selain itu, seorang pengguna harus memiliki izin `evidently:ExportProjectAsConfiguration` untuk dapat membuat sebuah proyek yang menggunakan evaluasi sisi klien. Ini membantu Anda mengontrol akses ke EvaluateFeaturetindakan yang dipanggil selama evaluasi sisi klien.

Jika Anda tidak berhati-hati, model keamanan evaluasi sisi klien dapat menumbangkan kebijakan yang telah Anda tetapkan pada Evidently lainnya. Pengguna yang memiliki `evidently:ExportProjectAsConfiguration` izin dapat membuat proyek dengan evaluasi sisi klien diaktifkan, dan kemudian menggunakan EvaluateFeaturetindakan untuk evaluasi sisi klien dengan proyek itu bahkan jika mereka secara tegas ditolak tindakan dalam EvaluateFeaturekebijakan IAM.

Memulai Lambda

Evidently saat ini mendukung evaluasi sisi klien dengan menggunakan lingkungan AWS Lambda . Untuk memulai, pertama-tama tentukan AWS AppConfig aplikasi dan lingkungan mana yang akan digunakan. Pilih sebuah aplikasi dan lingkungan yang ada, atau Anda bisa membuat yang baru.

Contoh AWS AppConfig AWS CLI perintah berikut membuat aplikasi dan lingkungan.

```
aws appconfig create-application --name YOUR_APP_NAME
```

```
aws appconfig create-environment --application-id YOUR_APP_ID --
name YOUR_ENVIRONMENT_NAME
```

Selanjutnya, buat proyek Evidently dengan menggunakan AWS AppConfig sumber daya ini. Untuk informasi selengkapnya, lihat Membuat sebuah proyek baru.

Evaluasi sisi klien didukung di Lambda dengan menggunakan sebuah lapisan Lambda. Ini adalah lapisan publik yang merupakan bagian dari`AWS-AppConfig-Extension`, AWS AppConfig ekstensi publik yang dibuat oleh AWS AppConfig layanan. Untuk informasi selengkapnya tentang lapisan Lambda, silakan lihat Lapisan.

Cara menggunakan evaluasi sisi klien, Anda harus menambahkan lapisan ini ke fungsi Lambda Anda dan kemudian melakukan konfigurasi izin dan variabel lingkungan.

Untuk menambahkan lapisan Lambda evaluasi sisi klien Evidently ke fungsi Lambda Anda dan mengonfigurasikannya

1. Buatlah sebuah fungsi Lambda, jika Anda belum membuatnya.
2. Tambahkan lapisan evaluasi sisi klien ke fungsi Anda. Anda dapat menentukan ARN atau memilihnya dari daftar AWS lapisan jika Anda belum melakukannya. Untuk informasi selengkapnya, lihat Mengkonfigurasi fungsi untuk menggunakan lapisan dan Versi ekstensi AWS AppConfig Lambda yang tersedia.
3. Buat kebijakan IAM bernama EvidentlyAppConfigCachingAgentPolicydengan konten berikut, dan lampirkan ke peran eksekusi fungsi. Untuk informasi selengkapnya, silakan lihat Peran eksekusi Lambda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "appconfig:GetLatestConfiguration",
                "appconfig:StartConfigurationSession",
                "evidently:PutProjectEvents"
            ],
            "Resource": "*"
        }
    ]
}
```

4. Menambahkan variabel lingkungan yang diperlukan `AWS_APPCONFIG_EXTENSION_EVIDENTLY_CONFIGURATIONS` ke fungsi Lambda Anda. Variabel lingkungan ini menentukan pemetaan antara proyek Evidently dan sumber daya. AWS AppConfig

   Jika Anda menggunakan fungsi ini untuk satu proyek Evidently, Anda harus menetapkan nilai variabel lingkungan dengan: `applications/`*`APP_ID`*`/environments/`*`ENVIRONMENT_ID`*`/configurations/`*`PROJECT_NAME`*

   Jika Anda menggunakan fungsi ini untuk beberapa proyek Evidently, maka Anda harus menggunakan koma untuk memisahkan nilai, seperti pada contoh berikut: `applications/`*`APP_ID_1`*`/environments/`*`ENVIRONMENT_ID_1`*`/`

`configurations/`*`PROJECT_NAME_1`*`, applications/`*`APP_ID_2`*`/environments/`*`ENVIRONMENT_ID_2`*`/configurations/`*`PROJECT_NAME_2`*

5. (Opsional) Tetapkan variabel-variabel lingkungan lainnya. Untuk informasi selengkapnya, lihat Mengonfigurasi ekstensi AWS AppConfig Lambda.

6. Dalam aplikasi Anda, Anda harus mendapatkan evaluasi Evidently secara lokal dengan mengirim `EvaluateFeature` ke `localhost`.

   Contoh Python:

```
import boto3
from botocore.config import Config

def lambda_handler(event, context):
    local_client = boto3.client(
        'evidently',
        endpoint_url="http://localhost:2772",
        config=Config(inject_host_prefix=False)
    )
    response = local_client.evaluate_feature(
        project=event['project'],
        feature=event['feature'],
        entityId=event['entityId']
    )
    print(response)
```

   Contoh Node.js:

```
const AWS = require('aws-sdk');
const evidently = new AWS.Evidently({
    region: "us-west-2",
    endpoint: "http://localhost:2772",
    hostPrefixEnabled: false
});

exports.handler = async (event) => {

    const evaluation = await evidently.evaluateFeature({
        project: 'John_ETCProject_Aug2022',
        feature: 'Feature_IceCreamFlavors',
        entityId: 'John'
    }).promise()
```

```
    console.log(evaluation)
    const response = {
        statusCode: 200,
        body: evaluation,
    };
    return response;
};
```

Contoh Kotlin:

```
String localhostEndpoint = "http://localhost:2772/"
public AmazonCloudWatchEvidentlyClient getEvidentlyLocalClient() {
    return AmazonCloudWatchEvidentlyClientBuilder.standard()

 .withEndpointConfiguration(AwsClientBuilder.EndpointConfiguration(localhostEndpoint,
 region))

 .withClientConfiguration(ClientConfiguration().withDisableHostPrefixInjection(true))
        .withCredentials(credentialsProvider)
        .build();
}

AmazonCloudWatchEvidentlyClient evidently = getEvidentlyLocalClient();

// EvaluateFeature via local client.
EvaluateFeatureRequest evaluateFeatureRequest = new
 EvaluateFeatureRequest().builder()
   .withProject(${YOUR_PROJECT}) //Required.
   .withFeature(${YOUR_FEATURE}) //Required.
   .withEntityId(${YOUR_ENTITY_ID}) //Required.
   .withEvaluationContext(${YOUR_EVAL_CONTEXT}) //Optional: a JSON object of
 attributes that you can optionally pass in as part of the evaluation event sent to
 Evidently.
   .build();

EvaluateFeatureResponse evaluateFeatureResponse =
 evidently.evaluateFeature(evaluateFeatureRequest);

// PutProjectEvents via local client.
PutProjectEventsRequest putProjectEventsRequest = new
 PutProjectEventsRequest().builder()
    .withData(${YOUR_DATA})
```

```
    .withTimeStamp(${YOUR_TIMESTAMP})
    .withType(${YOUR_TYPE})
    .build();

PutProjectEvents putProjectEventsResponse =
 evidently.putProjectEvents(putProjectEventsRequest);
```

Mengonfigurasikan seberapa sering klien mengirim data ke Evidently

Untuk menentukan seberapa sering evaluasi sisi klien mengirim data ke Evidently, Anda dapat secara opsional mengonfigurasi dua variabel lingkungan.

- `AWS_APPCONFIG_EXTENSION_EVIDENTLY_EVENT_BATCH_SIZE` menentukan jumlah peristiwa per proyek untuk batch sebelum mengirimnya ke Evidently. Nilai yang valid adalah bilangan bulat antara 1 dan 50, dan default-nya adalah 40.
- `AWS_APPCONFIG_EXTENSION_EVIDENTLY_BATCH_COLLECTION_DURATION` menentukan durasi dalam hitungan detik untuk menunggu peristiwa sebelum mengirimnya ke Evidently. Bawaannya adalah 30.

Pemecahan Masalah

Gunakan informasi berikut untuk membantu memecahkan masalah dengan menggunakan CloudWatch Evidently dengan evaluasi sisi klien - didukung oleh. AWS AppConfig

Terjadi kesalahan (BadRequestException) saat memanggil EvaluateFeature operasi: Metode HTTP tidak didukung untuk jalur yang disediakan

Variabel lingkungan Anda mungkin tidak dikonfigurasi dengan benar. Sebagai contoh, Anda mungkin telah menggunakan `EVIDENTLY_CONFIGURATIONS` sebagai nama variabel lingkungan, bukan `AWS_APPCONFIG_EXTENSION_EVIDENTLY_CONFIGURATIONS`.

ResourceNotFoundException: Penerapan tidak ditemukan

Pembaruan Anda ke metadata proyek belum diterapkan ke AWS AppConfig. Periksa penerapan aktif di AWS AppConfig lingkungan yang Anda gunakan untuk evaluasi sisi klien.

ValidationException: Tidak Jelas konfigurasi untuk proyek

Variabel lingkungan `AWS_APPCONFIG_EXTENSION_EVIDENTLY_CONFIGURATIONS` Anda mungkin dikonfigurasi dengan nama proyek yang salah.

Menambahkan sebuah fitur ke proyek

Fitur di CloudWatch Evidently mewakili fitur yang ingin Anda luncurkan atau yang ingin Anda uji variasinya.

Sebelum Anda dapat menambahkan fitur, Anda harus membuat sebuah proyek. Untuk informasi selengkapnya, lihat Membuat sebuah proyek baru.

Cara menambahkan sebuah fitur ke proyek

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Sinyal Aplikasi, Terbukti.
3. Pilih nama proyek.
4. Pilih Tambahkan fitur.
5. Untuk Nama fitur, masukkan nama yang akan digunakan untuk mengidentifikasi fitur ini dalam proyek ini.

   Secara opsional, Anda dapat menambahkan sebuah deskripsi fitur.
6. Untuk Variasi fitur, untuk Jenis variasi, pilih Boolean, Long, Double, atau String. Untuk informasi selengkapnya, lihat Jenis-jenis variasi.
7. Anda bisa menambahkan hingga lima variasi untuk fitur Anda. Nilai untuk masing-masing variasi harus valid untuk Jenis variasi yang sudah Anda pilih.

   Tentukan salah satu variasi yang akan dijadikan default. Variasi default ini adalah garis dasar bahwa variasi-variasi lain akan dibandingkan dengan, dan harus menjadi variasi yang sedang disajikan kepada pengguna Anda sekarang. Variasi ini juga akan menjadi variasi yang akan disajikan kepada para pengguna yang tidak ditambahkan ke peluncuran atau percobaan untuk fitur ini.
8. Pilih Kode contoh. Contoh kode ini menunjukkan hal-hal yang perlu ditambahkan ke aplikasi Anda untuk menyiapkan variasi dan menetapkan sesi pengguna untuk variasi tersebut. Anda dapat memilih antara JavaScript, Java, dan Python untuk kode.

   Anda tidak perlu menambahkan kode tersebut ke aplikasi Anda sekarang, tetapi Anda harus menambahkannya sebelum memulai peluncuran atau percobaan.

   Untuk informasi selengkapnya, lihat Menambahkan kode ke aplikasi Anda.
9. (Opsional) Untuk menentukan bahwa pengguna tertentu dapat selalu melihat variasi tertentu, Anda bisa memilih Override, Add override. Kemudian, tetapkan pengguna dengan memasukkan

ID pengguna, ID akun, atau pengenal lain di Pengidentifikasi, dan tentukan variasi mana yang akan ditampilkan untuk mereka.

Hal ini dapat berguna bagi para anggota tim pengujian Anda sendiri atau pengguna internal lainnya ketika Anda ingin memastikan agar para anggota itu melihat variasi tertentu. Sesi pengguna yang mendapatkan penetapan override tidak akan berkontribusi pada metrik peluncuran atau percobaan.

Anda dapat mengulangi ini untuk sebanyak 20 pengguna dengan memilih Add override lagi.

10. (Opsional) Untuk menambahkan tag ke fitur ini, silakan pilih Tag, Tambahkan tag baru.

    Kemudian, untuk Kunci, masukkan nama untuk tag tersebut. Anda dapat menambahkan sebuah nilai opsional untuk tanda di Nilai.

    Untuk menambahkan tanda lainnya, silakan pilih Tambahkan tanda baru lagi.

    Untuk informasi selengkapnya, lihat Menandai AWS Sumber Daya.
11. Pilih Tambahkan fitur.

Jenis-jenis variasi

Saat Anda membuat sebuah sebuah fitur dan menentukan variasi-variasi tersebut, Anda harus memilih sebuah jenis variasi. Jenis-jenis yang mungkin muncul adalah:

- Boolean
- Bilangan bulat panjang
- Nomor titik mengambang presisi ganda
- String

Jenis variasi menetapkan bagaimana variasi-variasi yang berbeda tersebut dibedakan dalam kode Anda. Anda dapat menggunakan jenis variasi untuk menyederhanakan implementasi CloudWatch Evidently dan juga untuk menyederhanakan proses memodifikasi fitur dalam peluncuran dan eksperimen Anda.

Sebagai contoh, jika Anda menentukan sebuah fitur dengan tipe variasi bilangan bulat panjang, maka bilangan bulat yang Anda tentukan untuk membedakan variasi dapat berupa angka yang dimasukkan langsung ke kode Anda. Salah satu contohnya mungkin menguji ukuran piksel dari sebuah tombol Nilai untuk jenis variasi tersebut dapat berupa jumlah piksel yang digunakan dalam

masing-masing variasi. Kode untuk masing-masing variasi dapat membaca nilai jenis variasi tersebut dan menggunakannya sebagai ukuran tombol. Untuk menguji ukuran sebuah tombol baru, Anda dapat mengubah nomor yang Anda gunakan untuk nilai variasi, tanpa perlu melakukan terhadap perubahan kode lainnya.

Saat Anda menetapkan nilai untuk jenis variasi Anda dalam suatu fitur, Anda harus menghindari menetapkan nilai yang sama ke beberapa variasi, kecuali jika Anda ingin melakukan pengujian A/A untuk awalnya mencoba CloudWatch Terbukti, atau memiliki alasan lain untuk melakukannya.

Evidently tidak memiliki dukungan asli untuk JSON sebagai sebuah tipe, tetapi Anda dapat memasukkan JSON dalam jenis variasi String, dan melakukan parsing atas JSON itu dalam kode Anda.

Gunakan segmen-segmen untuk memfokuskan pada audiens Anda

Anda dapat menentukan segmen audiens dan menggunakannya dalam peluncuran dan percobaan Anda. Suatu segmen adalah bagian dari audiens Anda yang berbagi satu atau beberapa karakteristik. Contohnya adalah pengguna browser Chrome, pengguna di Eropa, atau pengguna browser Firefox di Eropa yang juga sesuai dengan kriteria-kriteria lain yang dikumpulkan oleh aplikasi Anda, misalnya usia.

Menggunakan sebuah segmen dalam sebuah percobaan akan membatasi percobaan tersebut untuk hanya melakukan evaluasi terhadap pengguna yang cocok dengan kriteria segmen. Saat Anda menggunakan satu atau beberapa segmen dalam suatu peluncuran, Anda dapat menentukan pemisahan lalu lintas yang berbeda untuk segmen audiens yang berbeda itu.

Sintaks pola aturan segmen

Untuk membuat suatu segmen, Anda perlu menentukan sebuah pola aturan segmen. Tentukan atribut-atribut yang ingin Anda gunakan untuk mengevaluasi apakah sesi pengguna akan berada dalam segmen tersebut, atau tidak. Pola yang Anda buat akan dibandingkan dengan nilai `evaluationContext` yang ditemukan dalam sesi pengguna oleh Evidently. Untuk informasi selengkapnya, lihat Menggunakan EvaluateFeature.

Untuk membuat suatu pola aturan segmen, tentukan bidang-bidang yang ingin Anda cocokkan dengan pola. Anda juga dapat menggunakan logika dalam pola Anda, misalnya `And`, `Or`, `Not`, dan `Exists`.

Agar `evaluationContext` mencocokkan suatu pola, `evaluationContext` harus mencocokkan dengan semua bagian dari pola aturan tersebut. Evidently mengabaikan bidang-bidang yang ada di `evaluationContext` yang tidak disertakan dalam pola aturan.

Nilai-nilai yang cocok dengan pola aturan tersebut mengikuti aturan JSON. Anda dapat menyertakan string yang diapit oleh tanda kutip ("), angka, dan kata kunci `true`, `false`, dan `null`.

Untuk string, Terbukti menggunakan character-by-character pencocokan yang tepat tanpa case-folding atau normalisasi string lainnya. Oleh karena itu, kecocokan peraturan peka terhadap huruf besar-kecil. Sebagai contoh, jika `evaluationContext` Anda menyertakan atribut `browser` tetapi pola aturan Anda memeriksa `Browser`, maka itu tidak akan cocok.

Untuk angka, Evidently menggunakan pernyataan string. Sebagai contoh, 300, 300,0, dan 3,0e2 dianggap tidak sama.

Ketika Anda menulis pola aturan untuk mencocokkan `evaluationContext`, Anda dapat menggunakan API `TestSegmentPattern` atau perintah CLI `test-segment-pattern` untuk menguji apakah pola Anda cocok dengan JSON yang benar. Untuk informasi lebih lanjut, lihat TestSegmentPattern.

Ringkasan berikut menunjukkan semua operator-operator perbandingan yang tersedia dalam pola-pola segmen Evidently.

Perbandingan	Contoh	Sintaks aturan
Kosong	UserID adalah kosong	`{ "UserID": [ null ] }`
Kosong	LastName kosong	`{ "LastName": [""] }`
Setara	Browser adalah "Chrome"	`{ "Browser": [ "Chrome" ] }`

Perbandingan	Contoh	Sintaks aturan
Dan	Negara adalah "Prancis" dan Perangkat adalah "Seluler"	`{ "Country": [ "France" ], "Device": ["Mobile"] }`
Atau (beberapa nilai dari satu atribut tunggal)	Browser adalah "Chrome" atau "Firefox"	`{ "Browser": ["Chrome", "Firefox"] }`
Atau (atribut-atribut berbeda)	Browser adalah "Safari" atau Perangkat adalah "Tablet"	`{ "$or": [ {"Browser": ["Safari"]}, {"Device": ["Tablet" }] ] }`
Bukan	Browser bisa semuanya kecuali "Safari"	`{ "Browser": [ { "anything-but": [ "Safari" ] } ] }`
Numerik (sama dengan)	Harga 100	`{ "Price": [ { "numeric": [ "=", 100 ] } ] }`

Perbandingan	Contoh	Sintaks aturan
Numerik (rentang)	Harga lebih dari 10, dan kurang dari atau sama dengan 20	`{ "Price": [ { "numeric": [ ">", 10, "<=", 20 ] } ] }`
Ada	Bidang usia ada	`{ "Age": [ { "exists": true } ] }`
Tidak ada	Bidang usia tidak ada	`{ "Age": [ { "exists": false } ] }`
Dimulai dengan sebuah awalan	Wilayah berada di Amerika Serikat	`{ "Region": [ {"prefix": "us-" } ] }`
Diakhiri dengan akhiran	Lokasi memiliki akhiran "Barat"	`{ "Region": [ {"suffix": "West" } ] }`

Contoh-contoh aturan segmen

Semua contoh-contoh berikut mengasumsikan bahwa Anda meneruskan nilai `evaluationContext` dengan label bidang dan nilai yang sama yang Anda gunakan pola-dalam pola aturan Anda.

Contoh berikut cocok jika `Browser` adalah Chrome atau Firefox dan `Location` adalah AS-Barat.

```
{
```

```
  "Browser": ["Chrome", "Firefox"],
  "Location": ["US-West"]
}
```

Contoh berikut cocok jika `Browser` ada browser kecuali Chrome, `Location` dimulai dengan`US`, dan ada bidang `Age`.

```
{
  "Browser": [ {"anything-but": ["Chrome"]}],
  "Location": [{"prefix": "US"}],
  "Age": [{"exists": true}]
}
```

Contoh berikut cocok jika `Location` adalah Jepang dan `Browser` adalah Safari atau `Device` adalah Tablet.

```
{
 "Location": ["Japan"],
 "$or": [
   {"Browser": ["Safari"]},
   {"Device": ["Tablet"]}
 ]
}
```

Membuat sebuah segmen

Setelah Anda membuat sebuah segmen, Anda dapat menggunakannya dalam sebuah peluncuran atau percobaan apa pun di proyek apa pun.

Cara membuat sebuah segmen

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Sinyal Aplikasi, Terbukti.
3. Pilih tab Segmen.
4. Pilih Buat segmen.
5. Untuk Nama segmen, masukkan nama yang akan digunakan untuk mengidentifikasi segmen ini.

   Secara opsional, tambahkan sebuah deskripsi.
6. Untuk Pola segmen, masukkan blok JSON yang mendefinisikan pola aturan. Untuk informasi selengkapnya tentang sintaks pola aturan, silakan lihat Sintaks pola aturan segmen.

Membuat sebuah peluncuran

Untuk mengekspos sebuah fitur baru atau mengubah persentase tertentu dari pengguna Anda, Anda harus membuat sebuah peluncuran. Anda kemudian dapat memantau metrik utama seperti waktu pemuatan halaman dan konversi sebelum meluncurkan fitur secara gradual ke semua pengguna Anda.

Sebelum Anda dapat menambahkan sebuah peluncuran, Anda harus membuat sebuah proyek terlebih dahulu. Untuk informasi selengkapnya, lihat Membuat sebuah proyek baru.

Ketika Anda menambahkan sebuah peluncuran, Anda dapat menggunakan fitur yang sudah Anda buat, atau Anda bisa membuat sebuah fitur baru saat Anda membuat peluncuran tersebut.

Cara menambahkan sebuah peluncuran untuk suatu proyek

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Sinyal Aplikasi, Terbukti.
3. Pilih tombol yang ada di sebelah nama proyek dan pilih Tindakan proyek, Buat peluncuran.
4. Untuk Nama peluncuran, masukkan nama yang akan digunakan untuk mengidentifikasi fitur ini dalam proyek ini.

   Secara opsional, Anda dapat menambahkan sebuah deskripsi.
5. Pilih salah satu dari Pilih dari fitur yang ada atau Tambahkan fitur baru.

   Jika Anda menggunakan sebuah fitur yang ada, pilih fitur itu pada Nama fitur.

   Jika Anda memilih Tambahkan fitur baru, Anda perlu melakukan hal berikut:

   a. Untuk nama Fitur, masukkan nama yang akan digunakan untuk mengidentifikasi fitur ini dalam proyek ini. Secara opsional, Anda dapat menambahkan sebuah deskripsi.
   b. Untuk Variasi fitur, untuk Jenis variasi, pilih Boolean, Long, Double, atau String. Untuk informasi selengkapnya, lihat Jenis-jenis variasi.
   c. Anda bisa menambahkan hingga lima variasi untuk fitur Anda. Nilai untuk masing-masing variasi harus valid untuk Jenis variasi yang sudah Anda pilih.

      Tentukan salah satu variasi yang akan dijadikan default. Variasi default ini adalah garis dasar bahwa variasi-variasi lain akan dibandingkan dengan, dan harus menjadi variasi yang sedang disajikan kepada pengguna Anda sekarang. Jika Anda menghentikan sebuah percobaan, variasi default ini kemudian akan disajikan ke semua pengguna.

d. Pilih Kode contoh. Contoh kode ini menunjukkan hal-hal yang perlu ditambahkan ke aplikasi Anda untuk menyiapkan variasi dan menetapkan sesi pengguna untuk variasi tersebut. Anda dapat memilih antara JavaScript, Java, dan Python untuk kode.

   Anda tidak perlu menambahkan kode tersebut ke aplikasi Anda sekarang, tetapi Anda harus menambahkannya sebelum memulai peluncuran.

   Untuk informasi selengkapnya, lihat Menambahkan kode ke aplikasi Anda.

6. Untuk Konfigurasi peluncuran, Anda bisa memilih apakah akan segera memulai peluncuran atau menjadwalkan untuk memulai nanti.
7. (Opsional) Untuk menentukan pemisahan lalu lintas yang berbeda untuk segmen audiens yang telah Anda tentukan, Anda tidak bisa melakukan pemisahan lalu lintas yang akan Anda gunakan untuk audiens umum, oleh karena itu pilih Tambahkan Penggantian Segmen.

   Di Penggantian Segmen, silakan pilih segmen dan tentukan pemisahan lalu lintas yang akan digunakan untuk segmen tersebut.

   Jika Anda mau, Anda dapat menentukan lebih banyak segmen untuk menentukan pemisahan lalu lintas dengan memilih Tambah Penggantian Segmen. Sebuah peluncuran dapat memiliki hingga enam penggantian segmen.

   Untuk informasi selengkapnya, lihat Gunakan segmen-segmen untuk memfokuskan pada audiens Anda.
8. Untuk Konfigurasi lalu lintas, pilih persentase lalu lintas yang akan Anda tetapkan untuk masing-masing variasi untuk audiens umum yang tidak cocok dengan penggantian segmen. Anda juga dapat memilih untuk mengecualikan variasi agar tidak disajikan kepada para pengguna.

   Ringkasan lalu lintas menunjukkan berapa banyak lalu lintas keseluruhan Anda yang tersedia untuk peluncuran ini.
9. Jika Anda memilih untuk menjadwalkan peluncuran tersebut untuk memulai nanti, maka Anda dapat menambahkan beberapa langkah lain untuk peluncuran tersebut. Masing-masing langkah dapat menggunakan persentase yang berbeda untuk menyajikan variasi. Untuk melakukan hal ini, Anda harus memilih Tambahkan langkah lain dan kemudian tentukan jadwal dan persentase lalu lintas untuk langkah berikutnya. Anda dapat memasukkan sebanyak lima langkah dalam suatu peluncuran.

10. Jika Anda ingin melacak performa fitur Anda dengan metrik selama peluncuran tersebut, silakan pilih Metrik, Tambah metrik. Anda dapat menggunakan metrik CloudWatch RUM atau metrik khusus.

    Untuk menggunakan metrik kustom, Anda dapat membuat metrik di sini menggunakan EventBridge aturan Amazon. Cara membuat metrik kustom, lakukan hal-hal berikut:

    - Pilih Metrik kustom dan masukkan nama untuk metrik tersebut.
    - Pada Aturan metrik, untuk ID Entitas, masukkan cara untuk mengidentifikasi entitas tersebut. Entitas ini bisa berupa pengguna atau sesi yang melakukan tindakan yang menyebabkan sebuah nilai metrik direkam. Contohnya adalah `userDetails.userID`.
    - Untuk Kunci nilai, Anda perlu memasukkan nilai yang akan dilacak untuk menghasilkan metrik.
    - Jika Anda mau, masukkan sebuah nama untuk satuan untuk metrik tersebut. Nama unit ini hanya untuk tampilan saja, untuk digunakan pada grafik yang ada di konsol Evidently.

    Saat Anda memasukkan bidang tersebut, kotak tersebut menunjukkan contoh cara mengkodekan EventBridge aturan untuk membuat metrik. Untuk informasi selengkapnya EventBridge, lihat Apa itu Amazon EventBridge?

    Untuk menggunakan metrik-metrik RUM, Anda harus sudah menyiapkan sebuah monitor aplikasi RUM untuk aplikasi Anda. Untuk informasi selengkapnya, lihat Siapkan aplikasi untuk menggunakan CloudWatch RUM.

    > **Note**
    >
    > Jika Anda menggunakan metrik-metrik RUM, dan Anda tidak mengonfigurasi monitor aplikasi untuk mengambil sampel 100% sesi pengguna, maka tidak semua sesi pengguna yang berpartisipasi dalam peluncuran akan mengirim metrik ke Evidently. Untuk memastikan metrik-metrik peluncuran akurat, kami menyarankan agar monitor aplikasi menggunakan 100% sesi pengguna untuk pengambilan sampel.

11. (Opsional) Jika Anda membuat setidaknya satu metrik untuk peluncuran, Anda dapat mengaitkan CloudWatch alarm yang ada dengan peluncuran ini. Untuk melakukannya, pilih CloudWatch Alarm asosiasi.

Saat Anda mengaitkan alarm dengan peluncuran, CloudWatch Terbukti harus menambahkan tag ke alarm dengan nama proyek dan nama peluncuran. Ini agar CloudWatch Terbukti dapat menampilkan alarm yang benar dalam informasi peluncuran di konsol.

Untuk mengetahui bahwa CloudWatch Evidently akan menambahkan tag ini, pilih Izinkan Terbukti untuk menandai sumber daya alarm yang diidentifikasi di bawah ini dengan sumber daya peluncuran ini. Kemudian, pilih Kaitkan alarm dan masukkan nama alarm.

Untuk informasi tentang membuat CloudWatch alarm, lihat Menggunakan CloudWatch alarm Amazon.

12. (Opsional) Untuk menambahkan tag ke peluncuran ini, silakan pilih Tag, Tambahkan tag baru.

    Kemudian, untuk Kunci, masukkan nama untuk tag tersebut. Anda dapat menambahkan sebuah nilai opsional untuk tanda di Nilai.

    Untuk menambahkan tanda lainnya, silakan pilih Tambahkan tanda baru lagi.

    Untuk informasi selengkapnya, lihat Menandai AWS Sumber Daya.

13. Pilih Buat peluncuran.

Membuat sebuah percobaan

Gunakan percobaan untuk menguji berbagai versi dari sebuah fitur atau situs web serta mengumpulkan data dari sesi pengguna nyata. Dengan cara ini, Anda akan dapat membuat pilihan-pilihan untuk aplikasi Anda berdasarkan bukti dan data.

Sebelum Anda dapat menambahkan sebuah percobaan, Anda harus membuat sebuah proyek terlebih dahulu. Untuk informasi selengkapnya, lihat Membuat sebuah proyek baru.

Ketika Anda menambahkan sebuah percobaan, Anda dapat menggunakan fitur yang sudah Anda buat, atau Anda bisa membuat sebuah fitur baru saat Anda membuat percobaan tersebut.

Untuk menambahkan sebuah percobaan ke sebuah proyek

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Sinyal Aplikasi, Terbukti.
3. Pilih tombol yang ada di sebelah nama proyek dan pilih Tindakan proyek, Buat percobaan.

4. Untuk Nama percobaan, masukkan nama yang akan digunakan untuk mengidentifikasi fitur ini dalam proyek ini.

   Secara opsional, Anda dapat menambahkan sebuah deskripsi.

5. Pilih salah satu dari Pilih dari fitur yang ada atau Tambahkan fitur baru.

   Jika Anda menggunakan sebuah fitur yang ada, pilih fitur itu pada Nama fitur.

   Jika Anda memilih Tambahkan fitur baru, Anda perlu melakukan hal berikut:

   a. Untuk nama Fitur, masukkan nama yang akan digunakan untuk mengidentifikasi fitur ini dalam proyek ini. Jika Anda mau, Anda dapat memasukkan sebuah deskripsi.

   b. Untuk Variasi fitur, untuk Jenis variasi, pilih Boolean, Long, Double, atau String. Tipe tersebut akan mendefinisikan jenis nilai yang digunakan untuk masing-masing variasi. Untuk informasi selengkapnya, lihat Jenis-jenis variasi.

   c. Anda bisa menambahkan hingga lima variasi untuk fitur Anda. Nilai untuk masing-masing variasi harus valid untuk Jenis variasi yang sudah Anda pilih.

      Tentukan salah satu variasi yang akan dijadikan default. Variasi default ini adalah garis dasar bahwa variasi-variasi lain akan dibandingkan dengan, dan harus menjadi variasi yang sedang disajikan kepada pengguna Anda sekarang. Jika Anda menghentikan sebuah percobaan yang menggunakan fitur ini, maka variasi default-nya akan disajikan ke persentase pengguna yang berada dalam percobaan sebelumnya.

   d. Pilih Kode contoh. Contoh kode ini menunjukkan hal-hal yang perlu ditambahkan ke aplikasi Anda untuk menyiapkan variasi dan menetapkan sesi pengguna untuk variasi tersebut. Anda dapat memilih antara JavaScript, Java, dan Python untuk kode.

      Anda tidak perlu menambahkan kode tersebut ke aplikasi Anda sekarang, tetapi Anda harus menambahkannya sebelum memulai percobaan. Untuk informasi selengkapnya, lihat Menambahkan kode ke aplikasi Anda.

6. Untuk Audiens, secara opsional, silakan pilih sebuah segmen yang telah dibuat secara opsional jika Anda ingin percobaan ini hanya berlaku untuk para pengguna yang cocok dengan segmen tersebut. Untuk informasi selengkapnya tentang segmen, silakan lihat Gunakan segmen-segmen untuk memfokuskan pada audiens Anda.

7. Untuk Pemisahan lalu lintas untuk percobaan, tentukan persentase audiens yang dipilih yang sesinya akan digunakan dalam percobaan. Kemudian alokasikan lalu lintas itu untuk berbagai variasi yang digunakan percobaan.

Jika sebuah peluncuran dan sebuah percobaan keduanya berjalan pada saat yang sama untuk fitur yang sama pula, maka audiens pertama-tama akan diarahkan ke peluncuran. Kemudian, persentase lalu lintas yang ditentukan untuk peluncuran tersebut harus diambil dari keseluruhan audiens. Setelah itu, persentase yang Anda tentukan di sini adalah persentase audiens sisa yang digunakan untuk percobaan. Lalu lintas yang tersisa setelah itu akan berfungsi sebagai variasi default.

8. Untuk Metrik, pilih metrik yang akan Anda gunakan untuk melakukan evaluasi terhadap variasi selama percobaan. Anda harus menggunakan paling tidak satu metrik untuk melakukan evaluasi.

   a. Untuk sumber Metrik, pilih apakah akan menggunakan metrik CloudWatch RUM atau metrik khusus.

   b. Masukkan nama metrik. Untuk Sasaran, silakan pilih Naikkan jika Anda menghendaki nilai yang lebih tinggi untuk metrik tersebut untuk menunjukkan variasi yang lebih baik. Pilih Turunkan jika Anda menghendaki nilai yang lebih rendah untuk metrik untuk menunjukkan variasi yang lebih baik.

   c. Jika Anda menggunakan metrik khusus, Anda dapat membuat metrik di sini menggunakan EventBridge aturan Amazon. Cara membuat metrik kustom, lakukan hal-hal berikut:

      - Pada Aturan metrik, untuk ID Entitas, masukkan cara untuk mengidentifikasi entitas ini, hal itu bisa berupa pengguna atau sesi yang melakukan tindakan yang menyebabkan nilai metrik direkam. Contohnya adalah `userDetails.userID`.
      - Untuk Kunci nilai, Anda perlu memasukkan nilai yang akan dilacak untuk menghasilkan metrik.
      - Jika Anda mau, masukkan sebuah nama untuk satuan untuk metrik tersebut. Nama unit ini hanya untuk tampilan saja, untuk digunakan pada grafik yang ada di konsol Evidently.

   Anda hanya dapat menggunakan metrik RUM saja jika Anda telah mengatur RUM untuk memantau aplikasi ini. Untuk informasi selengkapnya, lihat Gunakan CloudWatch RUM.

   > **Note**
   >
   > Jika Anda menggunakan metrik-metrik RUM, dan Anda tidak mengonfigurasi monitor aplikasi untuk mengambil sampel 100% sesi pengguna, maka tidak semua sesi pengguna dalam percobaan akan mengirim metrik ke Evidently. Untuk

memastikan metrik percobaan yang akurat, kami menyarankan agar monitor aplikasi menggunakan 100% sesi pengguna untuk pengambilan sampel.

d. (Opsional) Untuk menambahkan lebih banyak metrik yang akan dievaluasi, Anda bisa memilih Tambahkan metrik. Anda dapat melakukan evaluasi atas tiga metrik selama percobaan.

9. (Opsional) Untuk membuat CloudWatch alarm yang akan digunakan dengan eksperimen ini, pilih CloudWatchalarm. Alarm tersebut dapat memantau apakah perbedaan hasil antara masing-masing variasi dan variasi default lebih besar dari ambang batas yang sudah Anda tentukan. Jika performa variasi lebih buruk daripada variasi default, dan perbedaannya lebih besar dari ambang batas yang Anda tentukan, maka itu membuat alarm beralih statusnya menjadi alarm dan akan memberikan notifikasi kepada Anda terkait hal itu.

   Membuat sebuah alarm di sini akan menciptakan satu alarm untuk masing-masing variasi yang bukan variasi default.

   Jika Anda membuat sebuah alarm, tentukan hal-hal berikut ini:

   - Untuk Nama metrik, pilih metrik percobaan yang akan digunakan untuk alarm.
   - Untuk Kondisi alarm pilih kondisi apa saja yang akan menyebabkan alarm beralih statusnya menjadi alarm, ketika nilai metrik variasi dibandingkan dengan nilai metrik variasi default. Sebagai contoh, pilih Lebih Besar atau Lebih Besar/Sama Dengan jika angka yang lebih tinggi memberikan indikasi yang menunjukkan bahwa variasi menunjukkan bahwa performanya buruk. Hal ini akan sesuai jika metrik tersebut mengukur, misalnya, waktu buka halaman.
   - Masukkan angka untuk ambang batas, yang merupakan perbedaan persentase dalam performa yang akan menyebabkan alarm beralih statusnya menjadi `ALARM`.
   - Untuk Rata-rata selama periode, pilih berapa banyak data metrik untuk setiap variasi yang akan dikumpulkan bersama sebelum dibandingkan.

   Anda dapat memilih Tambahkan alarm baru lagi untuk menambahkan lebih banyak alarm ke percobaan.

   Berikutnya, Anda harus memilih Setel notifikasi untuk alarm dan pilih atau buat topik Amazon Simple Notification Service yang akan Anda kirimi notifikasi alarm. Untuk informasi selengkapnya, silakan lihat Menyiapkan notifikasi Amazon SNS.

10. (Opsional) Untuk menambahkan tag ke percobaan ini, silakan pilih Tag, Tambahkan tag baru.

Kemudian, untuk Kunci, masukkan nama untuk tag tersebut. Anda dapat menambahkan sebuah nilai opsional untuk tanda di Nilai.

Untuk menambahkan tanda lainnya, silakan pilih Tambahkan tanda baru lagi.

Untuk informasi selengkapnya, lihat Menandai AWS Sumber Daya.

11. Pilih Buat percobaan.
12. Jika Anda belum membuatnya, buat varian-varian fitur ke dalam aplikasi Anda.
13. Pilih Selesai. Percobaan tidak akan dimulai sampai Anda memulainya.

Setelah Anda menyelesaikan langkah-langkah dalam prosedur berikut, percobaan akan dimulai segera setelahnya.

Untuk memulai sebuah percobaan yang telah Anda buat

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Sinyal Aplikasi, Terbukti.
3. Pilih nama proyek.
4. Pilih tab Percobaan.
5. Pilih tombol yang ada di sebelah nama percobaan, dan pilih Tindakan, Mulai percobaan.
6. (Opsional) Untuk menampilkan atau mengubah setelan percobaan yang sudah Anda buat saat membuatnya, pilih Penyiapan percobaan.
7. Pilih waktu kapan percobaan akan berakhir.
8. Pilih Mulai percobaan.

   Percobaan akan dimulai saat itu juga.

Mengelola fitur, peluncuran, dan percobaan

Gunakan prosedur-prosedur yang ada di bagian ini untuk mengelola fitur, peluncuran, dan percobaan yang telah Anda buat.

Topik

- Lihat aturan evaluasi saat ini dan lalu lintas audiens untuk sebuah fitur
- Mengubah lalu lintas peluncuran

- Mengubah langkah-langkah masa depan sebuah peluncuran
- Mengubah lalu lintas percobaan
- Menghentikan sebuah peluncuran
- Menghentikan sebuah percobaan

Lihat aturan evaluasi saat ini dan lalu lintas audiens untuk sebuah fitur

Anda dapat menggunakan konsol CloudWatch Evidently untuk melihat bagaimana aturan evaluasi fitur mengalokasikan lalu lintas audiens di antara peluncuran, eksperimen, dan variasi fitur saat ini.

Cara menampilkan lalu lintas pemirsa untuk suatu fitur

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Sinyal Aplikasi, Terbukti.
3. Pilih nama proyek yang berisi fitur tersebut.
4. Pilih tab Fitur.
5. Pilih nama fitur.

   Pada tab Aturan evaluasi, Anda dapat melihat arus lalu lintas audiens untuk fitur Anda, sebagai berikut:

   - Pertama, penggantian akan dievaluasi. Ini menentukan bahwa pengguna tertentu selalu dilayani dengan variasi tertentu. Sesi pengguna yang mendapatkan penetapan override tidak akan berkontribusi pada metrik peluncuran atau percobaan.
   - Berikutnya, lalu lintas yang tersisa tersedia untuk peluncuran yang sedang berlangsung, jika ada. Jika ada peluncuran yang sedang berlangsung, maka tabel yang ada di bagian Peluncuran akan menampilkan nama peluncuran dan lalu lintas peluncuran terbagi di antara variasi fitur. Pada sisi kanan bagian Peluncuran, indikator Lalu lintas akan menampilkan berapa banyak audiens yang tersedia (setelah penggantian) yang dialokasikan untuk peluncuran ini. Sisa lalu lintas yang tidak dialokasikan untuk peluncuran akan mengalir ke percobaan (jika ada) dan kemudian mengalir ke variasi default.
   - Berikutnya, lalu lintas yang tersisa tersedia untuk percobaan yang sedang berlangsung, jika ada. Jika ada percobaan yang sedang berlangsung, maka tabel yang ada di bagian Percobaan akan menampilkan nama dan kemajuan percobaan. Pada sisi kanan bagian Percobaan, indikator Lalu lintas akan menampilkan berapa banyak audiens yang tersedia (setelah penggantian an peluncuran) yang dialokasikan untuk percobaan ini. Sisa lalu lintas

yang tidak dialokasikan untuk peluncuran atau percobaan akan berfungsi sebagai variasi default fitur.

Mengubah lalu lintas peluncuran

Anda dapat mengubah alokasi lalu lintas untuk sebuah peluncuran kapan saja, termasuk saat peluncuran tersebut sedang berlangsung.

Jika Anda memiliki sebuah peluncuran yang sedang berlangsung dan percobaan yang sedang berlangsung untuk fitur yang sama, maka perubahan apa pun pada lalu lintas fitur akan menyebabkan terjadinya perubahan pada lalu lintas percobaan. Hal ini karena audiens yang tersedia untuk percobaan adalah bagian dari total audiens Anda yang belum dialokasikan untuk peluncuran. Meningkatkan lalu lintas peluncuran akan mengurangi audiens yang tersedia untuk percobaan, dan mengurangi lalu lintas peluncuran atau mengakhiri peluncuran akan meningkatkan audiens yang tersedia untuk percobaan.

Cara mengubah alokasi lalu lintas untuk sebuah peluncuran

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Sinyal Aplikasi, Terbukti.
3. Pilih nama proyek yang berisi peluncuran tersebut.
4. Pilih tab Peluncuran.
5. Pilih nama peluncuran.

   Pilih Ubah lalu lintas peluncuran.
6. Untuk Serve, pilih persentase lalu lintas baru yang akan ditetapkan untuk masing-masing variasi. Anda juga dapat memilih untuk mengecualikan variasi agar tidak disajikan kepada para pengguna. Saat Anda mengubah nilai-nilai ini, Anda akan dapat melihat efek yang diperbarui pada keseluruhan lalu lintas fitur Anda di Ringkasan lalu lintas.

   Ringkasan Lalu Lintas menunjukkan berapa banyak lalu lintas Anda secara keseluruhan yang tersedia untuk peluncuran ini, dan berapa banyak lalu lintas yang tersedia yang dialokasikan untuk peluncuran ini.
7. Pilih Ubah.

Mengubah langkah-langkah masa depan sebuah peluncuran

Anda dapat mengubah konfigurasi langkah-langkah peluncuran yang belum terjadi, dan juga menambahkan langkah-langkah lainnya ke sebuah peluncuran.

Cara mengubah langkah-langkah sebuah peluncuran

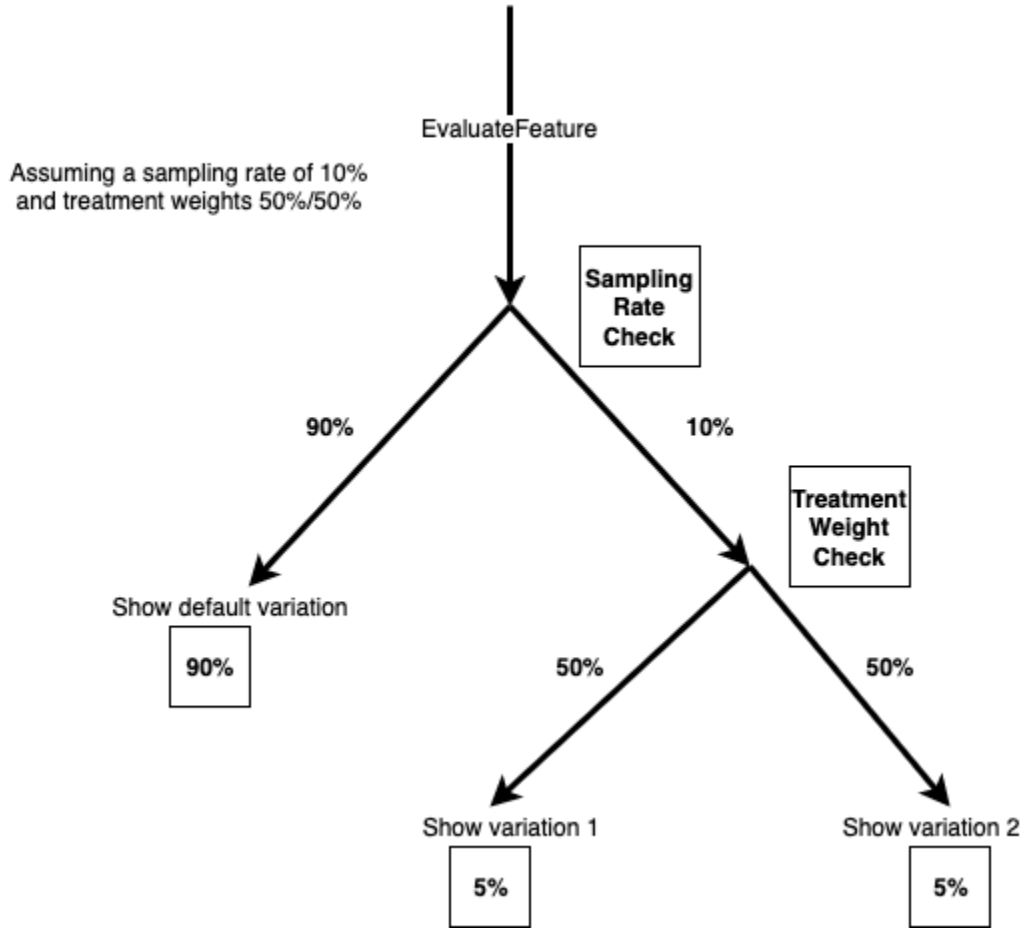
1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Sinyal Aplikasi, Terbukti.
3. Pilih nama proyek yang berisi peluncuran tersebut.
4. Pilih tab Peluncuran.
5. Pilih nama peluncuran.

   Pilih Ubah lalu lintas peluncuran.
6. Pilih Jadwalkan peluncuran.
7. Untuk setiap langkah yang belum dimulai, Anda dapat mengubah persentase audiens yang tersedia yang akan Anda gunakan dalam percobaan. Anda juga dapat mengubah bagaimana lalu lintasnya dialokasikan di antara variasi.

   Anda dapat menambahkan lebih banyak langkah-langkah ke peluncuran tersebut dengan memilih Tambahkan langkah lainnya. Sebuah peluncuran dapat memiliki maksimum lima langkah.
8. Pilih Ubah.

Mengubah lalu lintas percobaan

Anda dapat mengubah laju pengambilan sampel untuk sebuah percobaan kapan saja Anda kehendaki, termasuk ketika percobaan sedang berlangsung. Namun demikian, Anda tidak dapat memperbarui bobot perlakuan setelah sebuah percobaan dieksekusi. Oleh karena itu, Anda dapat mengubah total lalu lintas yang terpapar oleh percobaan tersebut setelah percobaan dieksekusi, tetapi bukan alokasi relatif untuk masing-masing perlakuan. Jika Anda mengubah lalu lintas sebuah percobaan yang sedang berlangsung, kami sarankan Anda hanya menaikkan alokasi lalu lintas, sehingga Anda tidak akan menimbulkan bias.



Cara mengubah alokasi lalu lintas untuk sebuah percobaan

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Pemantauan aplikasi, Evidently.
3. Pilih nama proyek yang berisi peluncuran tersebut.
4. Pilih tab Percobaan.
5. Pilih nama peluncuran.
6. Pilih Ubah lalu lintas percobaan.
7. Masukkan persentase atau gunakan penggeser untuk menentukan berapa banyak lalu lintas yang tersedia yang akan dialokasikan untuk percobaan ini. Lalu lintas yang tersedia adalah total audiens dikurangi lalu lintas yang sudah dialokasikan untuk peluncuran saat ini, jika ada. Lalu lintas yang tidak dialokasikan untuk peluncuran atau percobaan akan berfungsi sebagai variasi default.
8. Pilih Ubah.

Menghentikan sebuah peluncuran

Jika Anda menghentikan sebuah peluncuran yang sedang berlangsung, maka Anda tidak akan dapat melanjutkan kembali atau memulai ulang penghentian itu. Selain itu, penghentian ini tidak akan dievaluasi sebagai aturan untuk alokasi lalu lintas, dan lalu lintas yang dialokasikan untuk peluncuran tersebut akan tersedia untuk percobaan fitur, jika ada. Jika tidak, maka semua lalu lintas akan dijadikan sebagai variasi default setelah peluncuran dihentikan.

Cara menghentikan sebuah peluncuran secara permanen

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Sinyal Aplikasi, Terbukti.
3. Pilih nama proyek yang berisi peluncuran tersebut.
4. Pilih tab Peluncuran.
5. Pilih tombol yang ada di sebelah kiri nama peluncuran.
6. Pilih Tindakan, Batalkan peluncuran atau Tindakan, Tandai telah selesai.

Menghentikan sebuah percobaan

Jika Anda menghentikan sebuah percobaan yang sedang berlangsung, maka Anda tidak akan dapat melanjutkan kembali atau memulai ulang percobaan itu. Bagian lalu lintas yang sebelumnya sudah digunakan dalam percobaan akan dijadikan sebagai variasi default.

Ketika sebuah percobaan tidak dihentikan secara manual dan melewati tanggal berakhirnya, hal itu tidak akan mengubah lalu lintas. Bagian lalu lintas yang sudah dialokasikan untuk percobaan tersebut masih akan dialokasikan untuk percobaan tersebut. Untuk menghentikan ini, dan menyebabkan lalu lintas percobaan dijadikan sebagai variasi default, Anda harus menandai percobaan sudah selesai.

Saat Anda menghentikan sebuah percobaan, Anda dapat memilih untuk membatalkan percobaan atau menandai percobaan sebagai selesai. Jika Anda membatalkan percobaan itu, maka itu akan ditampilkan sebagai Dibatalkan dalam daftar percobaan. Jika Anda memilih untuk menandai percobaan sebagai selesai, maka percobaan itu akan ditampilkan sebagai Selesai.

Cara menghentikan sebuah percobaan secara permanen

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Sinyal Aplikasi, Terbukti.

3. Pilih nama proyek yang berisi percobaan tersebut.
4. Pilih tab Percobaan.
5. Pilih tombol yang ada di sebelah kiri nama percobaan.
6. Pilih Tindakan, Batalkan percobaan atau Tindakan, Tandai sudah selesai.

Menambahkan kode ke aplikasi Anda

Untuk bekerja dengan CloudWatch Evidently, Anda menambahkan kode ke aplikasi Anda untuk menetapkan variasi ke setiap sesi pengguna, dan untuk mengirim metrik ke Evidently. Anda menggunakan `EvaluateFeature` operasi CloudWatch Evidently untuk menetapkan variasi ke sesi pengguna, dan Anda menggunakan `PutProjectEvents` operasi untuk mengirim peristiwa ke Evidently untuk digunakan untuk menghitung metrik untuk peluncuran atau eksperimen Anda.

Saat Anda membuat variasi atau metrik khusus, konsol CloudWatch Evidently menyediakan contoh kode yang perlu Anda tambahkan.

end-to-end Sebagai contoh, lihatTutorial: Pengujian A/B dengan aplikasi sampel Evidently.

Menggunakan EvaluateFeature

Ketika variasi fitur digunakan dalam peluncuran atau percobaan, aplikasi menggunakan EvaluateFeatureoperasi untuk menetapkan setiap sesi pengguna variasi. Penetapan sebuah variasi untuk pengguna adalah sebuah peristiwa evaluasi. Saat Anda memanggil operasi ini, Anda harus memasukkan hal berikut ini:

- Nama fitur– Wajib. Evidently memproses evaluasi sesuai dengan aturan evaluasi fitur peluncuran atau percobaan, dan memilih sebuah variasi untuk entitas.
- EntityID– Wajib. Mewakili pengguna yang unik.
- evaluationContext– Opsional. Objek JSON yang mewakili informasi tambahan tentang seorang pengguna. Evidently akan menggunakan nilai ini untuk mencocokkan pengguna dengan sebuah segmen audiens Anda selama evaluasi-evaluasi fitur, jika Anda telah membuat segmen. Untuk informasi selengkapnya, lihat Gunakan segmen-segmen untuk memfokuskan pada audiens Anda.

  Berikut ini adalah contoh dari sebuah nilai `evaluationContext` yang dapat Anda kirim ke Evidently.

```
{
    "Browser": "Chrome",
```

```
    "Location": {
        "Country": "United States",
        "Zipcode": 98007
    }
}
```

Evaluasi lengket

CloudWatch Terbukti menggunakan evaluasi “lengket”. Sebuah konfigurasi tunggal `entityId`, fitur, konfigurasi fitur, dan `evaluationContext` selalu menerima penetapan variasi yang sama. Satu-satunya waktu saat penetapan variasi ini berubah adalah ketika ada sebuah entitas yang ditambahkan ke penggantian atau ada lalu lintas percobaan yang dipanggil.

Sebuah konfigurasi fitur mencakup hal-hal berikut:

- Variasi-variasi fitur
- Konfigurasi variasi (persentase yang ditetapkan untuk masing-masing variasi) untuk sebuah percobaan yang sedang berjalan untuk fitur ini, jika ada.
- Konfigurasi variasi untuk sebuah peluncuran yang sedang berjalan untuk fitur ini, jika ada. Konfigurasi variasi ini mencakup penggantian segmen yang sudah ditentukan, jika ada.

Jika alokasi lalu lintas untuk sebuah percobaan dinaikkan, setiap `entityId` yang sebelumnya ditetapkan ke sebuah kelompok perlakuan percobaan akan terus menerima perlakuan yang sama. Setiap `entityId` yang sebelumnya ditetapkan ke kelompok kontrol tersebut, dapat ditetapkan ke sebuah kelompok perlakuan percobaan, sesuai dengan konfigurasi variasi yang ditentukan untuk percobaan tersebut.

Jika alokasi lalu lintas dari sebuah percobaan diturunkan, sebuah `entityId` mungkin akan beralih dari sebuah kelompok perlakuan ke kelompok kontrol, tetapi tidak akan masuk ke sebuah kelompok perlakuan yang berbeda.

Menggunakan PutProjectEvents

Untuk mengkodekan metrik khusus untuk Evidently, Anda menggunakan PutProjectEventsoperasi. Berikut ini adalah sebuah contoh muatan sederhana.

```
{
    "events": [
        {
```

```
            "timestamp": {{$timestamp}},
            "type": "aws.evidently.custom",
            "data": "{\"details\": {\"pageLoadTime\": 800.0}, \"userDetails\":
 {\"userId\": \"test-user\"}}"
        }
    ]
}
```

`entityIdKey` bisa saja menjadi sebuah `entityId` atau Anda dapat mengganti namanya dengan nama lain, misalnya `userId`. Dalam peristiwa yang sebenarnya, `entityId` dapat berupa nama pengguna, ID sesi, dan sebagainya.

```
"metricDefinition":{
    "name": "noFilter",
    "entityIdKey": "userDetails.userId", //should be consistent with jsonValue in
 events "data" fields
    "valueKey": "details.pageLoadTime"
},
```

Untuk memastikan bahwa peristiwa terkait dengan peluncuran atau percobaan yang benar, Anda harus memasukkan `entityId` yang sama ketika Anda memanggil `EvaluateFeature` dan `PutProjectEvents`. Pastikan untuk menelepon `PutProjectEvents` setelah `EvaluateFeature` panggilan, jika tidak data dijatuhkan dan tidak akan digunakan oleh CloudWatch Evidently.

Operasi `PutProjectEvents` tidak memerlukan nama fitur sebagai sebuah parameter input. Dengan cara ini, Anda akan dapat menggunakan satu peristiwa dalam beberapa percobaan. Sebagai contoh, Anda memanggil `EvaluateFeature` dengan `entityId` yang diatur ke `userDetails.userId`. Jika Anda mengeksekusi dua percobaan atau lebih, maka Anda dapat memiliki satu peristiwa dari metrik pemancar sesi yang dimiliki pengguna tersebut untuk masing-masing percobaan itu. Untuk melakukan hal ini, Anda harus memanggil `PutProjectEvents` sekali untuk masing-masing percobaan, dengan menggunakan `entityId` yang sama.

Pengaturan waktu

Setelah aplikasi Anda memanggil `EvaluateFeature`, akan ada periode waktu satu jam di mana peristiwa metrik dari `PutProjectEvents` dikaitkan berdasarkan evaluasi tersebut. Jika ada lagi peristiwa yang terjadi setelah periode satu jam itu, maka peristiwa tersebut tidak akan dikaitkan.

Namun demikian, jika `entityId` yang sama digunakan untuk memanggil panggilan `EvaluateFeature` baru selama jendela waktu satu jam panggilan awal itu, maka hasil

`EvaluateFeature` selanjutnya sekarang akan digunakan sebagai gantinya, dan pengatur waktu satu jam dimulai ulang. Hal ini hanya dapat terjadi dalam keadaan tertentu, seperti ketika lalu lintas percobaan diputar di antara dua tugas, sebagaimana yang dijelaskan dalam uraian yang ada di bagian Evaluasi lengket sebelumnya.

end-to-end Sebagai contoh, lihat[Tutorial: Pengujian A/B dengan aplikasi sampel Evidently](#).

Penyimpanan data proyek

Evidently mengumpulkan dua jenis peristiwa:

- Peristiwa evaluasi dikaitkan dengan variasi fitur mana yang ditetapkan ke sesi pengguna. Evidently menggunakan peristiwa-peristiwa ini untuk menghasilkan metrik dan data percobaan dan peluncuran lainnya, dan itu semua dapat Anda lihat di konsol Evidently.

  Anda juga dapat memilih untuk menyimpan peristiwa evaluasi ini di Amazon CloudWatch Log atau Amazon S3.
- Peristiwa kustom digunakan untuk menghasilkan metrik dari tindakan-tindakan pengguna seperti klik dan checkout. Evidently tidak menyediakan metode yang bisa Anda gunakan untuk menyimpan peristiwa kustom. Jika Anda ingin menyimpan peristiwa kustom, maka Anda harus mengubah kode aplikasi Anda agar Anda bisa mengirimnya ke opsi penyimpanan di luar Evidently.

Format log peristiwa evaluasi

Jika Anda memilih untuk menyimpan peristiwa evaluasi di CloudWatch Log atau Amazon S3, setiap peristiwa evaluasi disimpan sebagai peristiwa log dengan format berikut:

```
{
    "event_timestamp": 1642624900215,
    "event_type": "evaluation",
    "version": "1.0.0",
    "project_arn": "arn:aws:evidently:us-east-1:123456789012:project/petfood",
    "feature": "petfood-upsell-text",
    "variation": "Variation1",
    "entity_id": "7",
    "entity_attributes": {},
    "evaluation_type": "EXPERIMENT_RULE_MATCH",
    "treatment": "Variation1",
    "experiment": "petfood-experiment-2"
}
```

Berikut ini adalah detail lebih lanjut tentang format peristiwa evaluasi sebelumnya:

- Stempel waktu dalam waktu UNIX dengan milidetik
- Variasi adalah nama dari variasi fitur yang ditetapkan untuk sesi pengguna ini.
- Entitas ID adalah sebuah string.
- Atribut-atribut entitas adalah sebuah hash dari nilai arbitrer yang dikirim oleh klien. Sebagai contoh, jika `entityId` dipetakan ke biru atau hijau, maka Anda dapat secara opsional akan mengirim userID, data sesi, atau apa pun yang Anda inginkan dari perspektif korelasi dan gudang data.

Kebijakan IAM dan enkripsi IAM untuk penyimpanan peristiwa evaluasi di Amazon S3

Jika Anda memilih untuk menggunakan Amazon S3 untuk menyimpan peristiwa evaluasi, maka Anda harus menambahkan kebijakan IAM seperti berikut ini untuk memungkinkan Evidently menerbitkan log ke bucket Amazon S3. Hal ini karena bucket Amazon S3 dan objek yang ada di dalamnya bersifat privat, dan bucket Amazon S3 tidak akan mengizinkan akses ke layanan lain secara default.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AWSLogDeliveryWrite",
            "Effect": "Allow",
            "Principal": {"Service": "delivery.logs.amazonaws.com"},
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::bucket_name/optional_folder/AWSLogs/account_id/
*",
            "Condition": {"StringEquals": {"s3:x-amz-acl": "bucket-owner-full-
control"}}
        },
        {
            "Sid": "AWSLogDeliveryCheck",
            "Effect": "Allow",
            "Principal": {"Service": "delivery.logs.amazonaws.com"},
            "Action": ["s3:GetBucketAcl", "s3:ListBucket"],
            "Resource": "arn:aws:s3:::bucket_name"
        }
    ]
}
```

Jika Anda menyimpan data Evidently di Amazon S3, maka Anda juga dapat memilih untuk melakukan enkripsi dengan Server-Side Encryption dengan AWS Key Management Service Keys (SSE-KMS). Untuk informasi selengkapnya, silakan lihat Melindungi data menggunakan enkripsi sisi server.

Jika Anda menggunakan kunci yang dikelola pelanggan dari AWS KMS, Anda harus menambahkan berikut ini ke kebijakan IAM untuk kunci Anda. Hal ini akan memungkinkan untuk menulis ke bucket.

```
{
    "Sid": "AllowEvidentlyToUseCustomerManagedKey",
    "Effect": "Allow",
    "Principal": {
        "Service": [
            "delivery.logs.amazonaws.com"
        ]
    },
   "Action": [
       "kms:Encrypt",
       "kms:Decrypt",
       "kms:ReEncrypt*",
       "kms:GenerateDataKey*",
       "kms:DescribeKey"
    ],
    "Resource": "*"
}
```

Cara Evidently menghitung hasil

Anda dapat menggunakan pengujian Amazon CloudWatch Evidently A/B sebagai alat untuk pengambilan keputusan berbasis data. Dalam sebuah pengujian A/B, pengguna secara acak ditetapkan ke kelompok kontrol (juga disebut variasi default), atau salah satu kelompok perlakuan (juga disebut variasi yang diuji). Sebagai contoh, pengguna dalam grup kontrol mungkin akan mendapatkan pengalaman situs web, layanan, atau aplikasi dengan cara yang sama seperti yang mereka lakukan sebelum percobaan dimulai. Sementara itu, pengguna dalam kelompok perlakuan mungkin akan mengalami perubahan.

CloudWatch Terbukti mendukung hingga lima variasi berbeda dalam sebuah eksperimen. Evidently secara acak menetapkan lalu lintas ke variasi-variasi ini. Dengan cara ini, Anda akan dapat melacak metrik-metrik bisnis (seperti pendapatan) dan metrik-metrik performa (seperti latensi) untuk masing-masing grup. Evidently melakukan hal-hal berikut:

- Membandingkan perlakuan dengan kontrol. (Sebagai contoh, membandingkan apakah pendapatan naik atau turun dengan proses checkout baru.)
- Menunjukkan apakah perbedaan yang diamati antara perlakuan dan kontrol signifikan. Untuk ini, Evidently menawarkan dua pendekatan: tingkat signifikansi Frequentist dan probabilitas Bayesian.

Mengapa menggunakan pendekatan Frequentist dan Bayesian?

Pertimbangkan sebuah kasus di mana perlakuan tidak berpengaruh dibandingkan dengan kontrol, atau kasus di mana perlakuan identik dengan kontrol (pengujian A/A). Anda masih akan mengamati sebuah perbedaan kecil antara perlakuan dan kontrol dalam data. Hal ini karena peserta pengujian terdiri dari sampel pengguna yang terbatas, mewakili sebagian kecil dari semua pengguna situs web, layanan, atau aplikasi. Tingkat signifikansi Frequentist dan probabilitas Bayesian dapat memberikan wawasan tentang apakah perbedaan yang diamati itu signifikan atau hanya karena kebetulan semata.

Evidently mempertimbangkan hal-hal berikut untuk menentukan apakah perbedaan yang diamati bersifat signifikan atau tidak:

- Seberapa besar perbedaannya
- Berapa banyak sampel yang menjadi bagian dari pengujian
- Bagaimana data didistribusikan

Analisis Frequentist di Evidently

Evidently menggunakan pengujian berurutan, hal ini untuk mencegah masalah yang biasa terjadi, pengintipan, sebuah perangkap umum statistik frequentist. Pengintipan adalah praktik memeriksa hasil pengujian A/B yang sedang berlangsung untuk menghentikannya dan membuat keputusan berdasarkan hasil yang diamati. Untuk informasi selengkapnya tentang pengujian berurutan, silakan lihat Urutan kepercayaan time-uniform, nonparametrik, nonasimtotik oleh Howard dkk. (Ann. Statist. 49 (2) 1055 - 1080, 2021).

Karena hasil-hasil dari Evidently valid kapan saja (hasil yang valid kapan saja), Anda dapat mengintip hasil selama percobaan dan masih bisa menarik kesimpulan yang masuk akal. Hal ini akan dapat mengurangi beberapa biaya percobaan, karena Anda dapat menghentikan percobaan sebelum waktu yang dijadwalkan jika hasilnya sudah memiliki signifikansi.

Evidently menghasilkan tingkat signifikansi yang valid kapan saja dan interval kepercayaan 95% yang valid kapan saja dari perbedaan antara variasi yang diuji dan variasi default dalam metrik target.

Kolom Hasil dalam hasil percobaan tersebut menunjukkan performa variasi yang diuji, yang dapat berupa salah satu dari berikut:

- Tidak meyakinkan – Tingkat signifikansi kurang dari 95%
- Lebih baik – Tingkat signifikansi 95% atau lebih tinggi dan salah satu dari hal berikut ini benar:
  - Batas bawah interval kepercayaan 95% berada pada level lebih tinggi dari nol dan metriknya harus naik
  - Batas atas interval kepercayaan 95% berada pada level lebih rendah dari nol dan metriknya harus turun
- Lebih buruk – Tingkat signifikansi 95% atau lebih tinggi dan salah satu dari hal berikut ini benar:
  - Batas atas interval kepercayaan 95% berada pada level lebih tinggi dari nol dan metriknya harus naik
  - Batas bawah interval kepercayaan 95% berada pada level lebih rendah dari nol dan metriknya harus turun
- Terbaik – Percobaan memiliki dua variasi yang diuji atau lebih selain dari variasi default, dan kondisi–kondisi berikut terpenuhi:
  - Variasi memenuhi syarat untuk penunjukan yang Lebih baik
  - Salah satu dari hal berikut adalah benar:
    - Batas bawah interval kepercayaan 95% berada pada level yang lebih tinggi dari batas atas interval kepercayaan 95% dari semua variasi lainnya dan metriknya harus naik
    - Batas atas interval kepercayaan 95% berada pada level lebih rendah dari batas bawah interval kepercayaan 95% dari semua variasi lainnya dan metriknya harus turun

Analisis Bayesian dalam Evidently

Dengan analisis Bayesian, Anda akan dapat mengkalkulasi probabilitas bahwa rata-rata dalam variasi yang diuji lebih besar atau lebih kecil dari rata-rata dalam variasi default. Evidently melakukan inferensi Bayesian untuk rata-rata metrik target dengan menggunakan prior konjugat. Dengan prior konjugat, Evidently akan dapat menyimpulkan distribusi posterior yang diperlukan untuk analisis Bayesian dengan lebih efisien.

Evidently akan menunggu sampai tanggal akhir percobaan untuk menghitung hasil dari analisis Bayesian. Halaman hasil menampilkan hal berikut:

- probabilitas peningkatan – Probabilitas bahwa rata–rata metrik dalam variasi yang diuji setidaknya 3% lebih besar dari rata–rata dalam variasi default
- probabilitas penurunan – Probabilitas bahwa rata–rata metrik dalam variasi yang diuji setidaknya 3% lebih kecil dari rata–rata dalam variasi bawaan
- probabilitas tidak ada perubahan – Probabilitas bahwa rata–rata metrik dalam variasi yang diuji berada pada ±3% dari rata–rata dalam variasi bawaan

Kolom Hasil menunjukkan performa variasi, dan dapat berupa salah satu dari yang berikut:

- Lebih baik – Probabilitas kenaikan setidaknya 90% dan metriknya harus naik, atau probabilitas penurunan setidaknya 90% dan metriknya harus turun
- Lebih buruk – Probabilitas penurunan setidaknya 90% dan metriknya harus naik, atau probabilitas kenaikan setidaknya 90% dan metriknya harus turun

Menampilkan hasil peluncuran di dasbor

Anda dapat melihat kemajuan dan hasil metrik dari sebuah percobaan saat sedang berlangsung dan setelah selesai.

Cara melihat kemajuan dan hasil dari sebuah peluncuran

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Sinyal Aplikasi, Terbukti.
3. Pilih nama proyek yang berisi peluncuran tersebut.
4. Pilih tab Peluncuran.
5. Pilih nama peluncuran.
6. Untuk melihat langkah-langkah peluncuran dan alokasi lalu lintas untuk masing-masing langkah, silakan pilih tab Luncurkan.
7. Untuk melihat jumlah sesi pengguna yang ditetapkan untuk masing-masing variasi dari waktu ke waktu, dan untuk melihat metrik performa untuk setiap variasi dalam peluncuran tersebut, pilih tab Pemantauan.

   Tampilan ini juga akan menampilkan apakah ada alarm peluncuran yang beralih statusnya menjadi `ALARM` selama peluncuran.
8. Untuk melihat variasi, metrik, alarm, dan tag untuk peluncuran ini, silakan pilih tab Konfigurasi.

Menampilkan hasil percobaan di dasbor

Anda dapat melihat hasil statistik dari sebuah percobaan saat sedang berlangsung dan setelah selesai. Hasil percobaan akan tersedia untuk Anda hingga 63 hari setelah percobaan dimulai. Mereka tidak tersedia setelah itu karena kebijakan penyimpanan CloudWatch data.

Tidak ada hasil statistik yang akan ditampilkan sampai masing-masing variasi memiliki paling sedikit 100 peristiwa.

Evidently melakukan analisis nilai-p luar jaringan tambahan di akhir percobaan. Analisis nilai-p luar jaringan dapat mendeteksi signifikansi statistik dalam beberapa kasus di mana nilai-p kapan saja yang digunakan selama percobaan tidak menemukan signifikansi statistik.

Untuk informasi lebih lanjut tentang bagaimana CloudWatch Evidently menghitung hasil eksperimen, lihat. Cara Evidently menghitung hasil

Untuk melihat hasil dari sebuah percobaan

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Sinyal Aplikasi, Terbukti.
3. Pilih nama proyek yang berisi percobaan tersebut.
4. Pilih tab Percobaan.
5. Pilih nama percobaan, kemudian pilih tab Hasil.
6. Berdasarkan Performa variasi, ada kontrol di mana Anda dapat memilih statistik percobaan mana yang akan ditampilkan. Jika Anda memilih lebih dari satu statistik, maka Evidently akan menampilkan sebuah grafik dan tabel untuk masing-masing statistik.

   Masing-masing grafik dan tabel akan menampilkan hasil percobaan sejauh ini.

   Masing-masing grafik dapat menampilkan hasil-hasil sebagai berikut. Anda dapat menggunakan kontrol yang ada di sebelah kanan grafik untuk menentukan item mana dari item-item berikut yang akan ditampilkan:

   - Jumlah peristiwa sesi pengguna yang direkam untuk masing-masing variasi.
   - Nilai rata-rata metrik yang dipilih ada di bagian atas grafik, untuk masing-masing variasi.
   - Signifikansi statistik percobaan. Statistik ini membandingkan perbedaan untuk metrik yang dipilih yang ada di bagian atas grafik dengan variasi default dan masing-masing variasi lainnya.

- Batas kepercayaan atas dan bawah 95% pada perbedaan metrik yang dipilih, antara masing-masing variasi dan variasi default.

Tabel akan menampilkan sebuah baris untuk setiap variasi. Untuk masing-masing variasi yang bukan variasi default, Evidently akan menampilkan apakah ia telah menerima cukup data untuk menyatakan hasil yang signifikan secara statistik. Evidently juga akan menunjukkan apakah peningkatan variasi dalam nilai statistik telah mencapai tingkat kepercayaan 95%.

Terakhir, di kolom Hasil, Evidently akan memberikan rekomendasi tentang variasi mana yang memiliki performa terbaik berdasarkan statistik ini, atau apakah hasilnya tidak meyakinkan.

Bagaimana CloudWatch Terbukti Mengumpulkan dan Menyimpan Data

Amazon CloudWatch Terbukti mengumpulkan dan menyimpan data yang terkait dengan konfigurasi proyek sehingga pelanggan dapat menjalankan eksperimen dan peluncuran. Datanya meliputi hal-hal berikut:

- Metadata tentang proyek, fitur, peluncuran, dan percobaan
- Peristiwa metrik
- Data evaluasi

Metadata sumber daya disimpan di Amazon DynamoDB. Data dienkripsi saat istirahat secara default, menggunakan. Kunci milik AWS Kunci ini adalah kumpulan AWS KMS kunci yang Layanan AWS dimiliki dan dikelola untuk digunakan dalam beberapa Akun AWS. Pelanggan tidak dapat melihat, mengelola, atau mengaudit penggunaan kunci-kunci ini. Pelanggan juga tidak diharuskan untuk melakukan tindakan atau mengubah program untuk melindungi kunci-kunci yang mengenkripsi data mereka.

Untuk informasi selengkapnya, lihat Kunci milik AWSdi Panduan AWS Key Management Service Pengembang.

Peristiwa metrik dan peristiwa evaluasi Evidently dikirimkan langsung ke lokasi milik pelanggan.

Data bergerak secara otomatis dienkripsi dengan HTTPS. Data ini akan dikirimkan ke lokasi milik pelanggan.

Anda juga dapat memilih untuk menyimpan acara evaluasi di Amazon Simple Storage Service atau Amazon CloudWatch Logs. Untuk informasi selengkapnya tentang cara mengamankan data di

layanan ini, lihat Mengaktifkan enkripsi bucket default Amazon S3 dan Mengenkripsi data log di Log menggunakan. CloudWatch AWS KMS

Mengambil data

Anda dapat mengambil data Anda menggunakan CloudWatch Evidently API. Untuk mengambil data proyek, gunakan GetProjectatau ListProjects.

Untuk mengambil data fitur, gunakan GetFeatureatau ListFeatures.

Untuk mengambil data peluncuran, gunakan GetLaunchatau ListLaunches.

Untuk mengambil data eksperimen, gunakan, GetExperimentListExperiments, atau GetExperimentResults.

Mengubah dan menghapus data

Anda dapat memodifikasi dan menghapus data Anda menggunakan CloudWatch Evidently API. Untuk data proyek, gunakan UpdateProjectatau DeleteProject.

Untuk data fitur, gunakan UpdateFeatureatau DeleteFeature.

Untuk data peluncuran, gunakan UpdateLaunchatau DeleteLaunch.

Untuk data eksperimen, gunakan UpdateExperimentatau DeleteExperiment.

Menggunakan peran terkait layanan untuk Evidently

CloudWatch Evidently menggunakan peran terkait layanan (IAM) AWS Identity and Access Management. Peran yang terkait dengan layanan adalah tipe peran IAM unik yang terkait langsung ke Evidently. peran terkait layanan ditentukan sebelumnya oleh Evidently dan mencakup semua izin yang diperlukan layanan untuk memanggil layanan AWS lainnya atas nama Anda.

Peran terkait layanan membuat pengaturan Evidently lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. Evidently mendefinisikan izin peran yang terkait dengan layanannya, dan kecuali ditentukan lain, hanya Evidently yang dapat menjalankan perannya. Izin-izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, serta bahwa kebijakan izin tidak dapat dilampirkan ke entitas IAM lainnya.

Anda dapat menghapus peran tertaut layanan hanya setelah terlebih dahulu menghapus sumber dayanya yang terkait. Ini melindungi sumber daya Anda karena Anda tidak dapat secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran tertaut layanan, silakan lihat Layanan AWS yang Bisa Digunakan dengan IAM dan carilah layanan yang memiliki opsi Ya di kolom Peran tertaut layanan. Pilih Ya dengan sebuah tautan untuk melihat dokumentasi peran tertaut layanan untuk layanan tersebut.

Izin peran yang terhubung dengan layanan untuk Evidently

Evidently menggunakan peran terkait layanan bernama AWSServiceRoleForCloudWatchEvidently – Memungkinkan CloudWatch Evidently mengelola sumber daya AWS terkait atas nama pelanggan.

Peran tertaut layanan AWSServiceRoleForCloudWatchEvidently memercayakan layanan berikut untuk menjalankan peran tersebut:

- `CloudWatch Evidently`

Kebijakan izin peran bernama AmazonCloudWatchEvidentlyServiceRolePolicy memungkinkan untuk menyelesaikan tindakan berikut pada sumber daya yang ditentukan:

- Tindakan:`appconfig:StartDeployment`,`appconfig:StopDeployment`,`appconfig:ListDeploym` dan `appconfig:TagResource` pada klien tebal Evidently.

Anda harus mengonfigurasi izin untuk mengizinkan entitas IAM (seperti pengguna, grup, atau peran) untuk membuat, menyunting, atau menghapus peran terkait layanan. Untuk informasi selengkapnya, silakan lihat Izin peran tertaut layanan di Panduan Pengguna IAM.

Membuat peran terkait layanan untuk Evidently

Anda tidak perlu membuat peran terkait layanan secara manual. Saat Anda mulai menggunakan klien tebal Evidently diAWS Management Console, AWS CLI, atau API AWS, Evidently menciptakan peran terkait layanan untuk Anda.

Jika Anda menghapus peran tertaut layanan ini, dan ingin membuatnya lagi, Anda dapat mengulangi proses yang sama untuk membuat kembali peran tersebut di akun Anda. Saat Anda mulai menggunakan klien tebal Evidently, Evidently menciptakan peran tertaut layanan untuk Anda.

Menyunting peran terkait layanan untuk Evidently

Evidently tidak mengizinkan Anda untuk menyunting peran tertaut layanan AWSServiceRoleForCloudWatchEvidently. Setelah Anda membuat peran terkait layanan, Anda

tidak dapat mengubah nama peran karena berbagai entitas mungkin mereferensikan peran tersebut. Namun demikian, Anda dapat menyunting penjelasan peran menggunakan IAM. Untuk informasi selengkapnya, silakan lihat Menyunting peran tertaut layanan dalam Panduan Pengguna IAM.

Menghapus peran tertaut dengan layanan untuk Evidently

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, kami merekomendasikan agar Anda menghapus peran tersebut. Dengan begitu, Anda tidak memiliki entitas yang tidak digunakan yang tidak dipantau atau dipelihara secara aktif. Tetapi, Anda harus membersihkan sumber daya peran terkait layanan sebelum menghapusnya secara manual. Anda harus menghapus semua proyek Evidently yang menggunakan klien tebal.

> **Note**
>
> Jika layanan Evidently menggunakan peran saat Anda mencoba menghapus sumber daya, maka penghapusan tersebut mungkin gagal. Jika hal itu terjadi, tunggu beberapa menit dan coba mengoperasikannya lagi.

Untuk menghapus sumber daya Evidently yang digunakan oleh AWSServiceRoleForCloudWatchEvidently

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Pemantauan aplikasi, Evidently.
3. Dalam daftar proyek, centang kotak di samping proyek yang menggunakan klien tebal.
4. Pilih Tindakan proyek, Hapus proyek.

Untuk menghapus peran tertaut layanan secara manual menggunakan IAM

Gunakan konsol IAM,AWS CLI, atau API AWS untuk menghapus peran tertaut layanan AWSServiceRoleForCloudWatchEvidently. Untuk informasi selengkapnya, silakan lihat Menghapus peran tertaut layanan di Panduan Pengguna IAM.

Wilayah yang didukung untuk peran tertaut layanan Evidently

Evidently mendukung peran tertaut layanan di semua wilayah tempat layanan tersedia. Untuk informasi selengkapnya, silakan lihat Wilayah AWS dan titik akhir.

CloudWatch Terbukti kuota

CloudWatch Terbukti memiliki kuota berikut.

Sumber daya	Kuota bawaan
Proyek	50 per Wilayah, per akun. Anda dapat meminta penambahan kuota.
Segmen	500 per Wilayah, per akun. Anda dapat meminta penambahan kuota.
Kuota per proyek	• 100 fitur total • 500 peluncuran total • 50 peluncuran berjalan • 500 percobaan total • 50 percobaan berjalan Anda dapat meminta kenaikan kuota untuk semua kuota-kuota ini.
Kuota API (semua kuota per Wilayah)	• PutProjectEvents: 1000 transaksi per detik (TPS) di AS Timur (Virginia N.), AS Barat (Oregon), dan Eropa (Irlandia). 200 TPS di semua Wilayah lainnya. • EvaluateFeature: 1000 TPS di AS Timur (Virginia N.), AS Barat (Oregon), dan Eropa (Irlandia). 200 TPS di semua Wilayah lainnya. • BatchEvaluateFeature: 50 TPS • API Buat, Baca, Perbarui, Hapus (CRUD): 10 TPS digabungkan di semua API CRUD Anda dapat meminta kenaikan kuota untuk semua kuota-kuota ini.

Tutorial: Pengujian A/B dengan aplikasi sampel Evidently

Bagian ini menyediakan tutorial untuk menggunakan Amazon CloudWatch Terbukti untuk pengujian A/B. Tutorial ini adalah contoh aplikasi Evidently, yang merupakan sebuah aplikasi reaksi sederhana. Aplikasi sampel akan dikonfigurasi untuk menampilkan fitur `showDiscount` atau tidak. Ketika fitur tersebut ditampilkan untuk seorang pengguna, harga yang ditampilkan di situs belanja itu ditampilkan dengan diskon 20%.

Selain menunjukkan diskon kepada beberapa pengguna dan bukan kepada orang lain, dalam tutorial ini Anda juga mengatur Evidently untuk mengumpulkan metrik waktu buka halaman dari kedua variasi.

> **Warning**
>
> Skenario ini mengharuskan pengguna IAM dengan akses terprogram dan kredensil jangka panjang, yang menghadirkan risiko keamanan. Untuk membantu mengurangi risiko ini, kami menyarankan agar Anda memberikan pengguna ini hanya izin yang mereka perlukan untuk melakukan tugas dan menghapus pengguna ini ketika mereka tidak lagi diperlukan. Kunci akses dapat diperbarui jika perlu. Untuk informasi selengkapnya, lihat Memperbarui kunci akses di Panduan Pengguna IAM.

Langkah 1: Mengunduh aplikasi sampel

Mulailah dengan mengunduh aplikasi sampel Evidently.

Cara mengunduh aplikasi sampel

1. Unduh aplikasi sampel dari bucket Amazon S3 berikut:

   ```
   https://evidently-sample-application.s3.us-west-2.amazonaws.com/evidently-sample-shopping-app.zip
   ```

2. Buka zip paketnya.

Langkah 2: Menambahkan titik akhir Evidently dan menyiapkan kredensial

Berikutnya, tambahkan Wilayah dan titik akhir untuk Evidently ke file `config.js` di direktori `src` dalam paket aplikasi sampel, seperti pada contoh berikut:

```
evidently: {
    REGION: "us-west-2",
    ENDPOINT: "https://evidently.us-west-2.amazonaws.com (https://evidently.us-
west-2.amazonaws.com/)",
},
```

Anda juga harus memastikan bahwa aplikasi memiliki izin untuk menelepon CloudWatch dengan jelas.

Cara memberikan izin aplikasi sampel untuk memanggil Evidently

1. Federasi ke AWS akun Anda.
2. Buat pengguna IAM dan lampirkan AmazonCloudWatchEvidentlyFullAccesskebijakan ke pengguna ini.
3. Catat ID kunci akses dan kunci akses rahasia dari pengguna IAM tersebut, karena Anda akan membutuhkannya di langkah berikutnya.
4. Dalam file `config.js` yang sama yang sudah Anda ubah sebelumnya di bagian ini, masukkan nilai ID kunci akses dan kunci akses rahasia, seperti pada contoh berikut:

   ```
   credential: {
      accessKeyId: "Access key ID",
      secretAccessKey: "Secret key"
   }
   ```

   > **Important**
   >
   > Kami menggunakan langkah ini untuk membuat aplikasi sampel tersebut sesederhana mungkin untuk Anda coba. Sebaiknya Anda tidak memasukkan kredensial pengguna IAM Anda ke dalam aplikasi produksi Anda yang sebenarnya. Sebaliknya, kami menyarankan Anda menggunakan Amazon Cognito untuk autentikasi. Untuk informasi selengkapnya, silakan lihat Mengintegrasikan Amazon Cognito dengan aplikasi web dan seluler.

Langkah 3: Menyiapkan kode untuk evaluasi fitur

Bila Anda menggunakan CloudWatch Evidently untuk mengevaluasi fitur, Anda harus menggunakan EvaluateFeatureoperasi untuk secara acak memilih variasi fitur untuk setiap sesi pengguna. Operasi

ini menetapkan sesi pengguna untuk masing-masing variasi fitur, sesuai dengan persentase yang Anda tentukan dalam percobaan.

Cara menyiapkan kode evaluasi fitur untuk aplikasi demo toko buku

1. Tambahkan pembuat klien dalam file `src/App.jsx` sehingga aplikasi sampel dapat memanggil Evidently.

```
import Evidently from 'aws-sdk/clients/evidently';
import config from './config';

const defaultClientBuilder = (
    endpoint,
    region,
) => {
    const credentials = {
        accessKeyId: config.credential.accessKeyId,
        secretAccessKey: config.credential.secretAccessKey
    }
    return new Evidently({
        endpoint,
        region,
        credentials,
    });
};
```

2. Tambahkan berikut ini ke bagian kode `const App` untuk memulai klien.

```
  if (client == null) {
      client = defaultClientBuilder(
        config.evidently.ENDPOINT,
        config.evidently.REGION,
      );
```

3. Membuat konsep `evaluateFeatureRequest` dengan menambahkan kode berikut. Kode ini akan secara otomatis mengisi nama proyek dan nama fitur yang kami rekomendasikan nanti dalam tutorial ini. Anda dapat mengganti nama proyek dan fitur Anda sendiri, selama Anda juga menentukan nama proyek dan fitur tersebut di konsol Evidently.

```
const evaluateFeatureRequest = {
              entityId: id,
              // Input Your feature name
```

```
              feature: 'showDiscount',
              // Input Your project name'
              project: 'EvidentlySampleApp',
            };
```

4. Tambahkan kode untuk memanggil Evidently untuk melakukan evaluasi fitur. Ketika permintaan tersebut dikirim, Evidently akan secara acak menetapkan sesi pengguna untuk melihat fitur `showDiscount` atau tidak.

```
client.evaluateFeature(evaluateFeatureRequest).promise().then(res => {
    if(res.value?.boolValue !== undefined) {
        setShowDiscount(res.value.boolValue);
    }
    getPageLoadTime()
 })
```

Langkah 4: Menyiapkan kode untuk metrik percobaan

Untuk metrik kustom, gunakan API `PutProjectEvents` Evidently untuk mengirim hasil metrik ke Evidently. Contoh-contoh berikut akan menunjukkan kepada Anda cara menyiapkan metrik kustom dan mengirim data percobaan ke Evidently.

Tambahkan fungsi berikut untuk menghitung waktu buka halaman dan gunakan `PutProjectEvents` untuk mengirimkan nilai metrik ke Evidently. Tambahkan fungsi berikut ke dalam `Home.tsx` dan memanggil fungsi ini di dalam API `EvaluateFeature`:

```
const getPageLoadTime = () => {
    const timeSpent = (new Date().getTime() - startTime.getTime()) * 1.000001;
    const pageLoadTimeData = `{
      "details": {
        "pageLoadTime": ${timeSpent}
      },
      "UserDetails": { "userId": "${id}", "sessionId": "${id}"}
    }`;
    const putProjectEventsRequest = {
      project: 'EvidentlySampleApp',
      events: [
        {
          timestamp: new Date(),
          type: 'aws.evidently.custom',
          data: JSON.parse(pageLoadTimeData)
```

```
        },
      ],
    };
    client.putProjectEvents(putProjectEventsRequest).promise();
  }
```

Berikut ini adalah tampilan file `App.js` setelah semua pengeditan yang telah Anda lakukan sejak Anda mengunduhnya.

```
import React, { useEffect, useState } from "react";
import { BrowserRouter as Router, Switch } from "react-router-dom";
import AuthProvider from "contexts/auth";
import CommonProvider from "contexts/common";
import ProductsProvider from "contexts/products";
import CartProvider from "contexts/cart";
import CheckoutProvider from "contexts/checkout";
import RouteWrapper from "layouts/RouteWrapper";
import AuthLayout from "layouts/AuthLayout";
import CommonLayout from "layouts/CommonLayout";
import AuthPage from "pages/auth";
import HomePage from "pages/home";
import CheckoutPage from "pages/checkout";
import "assets/scss/style.scss";
import { Spinner } from 'react-bootstrap';

import Evidently from 'aws-sdk/clients/evidently';
import config from './config';

const defaultClientBuilder = (
  endpoint,
  region,
) => {
  const credentials = {
    accessKeyId: config.credential.accessKeyId,
    secretAccessKey: config.credential.secretAccessKey
  }
  return new Evidently({
    endpoint,
    region,
    credentials,
  });
};
```

```
const App = () => {
  const [isLoading, setIsLoading] = useState(true);
  const [startTime, setStartTime] = useState(new Date());
  const [showDiscount, setShowDiscount] = useState(false);
  let client = null;
  let id = null;

  useEffect(() => {
    id = new Date().getTime().toString();
    setStartTime(new Date());
    if (client == null) {
      client = defaultClientBuilder(
        config.evidently.ENDPOINT,
        config.evidently.REGION,
      );
    }
    const evaluateFeatureRequest = {
      entityId: id,
      // Input Your feature name
      feature: 'showDiscount',
      // Input Your project name'
      project: 'EvidentlySampleApp',
    };

    // Launch
    client.evaluateFeature(evaluateFeatureRequest).promise().then(res => {
      if(res.value?.boolValue !== undefined) {
        setShowDiscount(res.value.boolValue);
      }
    });

    // Experiment
    client.evaluateFeature(evaluateFeatureRequest).promise().then(res => {
      if(res.value?.boolValue !== undefined) {
        setShowDiscount(res.value.boolValue);
      }
      getPageLoadTime()
    })

    setIsLoading(false);
  },[]);

  const getPageLoadTime = () => {
    const timeSpent = (new Date().getTime() - startTime.getTime()) * 1.000001;
```

```
    const pageLoadTimeData = `{
      "details": {
        "pageLoadTime": ${timeSpent}
      },
      "UserDetails": { "userId": "${id}", "sessionId": "${id}"}
    }`;
    const putProjectEventsRequest = {
      project: 'EvidentlySampleApp',
      events: [
        {
          timestamp: new Date(),
          type: 'aws.evidently.custom',
          data: JSON.parse(pageLoadTimeData)
        },
      ],
    };
    client.putProjectEvents(putProjectEventsRequest).promise();
  }
  return (
    !isLoading? (
    <AuthProvider>
      <CommonProvider>
        <ProductsProvider>
          <CartProvider>
            <CheckoutProvider>
              <Router>
                <Switch>
                  <RouteWrapper
                    path="/"
                    exact
                    component={() => <HomePage showDiscount={showDiscount}/>}
                    layout={CommonLayout}
                  />
                  <RouteWrapper
                    path="/checkout"
                    component={CheckoutPage}
                    layout={CommonLayout}
                  />
                  <RouteWrapper
                    path="/auth"
                    component={AuthPage}
                    layout={AuthLayout}
                  />
                </Switch>
```

```
              </Router>
            </CheckoutProvider>
          </CartProvider>
        </ProductsProvider>
      </CommonProvider>
    </AuthProvider> ) : (
      <Spinner animation="border" />
    )
  );
};

export default App;
```

Setiap kali ada seorang pengguna mengunjungi aplikasi sampel, metrik kustom akan dikirim ke Evidently untuk analisis. Evidently menganalisis setiap metrik dan menampilkan hasil secara waktu nyata di dasbor Evidently. Contoh berikut menunjukkan sebuah muatan metrik:

```
[ {"timestamp": 1637368646.468, "type": "aws.evidently.custom", "data": "{\"details
\":{\"pageLoadTime\":2058.002058},\"userDetails\":{\"userId\":\"1637368644430\",
\"sessionId\":\"1637368644430\"}}" } ]
```

Langkah 5: Membuat proyek, fitur, dan percobaan

Selanjutnya, Anda membuat proyek, fitur, dan eksperimen di konsol CloudWatch Evidently.

Untuk membuat proyek, fitur, dan percobaan untuk tutorial ini

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Sinyal Aplikasi, Terbukti.
3. Pilih Buat proyek dan isi bidang-bidang yang ditampilkan. Anda harus menggunakan **`EvidentlySampleApp`** untuk nama proyek agar sampel dapat berfungsi dengan benar. Untuk Penyimpanan peristiwa evaluasi, pilih Jangan simpan peristiwa evaluasi.

   Setelah mengisi bidang-bidang itu, pilih Buat Proyek.

   Untuk detail selengkapnya, silakan lihat Membuat sebuah proyek baru.
4. Setelah proyek selesai dibuat, buatlah sebuah fitur dalam proyek itu. Beri nama fitur dengan **`showDiscount`**. Dalam fitur ini, buatlah dua variasi tipe **`Boolean`**. Beri nama variasi pertama dengan nama **`disable`** dengan nilai **`False`** dan beri nama variasi kedua dengan nama **`enable`** dengan nilai**`True`**.

Untuk informasi selengkapnya tentang cara membuat sebuah fitur, silakan lihat Menambahkan sebuah fitur ke proyek.

5. Setelah Anda selesai membuat fitur tersebut, buatlah sebuah percobaan dalam proyek. Beri nama percobaan tersebut dengan nama **`pageLoadTime`**.

   Percobaan ini akan menggunakan metrik kustom yang bernama `pageLoadTime` yang mengukur waktu buka halaman dari halaman yang sedang diuji. Metrik khusus untuk eksperimen dibuat menggunakan Amazon EventBridge. Untuk informasi selengkapnya EventBridge, lihat Apa itu Amazon EventBridge? .

   Untuk membuat metrik kustom tersebut, lakukan hal-hal berikut saat Anda membuat percobaan:

   - Pada Metrik, untuk Sumber metrik, pilih Metrik kustom.
   - Untuk Nama metrik, masukkan **`pageLoadTime`**.
   - Untuk Tujuan pilih Turun. Hal ini menunjukkan bahwa kita menginginkan nilai yang lebih rendah dari metrik ini untuk menunjukkan variasi terbaik dari fitur tersebut.
   - Untuk Aturan metrik, masukkan berikut ini:
     - Untuk ID Entitas, masukkan **`UserDetails.userId`**.
     - Untuk Kunci nilai, masukkan **`details.pageLoadTime`**.
     - Untuk Unit, masukkan **`ms`**.
   - Pilih Tambahkan metrik.

   Untuk Audiens, pilih 100% sehingga semua pengguna akan dimasukkan dalam percobaan. Setel pembagian lalu lintas antara variasi menjadi 50% masing-masing.

   Kemudian, pilih Buat percobaan untuk membuat percobaan tersebut. Setelah Anda selesai membuatnya, percobaan itu tidak akan dimulai sampai Anda memberi tahu Evidently untuk memulainya.

Langkah 6: Mulai percobaan dan uji dengan CloudWatch jelas

Langkah terakhirnya adalah memulai percobaan dan memulai aplikasi sampel.

Untuk memulai percobaan dalam tutorial ini

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.

2. Di panel navigasi, pilih Sinyal Aplikasi, Terbukti.
3. Pilih EvidentlySampleAppproyek.
4. Pilih tab Percobaan.
5. Pilih tombol di samping pageLoadTimedan pilih Tindakan, Mulai eksperimen.
6. Pilih waktu kapan percobaan akan berakhir.
7. Pilih Mulai percobaan.

   Percobaan akan dimulai saat itu juga.

Berikutnya, Anda harus memulai aplikasi sampel Evidently dengan perintah berikut:

```
npm install -f && npm start
```

Setelah aplikasi sampel tersebut dimulai, Anda akan ditugaskan ke salah satu dari dua variasi fitur yang sedang diuji. Satu variasi akan menampilkan "Diskon 20%" dan variasi lainnya tidak. Terus segarkan halaman untuk melihat variasi-variasi yang berbeda.

> **Note**
>
> Evidently memiliki evaluasi yang lengket. Evaluasi fitur bersifat deterministik, artinya untuk `entityId` dan fitur yang sama, seorang pengguna akan selalu menerima penetapan variasi yang sama. Satu-satunya waktu perubahan penetapan variasi adalah ketika ada entitas yang ditambahkan untuk sebuah penggantian atau lalu lintas percobaan dipanggil.
> Namun demikian, untuk mempermudah penggunaan tutorial aplikasi sampel ini untuk Anda, Evidently menetapkan kembali evaluasi fitur aplikasi sampel setiap kali Anda menyegarkan halaman, sehingga Anda dapat mengalami kedua variasi tersebut tanpa harus menambahkan penggantian.

Pemecahan Masalah

Kami menyarankan Anda untuk menggunakan `npm` versi 6.14.14. Jika Anda melihat adanya kesalahan tentang cara membangun atau memulai aplikasi sampel dan Anda menggunakan versi npm yang berbeda, lakukan hal berikut.

Cara melakukan instalasi **npm** versi 6.14.14

1. Gunakan browser untuk terhubung ke https://nodejs.org/download/release/v14.17.5/.
2. Unduh node-v14.17.5.pkg dan jalankan pkg ini untuk melakukan instalasi npm.

   Jika Anda melihat sebuah kesalahan `webpack not found`, buka folder `evidently-sample-shopping-app` dan coba lakukan hal berikut ini:

   a. Hapus `package-lock.json`

   b. Hapus `yarn-lock.json`

   c. Hapus `node_modules`

   d. Hapus dependensi webpack dari `package.json`

   e. Jalankan hal berikut:

   ```
   npm install -f && npm
   ```

Gunakan CloudWatch RUM

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. Data yang dapat Anda visualisasikan dan analisis mencakup waktu pemuatan halaman, kesalahan sisi klien, dan perilaku pengguna. Ketika Anda melihat data ini, Anda dapat melihat semuanya digabungkan bersama-sama dan juga melihat detail berdasarkan browser dan perangkat yang digunakan pelanggan Anda.

Anda dapat menggunakan data yang dikumpulkan untuk mengidentifikasi dan men-debug masalah kinerja sisi klien dengan cepat. CloudWatch RUM membantu Anda memvisualisasikan anomali dalam kinerja aplikasi Anda dan menemukan data debugging yang relevan seperti pesan kesalahan, jejak tumpukan, dan sesi pengguna. Anda juga dapat menggunakan RUM untuk memahami rentang dampak pengguna akhir termasuk jumlah pengguna, geolokasi, dan browser yang digunakan.

Data pengguna akhir yang Anda kumpulkan untuk CloudWatch RUM disimpan selama 30 hari dan kemudian dihapus secara otomatis. Jika Anda ingin menyimpan acara RUM untuk waktu yang lebih lama, Anda dapat memilih agar monitor aplikasi mengirim salinan peristiwa ke CloudWatch Log di akun Anda. Kemudian, Anda dapat menyesuaikan periode retensi untuk grup log tersebut.

Untuk menggunakan RUM, Anda membuat monitor aplikasi dan memberikan beberapa informasi. RUM menghasilkan JavaScript cuplikan untuk Anda tempel ke aplikasi Anda. Potongan ini menarik kode klien web RUM. Klien web RUM menangkap data dari persentase sesi pengguna aplikasi Anda, yang ditampilkan dalam dasbor yang sudah dibuat sebelumnya. Anda dapat menentukan berapa persentase sesi pengguna untuk mengumpulkan data.

CloudWatch RUM terintegrasi dengan Application Signals, yang dapat menemukan dan memantau layanan aplikasi, klien, kenari Synthetics, dan dependensi layanan Anda. Gunakan Sinyal Aplikasi untuk melihat daftar atau peta visual layanan Anda, melihat metrik kesehatan berdasarkan tujuan tingkat layanan (SLO) Anda, dan menelusuri lebih dalam untuk melihat jejak X-Ray yang berkorelasi untuk pemecahan masalah yang lebih rinci. Untuk melihat permintaan halaman klien RUM di Sinyal Aplikasi, aktifkan pelacakan aktif X-Ray dengan membuat monitor aplikasi, atau mengonfigurasi klien web RUM secara manual. Klien RUM Anda ditampilkan pada Peta Layanan yang terhubung ke layanan Anda, dan di halaman Detail layanan dari layanan yang mereka panggil.

Klien web RUM adalah sumber terbuka. Untuk informasi lebih lanjut, lihat klien web CloudWatch RUM.

Pertimbangan performa

Bagian ini membahas pertimbangan kinerja menggunakan CloudWatch RUM.

- Memuat dampak kinerja — Klien web CloudWatch RUM dapat diinstal dalam aplikasi web Anda sebagai JavaScript modul, atau dimuat ke dalam aplikasi web Anda secara asinkron dari jaringan pengiriman konten (CDN). Itu tidak memblokir proses pemuatan aplikasi. CloudWatch RUM dirancang agar tidak ada dampak yang nyata terhadap waktu buka aplikasi.
- Dampak runtime — Klien web RUM melakukan pemrosesan untuk merekam dan mengirimkan data RUM ke layanan RUM. CloudWatch Karena kejadian jarang terjadi dan jumlah pemrosesannya kecil, CloudWatch RUM dirancang agar tidak ada dampak yang dapat dideteksi terhadap kinerja aplikasi.
- Dampak jaringan — Klien web RUM secara berkala mengirimkan data ke layanan CloudWatch RUM. Data dikirim secara berkala saat aplikasi berjalan, dan juga segera sebelum browser membongkar aplikasi. Data yang dikirim segera sebelum browser membongkar aplikasi dikirim sebagai suar, yang dirancang agar tidak memiliki dampak yang dapat dideteksi pada waktu pembongkaran aplikasi.

Harga RUM

Dengan CloudWatch RUM, Anda dikenakan biaya untuk setiap acara RUM yang diterima CloudWatch RUM. Setiap item data yang dikumpulkan menggunakan klien web RUM dianggap sebagai peristiwa RUM. Contoh peristiwa RUM termasuk tampilan halaman, JavaScript kesalahan, dan kesalahan HTTP. Anda memiliki opsi untuk jenis peristiwa yang dikumpulkan oleh setiap monitor aplikasi. Anda dapat mengaktifkan atau menonaktifkan opsi untuk mengumpulkan peristiwa telemetri kinerja, JavaScript kesalahan, kesalahan HTTP, dan jejak X-Ray. Untuk informasi selengkapnya tentang memilih opsi ini, silakan lihat Langkah 2: Membuat monitor aplikasi dan Informasi yang dikumpulkan oleh klien web CloudWatch RUM. Untuk informasi selengkapnya tentang harga, lihat CloudWatchHarga Amazon.

Ketersediaan Wilayah

CloudWatch RUM saat ini tersedia di Wilayah berikut:

- AS Timur (N. Virginia)
- AS Timur (Ohio)
- AS Barat (California Utara)
- AS Barat (Oregon)
- Afrika (Cape Town)
- Asia Pasifik (Jakarta)
- 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)

Topik

Kebijakan IAM untuk menggunakan CloudWatch RUM

Untuk dapat mengelola CloudWatch RUM sepenuhnya, Anda harus masuk sebagai pengguna IAM atau peran yang memiliki kebijakan AmazonCloudWatchRUM FullAccess IAM. Selain itu, Anda mungkin memerlukan kebijakan atau izin lain:

- Untuk membuat monitor aplikasi yang membuat kumpulan identitas Amazon Cognito baru untuk otorisasi, Anda harus memiliki peran Admin IAM atau kebijakan IAM. AdministratorAccess
- Untuk membuat monitor aplikasi yang mengirimkan data ke CloudWatch Log, Anda harus login ke peran atau kebijakan IAM yang memiliki izin berikut:

```
{
    "Effect": "Allow",
    "Action": [
        "logs:PutResourcePolicy"
```

```
    ],
    "Resource": [
        "*"
    ]
}
```

Pengguna lain yang perlu melihat data CloudWatch RUM tetapi tidak perlu membuat sumber daya CloudWatch RUM, dapat diberikan ReadOnlyAccess kebijakan AmazonCloudWatchRUM.

Siapkan aplikasi untuk menggunakan CloudWatch RUM

Gunakan langkah-langkah di bagian ini untuk mengatur aplikasi Anda agar mulai menggunakan CloudWatch RUM untuk mengumpulkan data kinerja dari sesi pengguna nyata.

Topik

Langkah 1: Otorisasi aplikasi Anda untuk mengirim data AWS

Untuk menggunakan CloudWatch RUM, aplikasi Anda harus memiliki otorisasi.

Anda memiliki tiga opsi untuk mengatur otorisasi:

- Biarkan CloudWatch RUM membuat kumpulan identitas Amazon Cognito baru untuk aplikasi. Metode ini membutuhkan usaha paling sedikit untuk mengatur. Ini adalah opsi default.

  Kolam identitas akan berisi identitas yang tidak diautentikasi. Hal ini memungkinkan klien web CloudWatch RUM untuk mengirim data ke CloudWatch RUM tanpa mengautentikasi pengguna aplikasi.

  Kolam identitas Amazon Cognito memiliki peran IAM terlampir. Identitas Amazon Cognito yang tidak diautentikasi memungkinkan klien web untuk mengambil peran IAM yang berwenang untuk mengirim data ke RUM. CloudWatch

- Gunakan kolam identitas Amazon Cognito yang ada. Dalam hal ini, Anda juga harus memodifikasi peran IAM yang dilampirkan ke kolam identitas.
- Gunakan autentikasi dari penyedia identitas yang sudah ada yang telah Anda atur. Dalam hal ini, Anda harus mendapatkan kredensial dari penyedia identitas dan aplikasi Anda harus meneruskan kredensial ini ke klien web RUM.

Bagian berikut mencakup detail lebih lanjut tentang opsi ini.

CloudWatch RUM membuat kumpulan identitas Amazon Cognito baru

Ini adalah opsi paling sederhana untuk diatur, dan jika Anda memilih ini, tidak diperlukan langkah pengaturan lebih lanjut. Anda harus memiliki izin administratif untuk menggunakan opsi ini. Untuk informasi selengkapnya, lihat Kebijakan IAM untuk menggunakan CloudWatch RUM.

Dengan opsi ini, CloudWatch RUM menciptakan sumber daya berikut:

- Sebuah kolam identitas Amazon Cognito baru
- Identitas Amazon Cognito yang tidak diautentikasi. Ini memungkinkan klien web RUM mengambil peran IAM tanpa mengautentikasi pengguna aplikasi.
- Peran IAM yang akan diasumsikan klien web RUM. Kebijakan IAM yang dilampirkan pada peran ini memungkinkannya menggunakan API `PutRumEvents` dengan sumber daya monitor aplikasi. Dengan kata lain, ini memungkinkan klien web RUM mengirim data ke RUM.

Klien web RUM menggunakan identitas Amazon Cognito untuk mendapatkan AWS kredensi. AWS Kredensialnya terkait dengan peran IAM. Peran IAM diizinkan untuk digunakan `PutRumEvents` dengan AppMonitor sumber daya.

Amazon Cognito mengirimkan token keamanan yang diperlukan untuk memungkinkan aplikasi Anda mengirim data ke CloudWatch RUM. Cuplikan JavaScript kode yang dihasilkan CloudWatch RUM mencakup baris berikut untuk mengaktifkan otentikasi.

```
   {
    identityPoolId: [identity pool id],  // e.g., 'us-west-2:EXAMPLE4a-66f6-4114-902a-
EXAMPLEbad7'
    guestRoleArn: [iam role arn]         // e.g., 'arn:aws:iam::123456789012:role/
Nexus-Monitor-us-east-1-123456789012_Unauth_5889316876161'
   }
```

```
);
```

Gunakan kolam identitas Amazon Cognito yang ada

Jika Anda memilih untuk menggunakan kumpulan identitas Amazon Cognito yang ada, Anda menentukan kumpulan identitas saat menambahkan aplikasi ke CloudWatch RUM. Kolam harus mendukung pengaktifan ke identitas yang tidak diautentikasi. Anda juga harus menambahkan izin berikut ke kebijakan IAM yang dilampirkan ke peran IAM yang terkait dengan kolam identitas ini.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "rum:PutRumEvents",
            "Resource": "arn:aws:rum:[region]:[accountid]:appmonitor/[app monitor
 name]"
        }
    ]
}
```

Amazon Cognito kemudian akan mengirim token keamanan yang diperlukan untuk memungkinkan aplikasi Anda mengakses CloudWatch RUM.

Penyedia pihak ketiga

Jika memilih untuk menggunakan autentikasi privat dari penyedia pihak ketiga, Anda harus mendapatkan kredensial dari penyedia identitas dan meneruskannya ke AWS. Cara terbaik untuk melakukannya adalah dengan menggunakan vendor token keamanan. Anda dapat menggunakan vendor token keamanan apa pun, termasuk Amazon Cognito dengan. AWS Security Token Service Untuk informasi selengkapnya AWS STS, lihat Selamat Datang di Referensi AWS Security Token Service API.

Jika Anda ingin menggunakan Amazon Cognito sebagai vendor token dalam skenario ini, Anda dapat mengonfigurasi Amazon Cognito agar berfungsi dengan penyedia autentikasi. Untuk informasi selengkapnya, silakan lihat Memulai Kolam Identitas Amazon Cognito (Gabungan Identitas).

Setelah mengonfigurasi Amazon Cognito untuk bekerja dengan penyedia identitas Anda, Anda juga perlu melakukan hal berikut ini:

- Buat peran IAM dengan izin sebagai berikut. Aplikasi Anda akan menggunakan peran ini untuk mengakses AWS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "rum:PutRumEvents",
            "Resource": "arn:aws:rum:[region]:[accountID]:appmonitor/[app monitor
 name]"
        }
    ]
}
```

- Tambahkan yang berikut ini ke aplikasi Anda agar dapat meneruskan kredensyal dari penyedia Anda ke CloudWatch RUM. Masukkan baris sehingga berjalan setelah pengguna masuk ke aplikasi Anda dan aplikasi telah menerima kredensial yang akan digunakan untuk mengakses AWS.

```
cwr('setAwsCredentials', {/* Credentials or CredentialProvider */});
```

Untuk informasi selengkapnya tentang penyedia kredensi di AWS JavaScript SDK, lihat Menyetel kredensional di browser web di panduan pengembang v3 untuk SDK for JavaScript, Menyetel kredensional di browser web di panduan pengembang v2 untuk SDK for, dan @aws -sdk/credential-providers. JavaScript

Anda juga dapat menggunakan SDK untuk klien web CloudWatch RUM untuk mengkonfigurasi metode otentikasi klien web. Untuk informasi selengkapnya tentang SDK klien web, lihat SDK klien web CloudWatch RUM.

Langkah 2: Membuat monitor aplikasi

Untuk mulai menggunakan CloudWatch RUM dengan aplikasi Anda, Anda membuat monitor aplikasi. Saat monitor aplikasi dibuat, RUM menghasilkan JavaScript cuplikan untuk Anda tempelkan ke aplikasi Anda. Potongan ini menarik kode klien web RUM. Klien web RUM menangkap data dari persentase sesi pengguna aplikasi Anda dan mengirimkannya ke RUM.

Buat monitor aplikasi

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Sinyal Aplikasi, RUM.
3. Pilih Tambahkan monitor aplikasi.
4. Masukkan informasi dan pengaturan untuk aplikasi Anda:

   - Untuk nama monitor Aplikasi, masukkan nama yang akan digunakan untuk mengidentifikasi monitor aplikasi ini dalam konsol CloudWatch RUM.
   - Untuk Domain aplikasi, masukkan nama domain tingkat atas tempat aplikasi Anda memiliki otoritas administratif. Ini harus dalam format domain URL.

     Pilih Sertakan sub domain agar monitor aplikasi juga mengumpulkan data dari semua subdomain di bawah domain tingkat atas.
5. Untuk Mengonfigurasi pengumpulan data RUM, tentukan apakah Anda ingin monitor aplikasi mengumpulkan masing-masing hal berikut:

   - Telemetri performa – Mengumpulkan informasi tentang pemuatan halaman dan waktu muat sumber daya
   - JavaScript kesalahan — Mengumpulkan informasi tentang kesalahan yang tidak tertangani yang ditimbulkan oleh JavaScript aplikasi Anda
   - Kesalahan HTTP – Mengumpulkan informasi tentang kesalahan HTTP yang dilemparkan oleh aplikasi Anda

   Memilih opsi ini memberikan informasi lebih lanjut tentang aplikasi Anda, tetapi juga menghasilkan lebih banyak acara CloudWatch RUM dan dengan demikian menimbulkan lebih banyak biaya.

   Jika Anda tidak memilih salah satu dari ini, monitor aplikasi masih mengumpulkan peristiwa awal sesi dan ID halaman sehingga Anda dapat melihat berapa banyak pengguna yang menggunakan aplikasi Anda, termasuk kerusakan berdasarkan jenis dan versi sistem operasi, jenis dan versi browser, jenis perangkat, dan lokasi.
6. Pilih Periksa opsi ini untuk mengizinkan Klien Web CloudWatch RUM mengatur cookie jika Anda ingin dapat mengumpulkan ID pengguna dan ID sesi dari sesi pengguna sampel. ID pengguna dihasilkan secara acak oleh RUM. Untuk informasi selengkapnya, lihat CloudWatch Cookie klien web RUM (atau teknologi serupa).

7. Untuk Sampel sesi, masukkan persentase sesi pengguna yang akan digunakan untuk mengumpulkan data RUM. Bawaannya adalah 100%. Mengurangi angka ini memberi Anda lebih sedikit data, tetapi mengurangi biaya Anda. Untuk informasi selengkapnya tentang harga RUM, silakan lihat Harga RUM.
8. Data pengguna akhir yang Anda kumpulkan untuk CloudWatch RUM disimpan selama 30 hari dan kemudian dihapus. Jika Anda ingin menyimpan salinan peristiwa RUM di CloudWatch Log dan mengonfigurasi berapa lama untuk menyimpan salinan ini, pilih Periksa opsi ini untuk menyimpan data telemetri aplikasi Anda di akun CloudWatch Log Anda di bawah Penyimpanan data. Secara default, grup CloudWatch log Log menyimpan data selama 30 hari. Anda dapat menyesuaikan periode retensi di konsol CloudWatch Log.
9. Untuk Otorisasi, tentukan apakah akan menggunakan kolam identitas Amazon Cognito baru atau yang sudah ada atau menggunakan penyedia identitas yang berbeda. Membuat kolam identitas baru merupakan opsi paling sederhana yang tidak memerlukan langkah pengaturan lainnya. Untuk informasi selengkapnya, lihat Langkah 1: Otorisasi aplikasi Anda untuk mengirim data AWS.

   Membuat kolam identitas Amazon Cognito baru memerlukan izin administratif. Untuk informasi selengkapnya, lihat Kebijakan IAM untuk menggunakan CloudWatch RUM.
10. (Opsional) Secara default, ketika Anda menambahkan cuplikan kode RUM ke aplikasi Anda, klien web menyuntikkan JavaScript tag untuk memantau penggunaan ke dalam kode HTML semua halaman aplikasi Anda. Untuk mengubah ini, pilih Konfigurasi halaman dan kemudian pilih Sertakan hanya halaman ini atau Kecualikan halaman ini. Kemudian, tentukan halaman yang akan disertakan atau dikecualikan. Untuk menentukan halaman yang akan disertakan atau dikecualikan, masukkan URL lengkapnya. Untuk menentukan halaman tambahan, pilih Tambahkan URL.
11. Untuk mengaktifkan AWS X-Ray penelusuran sesi pengguna yang diambil sampelnya oleh monitor aplikasi, pilih Penelusuran aktif dan pilih Lacak layanan saya dengan. AWS X-Ray

    Jika Anda memilih ini, permintaan `XMLHttpRequest` dan `fetch` yang dibuat selama sesi pengguna yang diambil sampelnya oleh monitor aplikasi akan dilacak. Anda kemudian dapat melihat jejak dan segmen dari sesi pengguna ini di dasbor RUM, dan peta jejak X-Ray dan halaman detail pelacakan. Sesi pengguna ini juga akan muncul sebagai halaman klien di Sinyal Aplikasi setelah Anda mengaktifkannya untuk aplikasi Anda.

    Dengan membuat perubahan konfigurasi tambahan pada klien web CloudWatch RUM, Anda dapat menambahkan header jejak X-Ray ke permintaan HTTP untuk mengaktifkan end-to-end

penelusuran sesi pengguna melalui layanan AWS terkelola hilir. Untuk informasi selengkapnya, lihat Mengaktifkan penelusuran X-Ray end-to-end .

12. (Opsional) Untuk menambahkan tanda ke monitor aplikasi, pilih Tag, Tambahkan tanda baru.

    Kemudian, untuk Kunci, masukkan nama untuk tanda tersebut. Anda dapat menambahkan sebuah nilai opsional untuk tanda di Nilai.

    Untuk menambahkan tanda lainnya, silakan pilih Tambahkan tanda baru lagi.

    Untuk informasi selengkapnya, lihat Menandai AWS Sumber Daya.
13. Pilih Tambahkan monitor aplikasi.
14. Di bagian kode Sampel, Anda dapat menyalin potongan kode yang akan digunakan untuk ditambahkan ke aplikasi Anda. Kami menyarankan Anda memilih JavaScriptatau TypeScriptdan menggunakan NPM untuk menginstal klien web CloudWatch RUM sebagai JavaScript modul.

    Atau, Anda dapat memilih HTML untuk menggunakan jaringan pengiriman konten (CDN) untuk menginstal klien web CloudWatch RUM. Kerugian menggunakan CDN adalah bahwa klien web sering diblokir oleh pemblokir iklan.
15. Pilih Salin atau Unduh, lalu pilih Selesai.

(Opsional) Langkah 3: Secara manual memodifikasi cuplikan kode untuk mengkonfigurasi klien web CloudWatch RUM

Anda dapat memodifikasi potongan kode sebelum memasukkannya ke dalam aplikasi Anda, untuk mengaktifkan atau menonaktifkan beberapa opsi. Untuk informasi selengkapnya, lihat dokumentasi klien web CloudWatch RUM.

Ada tiga opsi konfigurasi yang harus Anda ketahui, seperti yang dibahas di bagian ini.

Mencegah pengumpulan URL sumber daya yang mungkin berisi informasi pribadi

Secara default, klien web CloudWatch RUM dikonfigurasi untuk merekam URL sumber daya yang diunduh oleh aplikasi. Sumber daya ini termasuk file HTML, gambar, file CSS, JavaScript file, dan sebagainya. Untuk beberapa aplikasi, URL mungkin berisi informasi pengenal pribadi (PII).

Jika ini kasus untuk aplikasi Anda, kami sangat menyarankan agar Anda menonaktifkan pengumpulan URL sumber daya dengan mengatur `recordResourceUrl: false` dalam konfigurasi potongan kode, sebelum memasukkannya ke dalam aplikasi Anda.

Merekam tampilan halaman secara manual

Secara default, klien web mencatat tampilan halaman saat halaman pertama kali dimuat dan saat API riwayat browser dipanggil. ID halaman default adalah `window.location.pathname`. Namun demikian, dalam beberapa kasus Anda mungkin ingin mengganti perilaku ini dan instrumen aplikasi untuk merekam tampilan halaman secara terprogram. Melakukannya memberi Anda kendali atas ID halaman dan kapan direkam. Sebagai contoh, pertimbangkan aplikasi web yang memiliki URI dengan pengenal variabel, seperti `/entity/123` atau `/entity/456`. Secara default, CloudWatch RUM menghasilkan peristiwa tampilan halaman untuk setiap URI dengan ID halaman berbeda yang cocok dengan nama jalur, tetapi Anda mungkin ingin mengelompokkannya dengan ID halaman yang sama. Untuk mencapai hal ini, nonaktifkan otomatisasi tampilan halaman klien web menggunakan konfigurasi `disableAutoPageView`, dan gunakan perintah `recordPageView` untuk mengatur ID halaman yang diinginkan. Untuk informasi selengkapnya, lihat Konfigurasi Khusus Aplikasi pada. GitHub

Contoh skrip tersemat:

```
cwr('recordPageView', { pageId: 'entityPageId' });
```

JavaScript contoh modul:

```
awsRum.recordPageView({ pageId: 'entityPageId' });
```

Mengaktifkan penelusuran X-Ray end-to-end

Saat Anda membuat monitor aplikasi, memilih Lacak layanan saya dengan AWS X-Ray memungkinkan pelacakan permintaan `XMLHttpRequest` dan `fetch` yang dibuat selama sesi pengguna yang diambil sampelnya oleh monitor aplikasi. Anda kemudian dapat melihat jejak dari permintaan HTTP ini di dasbor CloudWatch RUM, dan halaman detail X-Ray Trace Map dan Trace.

Secara default, jejak sisi klien ini tidak terhubung ke jejak sisi server hilir. Untuk menghubungkan jejak sisi klien ke jejak sisi server dan mengaktifkan end-to-end penelusuran, atur `addXRayTraceIdHeader` opsi ke klien web. `true` Hal ini menyebabkan klien web CloudWatch RUM menambahkan header jejak X-Ray ke permintaan HTTP.

Blok kode berikut menunjukkan contoh penambahan jejak sisi klien. Beberapa opsi konfigurasi dihilangkan dari sampel ini untuk kemudahan pembacaan.

```
<script>
```

```
    (function(n,i,v,r,s,c,u,x,z){...})(
        'cwr',
        '00000000-0000-0000-0000-000000000000',
        '1.0.0',
        'us-west-2',
        'https://client.rum.us-east-1.amazonaws.com/1.0.2/cwr.js',
        {
            enableXRay: true,
            telemetries: [
                'errors',
                'performance',
                [ 'http', { addXRayTraceIdHeader: true } ]
            ]
        }
    );
</script>
```

> **Warning**
>
> Mengkonfigurasi klien web CloudWatch RUM untuk menambahkan header jejak X-Ray ke permintaan HTTP dapat menyebabkan berbagi sumber daya lintas asal (CORS) gagal atau membatalkan tanda tangan permintaan jika permintaan ditandatangani dengan SigV4. Untuk informasi selengkapnya, lihat dokumentasi klien web CloudWatch RUM. 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

Langkah 4: Masukkan potongan kode ke dalam aplikasi Anda

Berikutnya, Anda memasukkan potongan kode yang Anda buat di bagian sebelumnya ke dalam aplikasi Anda.

> **Warning**
>
> Klien web, yang diunduh dan dikonfigurasi oleh snippet kode, menggunakan cookie (atau teknologi serupa) untuk membantu Anda mengumpulkan data pengguna akhir. Sebelum Anda memasukkan potongan kode, silakan lihat Memfilter berdasarkan atribut metadata di konsol.

Jika Anda tidak memiliki potongan kode yang dibuat sebelumnya, Anda dapat menemukannya dengan mengikuti petunjuk di Bagaimana cara saya menemukan potongan kode yang sudah saya buat?.

Untuk memasukkan cuplikan kode CloudWatch RUM ke dalam aplikasi Anda

1. Masukkan potongan kode yang Anda salin atau unduh di bagian sebelumnya di dalam elemen <head> aplikasi Anda. Masukkan potongan tersebut sebelum elemen <body> atau tanda <script> lainnya.

   Berikut ini contoh potongan kode yang dihasilkan:

```
<script>
(function (n, i, v, r, s, c, x, z) {
    x = window.AwsRumClient = {q: [], n: n, i: i, v: v, r: r, c: c};
    window[n] = function (c, p) {
        x.q.push({c: c, p: p});
    };
    z = document.createElement('script');
    z.async = true;
    z.src = s;
    document.head.insertBefore(z, document.getElementsByTagName('script')[0]);
})('cwr',
    '194a1c89-87d8-41a3-9d1b-5c5cd3dafbd0',
    '1.0.0',
    'us-east-2',
    'https://client.rum.us-east-1.amazonaws.com/1.0.2/cwr.js',
    {
        sessionSampleRate: 1,
        guestRoleArn: "arn:aws:iam::123456789012:role/RUM-Monitor-us-
east-2-123456789012-5934510917361-Unauth",
        identityPoolId: "us-east-2:c90ef0ac-e3b8-4d1a-b313-7e73cfd21443",
        endpoint: "https://dataplane.rum.us-east-2.amazonaws.com",
        telemetries: ["performance", "errors", "http"],
        allowCookies: true,
        enableXRay: false
    });
</script>
```

2. Jika aplikasi Anda adalah aplikasi web multipage, Anda harus mengulangi langkah 1 untuk setiap halaman HTML yang ingin Anda sertakan dalam pengumpulan data.

Langkah 5: Menguji pengaturan monitor aplikasi Anda dengan membuat peristiwa pengguna

Setelah Anda memasukkan potongan kode dan aplikasi Anda yang diperbarui berjalan, Anda dapat mengujinya dengan membuat peristiwa pengguna secara manual. Untuk menguji ini, kami menyarankan Anda melakukan hal berikut. Pengujian ini menimbulkan biaya CloudWatch RUM standar.

- Menavigasi antar halaman dalam aplikasi web Anda.
- Buat beberapa sesi pengguna, menggunakan browser dan perangkat yang berbeda.
- Buat permintaan.
- Menyebabkan JavaScript kesalahan.

Setelah Anda membuat beberapa acara, lihat di dasbor CloudWatch RUM. Untuk informasi selengkapnya, lihat Melihat dasbor CloudWatch RUM.

Data dari sesi pengguna mungkin membutuhkan waktu hingga 15 menit untuk muncul di dasbor.

Jika tidak melihat data 15 menit setelah Anda membuat peristiwa dalam aplikasi, silakan lihat Pemecahan Masalah CloudWatch RUM.

Mengkonfigurasi klien web CloudWatch RUM

Aplikasi Anda dapat menggunakan salah satu cuplikan kode yang dihasilkan oleh CloudWatch RUM untuk menginstal klien web CloudWatch RUM. Cuplikan yang dihasilkan mendukung dua metode instalasi: sebagai JavaScript modul melalui NPM, atau dari jaringan pengiriman konten (CDN). Untuk performa terbaik, kami menyarankan menggunakan metode instalasi NPM. Untuk informasi selengkapnya tentang menggunakan metode ini, lihat Menginstal sebagai JavaScript Modul.

Jika Anda menggunakan opsi instalasi CDN, pemblokir iklan mungkin memblokir CDN default yang disediakan oleh RUM. CloudWatch Ini menonaktifkan pemantauan aplikasi untuk pengguna yang melakukan instalasi pemblokir iklan. Karena itu, kami menyarankan Anda menggunakan CDN default hanya untuk orientasi awal dengan CloudWatch RUM. Untuk informasi selengkapnya tentang cara mengurangi masalah ini, silakan lihat Instrumen aplikasi.

Potongan kode berada di tanda `<head>` file HTML dan melakukan instalasi klien web dengan mengunduh klien web, dan kemudian mengonfigurasi klien web untuk aplikasi yang dipantau.

Potongan ini adalah fungsi yang mengeksekusi diri sendiri yang terlihat mirip dengan yang berikut ini. Dalam contoh ini, tubuh fungsi potongan kode telah dihilangkan untuk kemudahaan pembacaan.

```
<script>
(function(n,i,v,r,s,c,u,x,z){...})(
'cwr',
'00000000-0000-0000-0000-000000000000',
'1.0.0',
'us-west-2',
'https://client.rum.us-east-1.amazonaws.com/1.0.2/cwr.js',
{ /* Configuration Options Here */ }
);
<script>
```

Argumen

Potongan kode menerima enam argumen:

- Namespace untuk menjalankan perintah pada klien web, seperti `'cwr'`
- ID monitor aplikasi, seperti `'00000000-0000-0000-0000-000000000000'`
- Versi aplikasi, seperti `'1.0.0'`
- AWS Wilayah monitor aplikasi, seperti `'us-west-2'`
- URL dari klien web, seperti `'https://client.rum.us-east-1.amazonaws.com/1.0.2/cwr.js'`
- Opsi konfigurasi khusus aplikasi. Untuk informasi selengkapnya, silakan lihat bagian berikut ini.

Mengabaikan kesalahan

Klien web CloudWatch RUM mendengarkan semua jenis kesalahan yang terjadi dalam aplikasi Anda. Jika aplikasi Anda memancarkan JavaScript kesalahan yang tidak ingin Anda lihat di dasbor CloudWatch RUM, Anda dapat mengonfigurasi klien web CloudWatch RUM untuk menyaring kesalahan ini sehingga Anda hanya melihat peristiwa kesalahan yang relevan di dasbor CloudWatch RUM. Misalnya, Anda mungkin memilih untuk tidak melihat beberapa JavaScript kesalahan di dasbor karena Anda telah mengidentifikasi perbaikan untuk mereka dan volume kesalahan ini menutupi kesalahan lainnya. Anda mungkin juga ingin mengabaikan kesalahan yang tidak dapat Anda perbaiki karena dimiliki oleh perpustakaan yang dimiliki oleh pihak ketiga.

Untuk informasi selengkapnya tentang cara instrumen klien web untuk memfilter JavaScript kesalahan tertentu, lihat contoh di Kesalahan dalam dokumentasi Github klien web.

Opsi konfigurasi

Untuk informasi tentang opsi konfigurasi yang tersedia untuk klien web CloudWatch RUM, lihat dokumentasi klien web CloudWatch RUM

Regionalisasi

Bagian ini menggambarkan strategi untuk menggunakan CloudWatch RUM dengan aplikasi di Wilayah yang berbeda.

Aplikasi web saya digunakan di beberapa Wilayah AWS

Jika aplikasi web Anda digunakan di AWS Wilayah berlipat ganda, Anda memiliki tiga opsi:

- Terapkan satu monitor aplikasi di satu Wilayah, dalam satu akun, yang melayani semua Wilayah.
- Terapkan monitor aplikasi terpisah untuk setiap Wilayah, di akun unik.
- Terapkan monitor aplikasi terpisah untuk setiap Wilayah, semuanya dalam satu akun.

Keuntungan menggunakan satu monitor aplikasi adalah bahwa semua data akan dipusatkan menjadi satu visualisasi, dan semua log ditulis ke grup log yang sama di CloudWatch Log. Dengan monitor aplikasi tunggal ada sejumlah kecil latensi tambahan untuk permintaan, dan satu titik kegagalan.

Menggunakan beberapa monitor aplikasi menghilangkan satu titik kegagalan, tetapi mencegah semua data digabungkan menjadi satu visualisasi.

CloudWatch RUM belum diluncurkan di beberapa Wilayah tempat aplikasi saya digunakan

CloudWatch RUM diluncurkan ke banyak Wilayah dan memiliki cakupan geografis yang luas. Dengan menyiapkan CloudWatch RUM di Wilayah yang tersedia, Anda bisa mendapatkan manfaatnya. Pengguna akhir dapat berada di mana saja dan masih menyertakan sesi mereka jika Anda telah mengatur monitor aplikasi di Wilayah yang menghubungkannya.

Namun, CloudWatch RUM belum diluncurkan di AWS GovCloud (AS-Timur), AWS GovCloud (AS-Barat), atau Wilayah mana pun di China. Anda tidak dapat mengirim data ke CloudWatch RUM dari Wilayah ini.

Gunakan grup halaman

Gunakan grup halaman untuk mengaitkan halaman yang berbeda dalam aplikasi Anda satu sama lain sehingga Anda dapat melihat analitik gabungan untuk grup halaman. Sebagai contoh, Anda mungkin ingin melihat waktu unggah halaman gabungan dari semua halaman arahan Anda.

Anda menempatkan halaman ke dalam grup halaman dengan menambahkan satu atau beberapa tag ke acara tampilan halaman di klien web CloudWatch RUM. Contoh berikut menempatkan halaman `/home` ke dalam grup halaman yang disebut `en` dan grup halaman yang disebut `landing`.

Contoh skrip tertanam

```
cwr('recordPageView', { pageId: '/home', pageTags: ['en', 'landing']});
```

JavaScript contoh modul

```
awsRum.recordPageView({ pageId: '/home', pageTags: ['en', 'landing']});
```

> **Note**
>
> Grup halaman dimaksudkan untuk memudahkan penggabungan analitik di berbagai halaman. Untuk informasi tentang cara mendefinisikan dan memanipulasi `pageIds` untuk aplikasi Anda, silakan lihat bagian Merekam tampilan halaman secara manual di (Opsional) Langkah 3: Secara manual memodifikasi cuplikan kode untuk mengkonfigurasi klien web CloudWatch RUM.

Tentukan metadata kustom

CloudWatch RUM melampirkan data tambahan ke setiap peristiwa sebagai metadata. Metadata peristiwa terdiri atas atribut dalam bentuk pasangan nilai kunci. Anda dapat menggunakan atribut ini untuk mencari atau memfilter peristiwa di konsol CloudWatch RUM. Secara default, CloudWatch RUM membuat beberapa metadata untuk Anda. Untuk informasi selengkapnya tentang metadata instans, silakan lihat Metadata peristiwa RUM.

Anda juga dapat menggunakan klien web CloudWatch RUM untuk menambahkan metadata khusus ke acara CloudWatch RUM. Metadata kustom dapat mencakup atribut sesi dan atribut halaman.

Untuk menambahkan metadata kustom, Anda harus menggunakan versi 1.10.0 atau yang lebih baru dari klien web RUM. CloudWatch

Persyaratan dan sintaks

Setiap peristiwa dapat menyertakan sebanyak 10 atribut khusus dalam metadata. Persyaratan sintaks untuk atribut kustom adalah sebagai berikut:

- Kunci
  - Maksimal 128 karakter
  - Dapat menyertakan karakter alfanumerik, titik bawah (:), dan garis bawah (_)
  - Tidak bisa mulai dengan `aws:`.
  - Tidak dapat sepenuhnya terdiri atas salah satu kata kunci yang dicadangkan yang tercantum di bagian berikut. Dapat menggunakan kata kunci tersebut sebagai bagian dari nama kunci yang lebih panjang.
- Nilai-nilai
  - Maksimal 256 karakter
  - Harus berupa string, angka, atau nilai Boolean

Kata Kunci Cadangan

Anda tidak dapat menggunakan kata kunci berikut sebagai nama kunci lengkap. Anda dapat menggunakan kata kunci berikut sebagai bagian dari nama kunci yang lebih panjang, seperti `applicationVersion`.

- `browserLanguage`
- `browserName`
- `browserVersion`
- `countryCode`
- `deviceType`
- `domain`
- `interaction`
- `osName`
- `osVersion`

- `pageId`
- `pageTags`
- `pageTitle`
- `pageUrl`
- `parentPageId`
- `platformType`
- `referrerUrl`
- `subdivisionCode`
- `title`
- `url`
- `version`

> **Note**
>
> CloudWatch RUM menghapus atribut kustom dari peristiwa RUM jika atribut menyertakan kunci atau nilai yang tidak valid, atau jika batas 10 atribut kustom per peristiwa telah tercapai.

Tambahkan atribut sesi

Jika Anda mengonfigurasi atribut sesi kustom, atribut tersebut akan ditambahkan ke semua peristiwa dalam sesi. Anda mengonfigurasi atribut sesi baik selama inisialisasi klien web CloudWatch RUM atau saat runtime dengan menggunakan perintah. `addSessionAttributes`

Sebagai contoh, Anda dapat menambahkan versi aplikasi Anda sebagai atribut sesi. Kemudian, di konsol CloudWatch RUM, Anda dapat memfilter kesalahan berdasarkan versi untuk mengetahui apakah tingkat kesalahan yang meningkat dikaitkan dengan versi tertentu dari aplikasi Anda.

Menambahkan atribut sesi pada inisialisasi, contoh NPM

Bagian kode dalam huruf tebal menambahkan atribut sesi.

```
import { AwsRum, AwsRumConfig } from 'aws-rum-web';

try {
```

```
  const config: AwsRumConfig = {
    allowCookies: true,
    endpoint: "https://dataplane.rum.us-west-2.amazonaws.com",
    guestRoleArn: "arn:aws:iam::000000000000:role/RUM-Monitor-us-
west-2-000000000000-00xx-Unauth",
    identityPoolId: "us-west-2:00000000-0000-0000-0000-000000000000",
    sessionSampleRate: 1,
    telemetries: ['errors', 'performance'],
    sessionAttributes: {
        applicationVersion: "1.3.8"
    }
  };

  const APPLICATION_ID: string = '00000000-0000-0000-0000-000000000000';
  const APPLICATION_VERSION: string = '1.0.0';
  const APPLICATION_REGION: string = 'us-west-2';

  const awsRum: AwsRum = new AwsRum(
    APPLICATION_ID,
    APPLICATION_VERSION,
    APPLICATION_REGION,
    config
  );
} catch (error) {
  // Ignore errors thrown during CloudWatch RUM web client initialization
}
```

Menambahkan atribut sesi pada runtime, contoh NPM

```
awsRum.addSessionAttributes({
    applicationVersion: "1.3.8"
})
```

Menambahkan atribut sesi pada inisialisasi, contoh skrip tersemat

Bagian kode dalam huruf tebal menambahkan atribut sesi.

```
<script>
    (function(n,i,v,r,s,c,u,x,z){...})(
        'cwr',
        '00000000-0000-0000-0000-000000000000',
        '1.0.0',
        'us-west-2',
```

```
        'https://client.rum.us-east-1.amazonaws.com/1.0.2/cwr.js',
        {
            sessionSampleRate:1,
            guestRoleArn:'arn:aws:iam::000000000000:role/RUM-Monitor-us-
west-2-000000000000-00xx-Unauth',
            identityPoolId:'us-west-2:00000000-0000-0000-0000-000000000000',
            endpoint:'https://dataplane.rum.us-west-2.amazonaws.com',
            telemetries:['errors','http','performance'],
            allowCookies:true,
            sessionAttributes: {
                applicationVersion: "1.3.8"
            }
        }
    );
</script>
```

Menambahkan atribut sesi pada runtime, contoh skrip tersemat

```
<script>
    function addSessionAttribute() {
        cwr('addSessionAttributes', {
            applicationVersion: "1.3.8"
        })
    }

</script>
```

Menambahkan atribut halaman

Jika Anda mengonfigurasi atribut sesi kustom, atribut tersebut akan ditambahkan ke semua peristiwa di halaman saat ini. Anda mengonfigurasi atribut halaman baik selama inisialisasi klien web CloudWatch RUM atau saat runtime dengan menggunakan perintah. `recordPageView`

Sebagai contoh, Anda dapat menambahkan template Anda sebagai atribut halaman. Kemudian, di konsol CloudWatch RUM, Anda dapat memfilter kesalahan berdasarkan templat halaman untuk mengetahui apakah tingkat kesalahan yang meningkat dikaitkan dengan templat halaman tertentu dari aplikasi Anda.

Menambahkan atribut halaman pada inisialisasi, contoh NPM

Bagian kode dalam huruf tebal menambahkan atribut halaman.

```
const awsRum: AwsRum = new AwsRum(
    APPLICATION_ID,
    APPLICATION_VERSION,
    APPLICATION_REGION,
    { disableAutoPageView:  true // optional }
);
awsRum.recordPageView({
    pageId:'/home',
    pageAttributes: {
      template: 'artStudio'
    }
});
const credentialProvider = new CustomCredentialProvider();
if(awsCreds) awsRum.setAwsCredentials(credentialProvider);
```

Menambahkan atribut halaman pada runtime, contoh NPM

```
awsRum.recordPageView({
    pageId: '/home',
    pageAttributes: {
        template: 'artStudio'
    }
});
```

Menambahkan atribut halaman pada inisialisasi, contoh skrip yang ditanamkan

Bagian kode dalam huruf tebal menambahkan atribut halaman.

```
<script>
    (function(n,i,v,r,s,c,u,x,z){...})(
        'cwr',
        '00000000-0000-0000-0000-000000000000',
        '1.0.0',
        'us-west-2',
        'https://client.rum.us-east-1.amazonaws.com/1.0.2/cwr.js',
        {
            disableAutoPageView: true //optional
        }
    );
    cwr('recordPageView', {
       pageId: '/home',
       pageAttributes: {
```

```
            template: 'artStudio'
        }
    });
    const awsCreds = localStorage.getItem('customAwsCreds');
    if(awsCreds) cwr('setAwsCredentials', awsCreds)
</script>
```

Menambahkan atribut halaman pada runtime, contoh skrip yang ditanamkan

```
<script>
    function recordPageView() {
        cwr('recordPageView', {
            pageId: '/home',
            pageAttributes: {
                template: 'artStudio'
            }
        });
    }
</script>
```

Memfilter berdasarkan atribut metadata di konsol

Untuk memfilter visualisasi di konsol CloudWatch RUM dengan atribut metadata bawaan atau kustom, gunakan bilah pencarian. Di bilah pencarian, Anda dapat menentukan sebanyak 20 istilah filter dalam bentuk key=value untuk diterapkan pada visualisasi. Misalnya, untuk memfilter data hanya untuk browser Chrome, Anda dapat menambahkan istilah filter browserName=Chrome.

Secara default, konsol CloudWatch RUM mengambil 100 kunci dan nilai atribut paling umum untuk ditampilkan di dropdown di bilah pencarian. Untuk menambahkan lebih banyak atribut metadata sebagai istilah filter, masukkan kunci atribut lengkap dan nilai ke dalam bilah pencarian.

Filter dapat mencakup sebanyak 20 istilah filter, dan Anda dapat menyimpan hingga 20 filter per monitor aplikasi. Saat Anda menyimpan filter, itu disimpan di dropdown Filter yang disimpan. Anda juga dapat menghapus filter yang disimpan.

Kirim peristiwa kustom

CloudWatch RUM mencatat dan mencerna peristiwa yang tercantum diInformasi yang dikumpulkan oleh klien web CloudWatch RUM. Jika Anda menggunakan versi 1.12.0 atau yang lebih baru dari klien web CloudWatch RUM, Anda dapat menentukan, merekam, dan mengirim acara khusus

tambahan. Anda menentukan nama jenis peristiwa dan data yang akan dikirim untuk setiap jenis peristiwa yang Anda tentukan. Setiap muatan peristiwa khusus dapat mencapai 6 KB.

Peristiwa kustom hanya diserap jika monitor aplikasi telah mengaktifkan peristiwa kustom. Untuk memperbarui setelan konfigurasi monitor aplikasi Anda, gunakan konsol CloudWatch RUM atau UpdateAppMonitorAPI.

Setelah mengaktifkan peristiwa kustom, dan kemudian menentukan dan mengirim peristiwa kustom, Anda dapat mencarinya. Untuk mencarinya, gunakan tab Acara di konsol CloudWatch RUM. Cari menggunakan jenis peristiwa.

Persyaratan dan sintaks

Peristiwa kustom terdiri atas jenis peristiwa dan detail peristiwa. Persyaratan untuk ini adalah sebagai berikut:

- Jenis peristiwa
  - Ini bisa berupa jenis atau nama peristiwa Anda. Misalnya, tipe acara bawaan CloudWatch RUM yang disebut JsErrormemiliki tipe acara`com.amazon.rum.js_error_event`.
  - Harus antara 1 hingga 256 karakter.
  - Dapat berupa kombinasi karakter alfanumerik, garis bawah, tanda hubung, dan titik.
- Detail peristiwa
  - Berisi data aktual yang ingin Anda rekam dalam CloudWatch RUM.
  - Harus berupa objek yang terdiri atas bidang dan nilai.

Contoh merekam peristiwa kustom

Ada dua cara untuk merekam peristiwa khusus di klien web CloudWatch RUM.

- Gunakan `recordEvent` API klien web CloudWatch RUM.
- Gunakan plugin yang disesuaikan.

Kirim peristiwa kustom menggunakan API **`recordEvent`**, contoh NPM

```
awsRum.recordEvent('my_custom_event', {
        location: 'IAD',
        current_url: 'amazonaws.com',
        user_interaction: {
```

```
            interaction_1 : "click",
            interaction_2 : "scroll"
        },
        visit_count:10
    }
)
```

Kirim peristiwa kustom menggunakan API **`recordEvent`**, contoh skrip yang ditanamkan

```
cwr('recordEvent', {
    type: 'my_custom_event',
    data: {
        location: 'IAD',
        current_url: 'amazonaws.com',
        user_interaction: {
            interaction_1 : "click",
            interaction_2 : "scroll"
        },
        visit_count:10
    }
})
```

Contoh pengiriman peristiwa khusus menggunakan plugin yang disesuaikan

```
// Example of a plugin that listens to a scroll event, and
// records a 'custom_scroll_event' that contains the timestamp of the event.
class MyCustomPlugin implements Plugin {
    // Initialize MyCustomPlugin.
    constructor() {
        this.enabled;
        this.context;
        this.id = 'custom_event_plugin';
    }
    // Load MyCustomPlugin.
    load(context) {
        this.context = context;
        this.enable();
    }
    // Turn on MyCustomPlugin.
    enable() {
        this.enabled = true;
        this.addEventHandler();
    }
```

```
    // Turn off MyCustomPlugin.
    disable() {
        this.enabled = false;
        this.removeEventHandler();
    }
    // Return MyCustomPlugin Id.
    getPluginId() {
        return this.id;
    }
    // Record custom event.
    record(data) {
        this.context.record('custom_scroll_event', data);
    }
    // EventHandler.
    private eventHandler = (scrollEvent: Event) => {
        this.record({timestamp: Date.now()})
    }
    // Attach an eventHandler to scroll event.
    private addEventHandler(): void {
        window.addEventListener('scroll', this.eventHandler);
    }
    // Detach eventHandler from scroll event.
    private removeEventHandler(): void {
        window.removeEventListender('scroll', this.eventHandler);
    }
}
```

Melihat dasbor CloudWatch RUM

CloudWatch RUM membantu Anda mengumpulkan data dari sesi pengguna tentang kinerja aplikasi Anda, termasuk waktu muat halaman, skor Apdex, browser dan perangkat yang digunakan, geolokasi sesi pengguna, dan sesi dengan kesalahan. Semua informasi ini ditampilkan di dasbor.

Untuk melihat dasbor RUM

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Sinyal Aplikasi, RUM.

   Tab Gambaran umum menampilkan informasi yang dikumpulkan oleh salah satu monitor aplikasi yang telah Anda buat.

   Baris atas panel menampilkan informasi berikut untuk monitor aplikasi ini:

- Jumlah pemuatan halaman
- Rata-rata kecepatan pemuatan halaman
- Skor Apdex
- Status alarm apa pun yang terkait dengan monitor aplikasi

Skor indeks performa aplikasi (Apdex) menunjukkan tingkat kepuasan pengguna akhir. Skor berkisar dari 0 (paling tidak puas) hingga 1 (paling puas). Skor didasarkan pada performa aplikasi saja. Pengguna tidak diminta untuk menilai aplikasi. Untuk informasi selengkapnya tentang skor Apdex, silakan lihat Bagaimana CloudWatch RUM menetapkan skor Apdex.

Beberapa panel ini menyertakan tautan yang dapat Anda gunakan untuk memeriksa data lebih lanjut. Memilih salah satu tautan ini menampilkan tampilan terperinci dengan Kinerja, Kesalahan, permintaan HTTP, Sesi, Browser & Perangkat Peristiwa, dan tab Perjalanan Pengguna di bagian atas layar.

3. Untuk lebih fokus, pilih tab Tampilan daftar dan kemudian pilih nama monitor aplikasi yang ingin Anda fokuskan. Ini menampilkan tab berikut untuk monitor aplikasi yang dipilih.

   - Tab Kinerja menampilkan informasi performa halaman termasuk waktu muat, informasi sesi, informasi permintaan, vital web, dan pemuatan halaman dari waktu ke waktu. Tampilan ini mencakup kendali untuk mengubah tampilan antara fokus pada pemuatan Halaman, Permintaan, dan Lokasi.
   - Tab Kesalahan menampilkan informasi kesalahan Javascript termasuk pesan kesalahan yang paling sering dilihat pengguna serta perangkat dan browser dengan kesalahan terbanyak. Tampilan ini mencakup histogram kesalahan dan tampilan daftar kesalahan. Anda dapat memfilter daftar kesalahan berdasarkan detail pengguna dan peristiwa. Pilih pesan kesalahan untuk melihat detail lebih lanjut.
   - Tab permintaan HTTP menampilkan informasi permintaan HTTP termasuk URL permintaan dengan sebagian besar kesalahan serta perangkat dan browser dengan kesalahan terbanyak. Tab ini mencakup histogram permintaan, tampilan daftar permintaan, dan tampilan daftar kesalahan jaringan. Anda dapat memfilter daftar berdasarkan detail pengguna dan peristiwa. Pilih kode respons atau pesan kesalahan untuk mendapatkan detail lebih lanjut tentang masing-masing permintaan atau kesalahan jaringan.
   - Tab Sesi menampilkan metrik sesi. Tab ini mencakup histogram peristiwa awal sesi dan tampilan daftar sesi. Anda dapat memfilter daftar sesi berdasarkan jenis peristiwa, detail pengguna, dan detail peristiwa. Pilih sessionId untuk melihat detail lebih lanjut tentang sesi.

- Tab Peristiwa menampilkan histogram peristiwa RUM dan tampilan daftar peristiwa. Anda dapat memfilter daftar peristiwa berdasarkan jenis peristiwa, detail pengguna, dan detail peristiwa. Pilih peristiwa RUM untuk melihat peristiwa mentah.
- Tab Browser & Perangkat menampilkan informasi seperti performa dan penggunaan berbagai browser dan perangkat untuk mengakses aplikasi Anda. Tampilan ini mencakup kendali untuk mengaktifkan tampilan antara fokus pada Browser dan Perangkat.

  Jika mempersempit cakupan ke satu browser, Anda melihat data dipecah berdasarkan versi browser.
- Tab Perjalanan Pengguna menampilkan jalur yang digunakan pelanggan Anda untuk menavigasi aplikasi Anda. Anda dapat melihat dari mana pelanggan Anda memasukkan aplikasi Anda dan dari halaman mana mereka keluar dari aplikasi Anda. Anda juga dapat melihat jalur yang mereka ambil dan persentase pelanggan yang mengikuti jalur tersebut. Anda dapat berhenti sejenak pada simpul untuk mendapatkan detail lebih lanjut tentang halaman itu. Anda dapat memilih satu jalur untuk menyorot koneksi agar lebih mudah dilihat.

4. (Opsional) Pada salah satu dari enam tab pertama, Anda dapat memilih tombol Halaman dan memilih halaman atau grup halaman dari daftar. Ini mempersempit data yang ditampilkan ke satu halaman atau grup halaman aplikasi Anda. Anda juga dapat menandai halaman dan grup halaman dalam daftar sebagai favorit.

Bagaimana CloudWatch RUM menetapkan skor Apdex

Apdex (Application Performance Index) adalah standar terbuka yang mendefinisikan metode untuk melaporkan, mengukur, dan menilai waktu respons aplikasi. Skor Apdex membantu Anda memahami dan mengidentifikasi dampak pada performa aplikasi dari waktu ke waktu.

Skor Apdex menunjukkan tingkat kepuasan pengguna akhir Skor yang berkisar dari 0 (paling tidak puas) hingga 1 (paling puas). Skor didasarkan pada performa aplikasi saja. Pengguna tidak diminta untuk menilai aplikasi.

Setiap skor Apdex individu jatuh ke dalam salah satu dari tiga ambang batas. Berdasarkan ambang batas Apdex dan waktu respons aplikasi aktual, ada tiga jenis performa, sebagai berikut:

- Puas– Waktu respons aplikasi sebenarnya kurang dari atau sama dengan ambang batas Apdex. Untuk CloudWatch RUM, ambang batas ini adalah 2000 ms atau kurang.

- Dapat Ditoleransi– Waktu respons aplikasi sebenarnya lebih besar dari ambang batas Apdex, tetapi kurang dari atau sama dengan empat kali ambang batas Apdex. Untuk CloudWatch RUM, kisaran ini adalah 2000-8000 ms.
- Mengecewakan– Waktu respons aplikasi sebenarnya lebih besar dari empat kali ambang batas Apdex. Untuk CloudWatch RUM, kisaran ini lebih dari 8000 ms.

Total skor Apdex 0-1 dihitung menggunakan rumus berikut:

```
(positive scores + tolerable scores/2)/total scores * 100
```

CloudWatch metrik yang dapat Anda kumpulkan dengan CloudWatch RUM

Tabel di bagian ini mencantumkan metrik yang Anda kumpulkan secara otomatis dengan CloudWatch RUM. Anda dapat melihat metrik ini di CloudWatch konsol. Untuk informasi selengkapnya, lihat Lihat metrik yang tersedia.

Anda juga dapat secara opsional mengirim metrik yang diperluas ke CloudWatch atau CloudWatch Terbukti. Untuk informasi selengkapnya, lihat Metrik yang diperluas.

Metrik-metrik ini diterbitkan di namespace metrik yang disebut `AWS/RUM`. Semua metrik berikut diterbitkan dengan dimensi `application_name`. Nilai dimensi ini adalah nama monitor aplikasi. Beberapa metrik juga diterbitkan dengan dimensi tambahan, seperti yang tercantum dalam tabel.

Metrik	Unit	Deskripsi
`HttpStatusCodeCount`	Hitungan	Hitungan respons HTTP dalam aplikasi, dengan kode status responsnya. Dimensi tambahan: • `event_details.response.status` adalah kode status respons, seperti 200, 400, 404, dan seterusnya.

Metrik	Unit	Deskripsi
		• `event_type` Jenis peristiwa . Saat ini, satu-satunya nilai yang mungkin untuk dimensi ini adalah `http`.
`Http4xxCount`	Hitungan	Hitungan respons HTTP dalam aplikasi, dengan kode status respons 4xx. Ini dihitung berdasarkan peristiwa `http_event` RUM yang menghasilkan kode 4xx.
`Http5xxCount`	Hitungan	Hitungan respons HTTP dalam aplikasi, dengan kode status respons 5xx. Ini dihitung berdasarkan peristiwa `http_event` RUM yang menghasilkan kode 5xx.
`JsErrorCount`	Hitungan	Hitungan peristiwa JavaScript kesalahan yang dicerna.

Metrik	Unit	Deskripsi
NavigationFrustratedCount	Hitungan	Hitungan peristiwa navigasi dengan ambang batas duration yang lebih tinggi dari ambang batas mengecewakan, yaitu 8000 ms. Durasi peristiwa navigasi dilacak dalam metrik PerformanceNavigationDuration .
NavigationSatisfiedCount	Hitungan	Hitungan peristiwa navigasi dengan duration yang kurang dari tujuan Apdex, yaitu 2000 ms. Durasi peristiwa navigasi dilacak dalam metrik PerformanceNavigationDuration .

Metrik	Unit	Deskripsi
`NavigationToleratedCount`	Hitungan	Hitungan peristiwa navigasi dengan `duration` antara 2000 ms dan 8000 ms. Durasi peristiwa navigasi dilacak dalam metrik `PerformanceNavigationDuration` .
`PageViewCount`	Hitungan	Hitungan peristiwa tampilan halaman yang dicerna oleh monitor aplikasi. Ini dihitung dengan menghitung peristiwa `page_view_event` RUM.

Metrik	Unit	Deskripsi
`PerformanceResourceDuration`	Milidetik	`duration` dari peristiwa sumber daya. Dimensi tambahan: • `event_details.file.type` adalah jenis file dari peristiwa sumber daya, seperti stylesheet, dokumen, gambar, skrip, atau font. • `event_type` Jenis peristiwa . Saat ini, satu-satunya nilai yang mungkin untuk dimensi ini adalah `resource`.
`PerformanceNavigationDuration`	Milidetik	`duration` dari peristiwa navigasi.

Metrik	Unit	Deskripsi
`RumEventPayloadSize`	Byte	Ukuran setiap acara yang dicerna oleh CloudWatch RUM. Anda juga dapat menggunakan statistik `SampleCount` untuk metrik ini untuk memantau jumlah peristiwa yang diserap monitor aplikasi.
`SessionCount`	Hitungan	Hitungan peristiwa awal sesi yang diserap monitor aplikasi. Dengan kata lain, jumlah sesi baru dimulai.
`WebVitalsCumulativeLayoutShift`	Tidak ada	Melacak nilai peristiwa pergeseran tata letak kumulatif.
`WebVitalsFirstInputDelay`	Milidetik	Melacak nilai peristiwa penundaan input pertama.
`WebVitalsLargestContentfulPaint`	Milidetik	Melacak nilai peristiwa catatan terbesar yang memuaskan.

Metrik kustom dan metrik diperpanjang yang dapat Anda kirim dan Terbukti CloudWatch CloudWatch

Secara default, monitor aplikasi RUM mengirim metrik ke CloudWatch. Metrik dan dimensi default ini tercantum dalam CloudWatch metrik yang dapat Anda kumpulkan dengan CloudWatch RUM.

Anda juga dapat mengatur monitor aplikasi untuk mengirim metrik yang diperluas, metrik khusus, atau keduanya ke CloudWatch atau ke CloudWatch Evidently.

- Metrik kustom– Metrik kustom adalah metrik yang Anda tentukan. Dengan metrik kustom, Anda dapat menggunakan nama metrik dan namespace apa pun. Untuk mendapatkan metrik, Anda dapat menggunakan peristiwa kustom, peristiwa bawaan, atribut kustom, atau atribut default.

  Anda dapat mengirim metrik khusus ke keduanya CloudWatch dan CloudWatch Terbukti.
- Metrik yang diperluas — Memungkinkan Anda mengirim metrik CloudWatch RUM default ke CloudWatch Evidently untuk digunakan dalam eksperimen Evidently. Anda juga dapat mengirim salah satu metrik CloudWatch RUM default CloudWatch dengan dimensi tambahan. Dengan cara ini, metrik ini dapat memberi Anda tampilan yang lebih halus.

Topik

Metrik-metrik kustom

Untuk mengirim metrik khusus, Anda harus menggunakan AWS API atau AWS CLI bukan konsol. Untuk informasi selengkapnya tentang penggunaan AWS API, lihat PutRumMetricsDestinationdan BatchCreateRumMetricDefinitions.

Jumlah maksimum definisi metrik yang diperluas dan metrik kustom yang dapat memuat satu tujuan adalah 2000. Untuk setiap metrik kustom atau metrik yang diperluas yang Anda kirim ke setiap tujuan, setiap kombinasi nama dimensi dan nilai dimensi diperhitungkan dalam batas ini. Ini juga dihitung sebagai metrik CloudWatch khusus untuk harga.

Contoh berikut menunjukkan cara membuat metrik kustom yang berasal dari peristiwa kustom. Berikut adalah contoh peristiwa kustom yang digunakan:

```
cwr('recordEvent', {
```

```
    type: 'my_custom_event',
    data: {
        location: 'IAD',
        current_url: 'amazonaws.com',
        user_interaction: {
            interaction_1 : "click",
            interaction_2 : "scroll"
        },
        visit_count:10
    }
})
```

Dengan peristiwa kustom ini, Anda dapat membuat metrik kustom yang menghitung jumlah kunjungan ke URL `amazonaws.com` dari browser Chrome. Definisi berikut membuat metrik yang disebut `AmazonVisitsCount` di akun Anda, di namespace `RUM/CustomMetrics/PageVisits`.

```
{
    "AppMonitorName":"customer-appMonitor-name",
    "Destination":"CloudWatch",
    "MetricDefinitions":[
        {
            "Name":"AmazonVisitsCount",
            "Namespace":"PageVisit",
            "ValueKey":"event_details.visit_count",
            "UnitLabel":"Count",
            "DimensionKeys":{
                "event_details.current_url": "URL"
            },
            "EventPattern":"{\"metadata\":{\"browserName\":[\"Chrome\"]},\"event_type
\":[\"my_custom_event\"],\"event_details\": {\"current_url\": [\"amazonaws.com\"]}}"
        }
    ]
}
```

Metrik yang diperluas

Jika mengatur metrik yang diperluas, Anda dapat melakukan salah satu atau kedua hal berikut:

- Kirim metrik CloudWatch RUM default ke CloudWatch Evidently untuk digunakan dalam eksperimen Evidently. Hanya WebVitalsLargestContentfulPaintmetrik PerformanceNavigationDuration, PerformanceResourceDuration, WebVitalsCumulativeLayoutShift, WebVitalsFirstInputDelay, dan yang dapat dikirim ke Evidently.

- Kirim salah satu metrik CloudWatch RUM default CloudWatch dengan dimensi tambahan sehingga metrik memberi Anda tampilan yang lebih halus. Sebagai contoh, Anda dapat melihat metrik kustom untuk browser tertentu yang digunakan pengguna Anda, atau metrik untuk pengguna di geolokasi tertentu.

Untuk informasi selengkapnya tentang metrik CloudWatch RUM default, lihatCloudWatch metrik yang dapat Anda kumpulkan dengan CloudWatch RUM.

Jumlah maksimum definisi metrik yang diperluas dan metrik kustom yang dapat memuat satu tujuan adalah 2000. Untuk setiap metrik kustom atau metrik yang diperluas yang Anda kirim ke setiap tujuan, setiap kombinasi nama dimensi dan nilai dimensi diperhitungkan untuk batas ini. Ini juga dihitung sebagai metrik CloudWatch khusus untuk harga.

Saat Anda mengirim metrik yang diperluas ke CloudWatch, Anda dapat menggunakan konsol CloudWatch RUM untuk membuat CloudWatch alarm pada mereka.

Metrik yang diperluas dikenakan biaya sebagai metrik CloudWatch khusus. Untuk informasi selengkapnya, silakan lihat Harga Amazon CloudWatch.

Dimensi berikut didukung untuk metrik yang diperluas untuk semua nama metrik yang dapat dikirim monitor aplikasi. Nama-nama metrik ini tercantum dalam CloudWatch metrik yang dapat Anda kumpulkan dengan CloudWatch RUM.

- `BrowserName`

  Contoh nilai dimensi: `Chrome`, `Firefox`, `Chrome Headless`
- `CountryCode` Ini menggunakan format ISO-3166, dengan kode dua huruf.

  Contoh nilai dimensi: `US`, `JP`, `DE`
- `DeviceType`

  Contoh nilai dimensi: `desktop`, `mobile`, `tablet`, `embedded`
- `FileType`

  Contoh nilai dimensi: `Image`, `Stylesheet`
- `OSName`

  Contoh nilai dimensi: `Linux`, `Windows, iOS`, `Android`

- `PageId`

Atur metrik yang diperluas menggunakan konsol

Untuk menggunakan konsol untuk mengirim metrik yang diperluas CloudWatch, gunakan langkah-langkah berikut.

Untuk mengirim metrik yang diperluas ke CloudWatch Evidently, Anda harus menggunakan AWS API atau AWS CLI bukan konsol. Untuk informasi tentang penggunaan AWS API untuk mengirim metrik yang diperluas ke salah satu CloudWatch atau Terbukti, lihat PutRumMetricsDestinationdan. BatchCreateRumMetricDefinitions

Untuk menggunakan konsol untuk menyiapkan monitor aplikasi dan mengirim metrik diperpanjang RUM ke CloudWatch

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Sinyal Aplikasi, RUM.
3. Pilih Tampilan daftar lalu pilih nama monitor aplikasi yang akan mengirim metrik.
4. Pilih tab Konfigurasi dan kemudian pilih metrik yang diperluas RUM.
5. Pilih Kirim metrik.
6. Pilih satu atau beberapa nama metrik untuk dikirim dengan dimensi tambahan.
7. Pilih satu faktor atau lebih untuk digunakan sebagai dimensi untuk metrik ini. Saat Anda membuat pilihan, jumlah metrik tambahan yang dibuat pilihan Anda ditampilkan di Jumlah metrik terekstensi.

   Angka ini dihitung dengan mengalikan jumlah nama metrik yang dipilih dengan jumlah dimensi berbeda yang Anda buat. Angka ini menunjukkan berapa banyak metrik kustom yang dikenakan biaya. Untuk informasi selengkapnya tentang CloudWatch harga, lihat CloudWatchHarga Amazon.

   a. Untuk mengirim metrik dengan ID halaman sebagai dimensi, pilih Jelajahi untuk ID halaman lalu pilih ID halaman yang akan digunakan.

   b. Untuk mengirim metrik dengan tipe perangkat sebagai dimensi, pilih perangkat Desktop atau Ponsel dan tablet.

   c. Untuk mengirim metrik dengan sistem operasi sebagai dimensi, pilih satu atau beberapa sistem operasi di bawah Sistem operasi.

d. Untuk mengirim metrik dengan jenis browser sebagai dimensi, pilih satu browser atau lebih di bawah Browser.

e. Untuk mengirim metrik dengan geolokasi sebagai dimensi, pilih satu lokasi atau lebih di bawah Lokasi.

   Hanya lokasi tempat monitor aplikasi ini melaporkan metrik yang akan muncul dalam daftar yang akan dipilih.

8. Setelah Anda selesai dengan pilihan Anda, pilih Kirim metrik.
9. (Opsional) Dalam daftar Metrik yang diperluas, untuk membuat alarm yang mengawasi salah satu metrik, pilih Buat alarm di baris metrik tersebut.

   Untuk informasi umum tentang CloudWatch alarm, lihat Menggunakan CloudWatch alarm Amazon. Untuk tutorial untuk mengatur alarm pada metrik diperpanjang CloudWatch RUM, lihatTutorial: membuat metrik yang diperluas dan membuatnya menjadi alarm.

Berhenti mengirim metrik terekstensi

Untuk menggunakan konsol untuk berhenti mengirim metrik terekstensi

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Sinyal Aplikasi, RUM.
3. Pilih Tampilan daftar lalu pilih nama monitor aplikasi yang akan mengirim metrik.
4. Pilih tab Konfigurasi dan kemudian pilih metrik yang diperluas RUM.
5. Pilih satu atau beberapa kombinasi nama dan dimensi metrik untuk berhenti mengirim. Kemudian pilih Tindakan, Hapus.

Tutorial: membuat metrik yang diperluas dan membuatnya menjadi alarm

Tutorial ini menunjukkan cara mengatur metrik yang diperluas untuk dikirim CloudWatch, dan kemudian cara mengatur alarm pada metrik itu. Dalam tutorial ini, Anda membuat metrik yang melacak JavaScript kesalahan pada browser Chrome.

Untuk mengatur metrik yang diperluas ini dan mengatur alarm di atasnya

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Sinyal Aplikasi, RUM.

3. Pilih Tampilan daftar dan kemudian pilih nama monitor aplikasi yang akan mengirim metrik.
4. Pilih tab Konfigurasi dan kemudian pilih metrik yang diperluas RUM.
5. Pilih Kirim metrik.
6. Pilih JS ErrorCount.
7. Di bawah Browser, pilih Chrome.

   Kombinasi JS ErrorCount dan Chrome ini akan mengirimkan satu metrik yang diperluas ke CloudWatch. Metrik menghitung JavaScript kesalahan hanya untuk sesi pengguna yang menggunakan browser Chrome. Nama metrik akan menjadi JsErrorCountdan nama dimensi akan menjadi Browser.
8. Pilih Kirim metrik.
9. Dalam daftar Metrik diperpanjang, pilih Buat alarm di baris yang ditampilkan JsErrorCountdi bawah Nama dan menampilkan Chrome di bawah BrowserName.
10. Di bawah Tentukan metrik dan kondisi, konfirmasikan bahwa nama dan BrowserNamebidang Metrik telah diisi sebelumnya dengan nilai yang benar.
11. Untuk Statistik, pilih statistik yang ingin Anda gunakan untuk alarm. Rata-rata adalah pilihan yang baik untuk jenis metrik penghitungan ini.
12. Untuk Periode, pilih 5 menit.
13. Pada Kondisi, lakukan hal berikut:

    - Pilih Statis.
    - Pilih Lebih Besar untuk menentukan bahwa alarm harus masuk ke status ALARM ketika jumlah kesalahan lebih tinggi dari ambang batas yang akan Anda tentukan.
    - Di bawah dari... , masukkan angka untuk ambang batas alarm. Alarm masuk ke status ALARM ketika jumlah kesalahan selama periode 5 menit melebihi angka ini.
14. (Opsional) Secara default, alarm masuk ke status ALARM segera setelah jumlah kesalahan melebihi angka ambang batas yang Anda tetapkan selama periode 5 menit. Anda dapat mengubah ini secara opsional sehingga alarm masuk ke status ALARM hanya jika jumlah ini terlampaui selama lebih dari satu periode 5 menit.

    Untuk melakukan hal itu, pilih Konfigurasi tambahan dan kemudian untuk Titik data untuk alarm, tentukan berapa periode 5 menit yang perlu memiliki jumlah kesalahan melebihi ambang batas untuk memicu alarm. Sebagai contoh, Anda dapat memilih 2 dari 2 untuk membuat alarm hanya berbunyi saat dua periode 5 menit berturut-turut melampaui ambang batas, atau 2 dari 3 untuk

membuat alarm berbunyi jika dua dari tiga periode 5 menit berturut-turut melampaui ambang batas.

Untuk informasi selengkapnya tentang jenis evaluasi alarm ini, silakan lihat Melakukan evaluasi alarm.

15. Pilih Berikutnya.
16. Untuk Mengonfigurasi tindakan, tentukan apa yang harus terjadi ketika alarm masuk ke status alarm. Untuk menerima notifikasi dengan Amazon SNS, lakukan hal berikut:
    - Pilih Tambahkan notifikasi.
    - Pilih Dalam alarm.
    - Pilih topik SNS yang ada atau buat yang baru. Jika Anda membuat yang baru, tentukan nama untuk itu dan tambahkan setidaknya satu alamat email ke dalamnya.
17. Pilih Berikutnya.
18. Masukkan nama dan deskripsi opsional untuk alarm, lalu pilih Berikutnya.
19. Tinjau detailnya dan pilih Buat alarm.

Perlindungan data dan privasi data dengan CloudWatch RUM

Model tanggung jawab AWS bersama berlaku untuk perlindungan data dan privasi data di Amazon CloudWatch RUM. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS cloud. Anda harus bertanggung jawab untuk memelihara kendali atas konten yang di-hosting di infrastruktur ini. Untuk informasi selengkapnya tentang privasi data, silakan lihat Pertanyaan Umum Privasi Data. Untuk informasi tentang perlindungan data di Eropa, lihat Model Tanggung Jawab AWS Bersama dan posting blog GDPR di Blog AWS Keamanan. Untuk informasi selengkapnya tentang mematuhi persyaratan GDPR, silakan lihat Pusat Peraturan Perlindungan Data Umum (GDPR).

Amazon CloudWatch RUM menghasilkan cuplikan kode untuk Anda sematkan ke situs web atau kode aplikasi web Anda, berdasarkan masukan data pengguna akhir yang ingin Anda kumpulkan. Klien web, yang diunduh dan dikonfigurasi oleh snippet kode, menggunakan cookie (atau teknologi serupa) untuk membantu Anda mengumpulkan data pengguna akhir. Penggunaan cookie (atau teknologi serupa) tunduk pada peraturan privasi data di wilayah hukum tertentu. Sebelum menggunakan Amazon CloudWatch RUM, kami sangat menyarankan agar Anda menilai kewajiban kepatuhan Anda berdasarkan hukum yang berlaku, termasuk persyaratan hukum yang berlaku untuk memberikan pemberitahuan privasi yang memadai secara hukum dan mendapatkan persetujuan

yang diperlukan untuk penggunaan cookie dan pemrosesan (termasuk pengumpulan) data pengguna akhir. Untuk informasi lebih lanjut tentang bagaimana klien web menggunakan cookie (atau teknologi serupa) dan data pengguna akhir apa yang dikumpulkan klien web, lihat dan. Informasi yang dikumpulkan oleh klien web CloudWatch RUM CloudWatch Cookie klien web RUM (atau teknologi serupa)

Kami sangat merekomendasikan agar Anda tidak memasukkan informasi identifikasi yang sensitif, misalnya nomor akun, alamat email, atau informasi pribadi lain Anda atau para pengguna, ke dalam bidang formulir kosong. Data apa pun yang Anda masukkan ke Amazon CloudWatch RUM atau layanan lain mungkin disertakan dalam log diagnostik.

CloudWatch Cookie klien web RUM (atau teknologi serupa)

Klien web CloudWatch RUM mengumpulkan data tertentu tentang sesi pengguna secara default. Anda dapat memilih untuk mengaktifkan cookie agar klien web mengumpulkan ID pengguna dan ID sesi yang bertahan di seluruh pemuatan halaman. ID pengguna secara acak dihasilkan RUM.

Jika cookie ini diaktifkan, RUM dapat menampilkan jenis data berikut saat Anda melihat dasbor RUM untuk monitor aplikasi ini.

- Data agregat berdasarkan ID pengguna, seperti jumlah pengguna unik dan jumlah pengguna berbeda yang mengalami kesalahan.
- Data agregat berdasarkan ID sesi, seperti jumlah sesi dan jumlah sesi yang mengalami kesalahan.
- Perjalanan pengguna, yang merupakan urutan halaman yang disertakan oleh setiap sesi pengguna sampel.

> **Important**
>
> Jika Anda tidak mengaktifkan cookie ini (atau teknologi serupa), klien web masih mencatat informasi tertentu tentang sesi pengguna akhir seperti tipe/versi browser, tipe/versi sistem operasi, jenis perangkat, dan sebagainya. Ini dikumpulkan untuk memberikan wawasan khusus halaman agregat, seperti tanda vital web, tampilan halaman, dan halaman yang mengalami kesalahan. Untuk informasi selengkapnya tentang data yang direkam, silakan lihat Informasi yang dikumpulkan oleh klien web CloudWatch RUM.

Informasi yang dikumpulkan oleh klien web CloudWatch RUM

Bagian ini mendokumentasikan PutRumEventsskema, yang mendefinisikan struktur data yang dapat Anda kumpulkan dari sesi pengguna menggunakan CloudWatch RUM.

PutRumEventsPermintaan mengirimkan struktur data dengan bidang berikut ke CloudWatch RUM.

- ID batch peristiwa RUM ini
- Detail monitor aplikasi, yang mencakup hal berikut:
  - ID monitor aplikasi
  - Versi aplikasi yang dipantau
- Detail monitor aplikasi, yang mencakup hal berikut. Ini dikumpulkan hanya jika monitor aplikasi mengaktifkan cookie.
  - ID pengguna yang dihasilkan klien web
  - ID Sesi
- Susunan peristiwa RUM dalam batch ini.

Skema peristiwa RUM

Struktur setiap peristiwa RUM mencakup bidang-bidang berikut.

- ID peristiwa
- Timestamp
- Jenis peristiwa
- Agen pengguna
- Metadata
- Detail peristiwa RUM

Metadata peristiwa RUM

Metadata mencakup metadata halaman, metadata agen pengguna, metadata geolokasi, dan metadata domain.

Metadata halaman

Metadata halaman meliputi berikut ini:

- ID Halaman
- Judul halaman
- ID halaman induk. – Ini dikumpulkan hanya jika monitor aplikasi mengaktifkan cookie.
- Kedalaman interaksi – Ini dikumpulkan hanya jika monitor aplikasi mengaktifkan cookie.
- Tanda halaman – Anda dapat menambahkan tanda ke peristiwa halaman untuk mengelompokkan halaman bersama–sama. Untuk informasi selengkapnya, lihat Gunakan grup halaman.

Metadata agen pengguna

Metadata agen pengguna meliputi berikut ini:

- Bahasa Browser
- Nama Browser
- Versi browser
- Nama sistem operasi
- Versi Sistem Operasi
- Jenis perangkat
- Jenis platform

Metadata geolokasi

Metadata geolokasi meliputi berikut ini:

- Kode negara
- Kode subdivisi

Metadata domain

Metadata domain mencakup domain URL.

Detail peristiwa RUM

Detail peristiwa mengikuti salah satu jenis skema berikut, tergantung jenis peristiwa.

Peristiwa awal sesi

Peristiwa ini tidak berisi bidang. Ini dikumpulkan hanya jika monitor aplikasi mengaktifkan cookie.

Skema tampilan halaman

Peristiwa tampilan halaman berisi properti berikut. Anda dapat menonaktifkan koleksi tampilan halaman dengan mengonfigurasi klien web. Untuk informasi selengkapnya, lihat dokumentasi klien web CloudWatch RUM.

Nama	Tipe	Deskripsi
ID Halaman	String	ID yang secara unik mewakili halaman ini dalam aplikasi. Secara default, ini adalah jalur URL.
ID halaman induk	String	ID halaman yang digunakan pengguna saat mereka menavigasi ke halaman saat ini. Ini dikumpulkan hanya jika monitor aplikasi mengaktifkan cookie.
Kedalaman interaksi	String	Ini dikumpulkan hanya jika monitor aplikasi mengaktifkan cookie.

JavaScript skema kesalahan

JavaScript peristiwa kesalahan yang dihasilkan oleh agen berisi properti berikut. Klien web mengumpulkan peristiwa ini hanya jika Anda memilih untuk mengumpulkan telemetri kesalahan.

Nama	Tipe	Deskripsi
Jenis kesalahan	String	Nama kesalahan, jika ada. Untuk informasi selengkap nya, silakan lihat Error.prototype.name. Beberapa browser mungkin tidak mendukung jenis kesalahan.
Pesan kesalahan	String	Pesan kesalahan. Untuk informasi selengkapnya, silakan lihat Error.prototype.message. Jika bidang kesalahan tidak ada, ini adalah pesan dari peristiwa kesalahan. Untuk informasi lebih lanjut, lihat ErrorEvent. Pesan kesalahan mungkin tidak konsisten di berbagai browser.

Nama	Tipe	Deskripsi
Jejak tumpukan	String	Jejak tumpukan kesalahan, jika ada, terpotong menjadi 150 karakter. Untuk informasi selengkapnya, silakan lihat Error.prototype.stack. Beberapa browser mungkin tidak mendukung jejak tumpukan.

Skema peristiwa DOM

Peristiwa document object model (DOM) yang dihasilkan agen berisi properti berikut. Peristiwa ini tidak dikumpulkan secara default. Mereka dikumpulkan hanya jika Anda mengaktifkan telemetri interaksi. Untuk informasi selengkapnya, lihat dokumentasi klien web CloudWatch RUM.

Nama	Tipe	Deskripsi
Peristiwa	String	Jenis peristiwa DOM, seperti klik, gulir, atau arahkan kursor. Untuk informasi selengkapnya, silakan lihat Referensi Perintah.
Elemen	String	Jenis elemen DOM
ID Elemen	String	Jika elemen yang menghasilkan peristiwa memiliki ID, properti ini menyimpan ID tersebut. Untuk informasi selengkapnya, silakan lihat Element.id.
CSSLocator	String	Locator CSS digunakan untuk mengidentifikasi elemen DOM.
InteractionId	String	ID unik untuk interaksi antara pengguna dan UI.

Skema peristiwa navigasi

Peristiwa navigasi dikumpulkan hanya jika monitor aplikasi mengaktifkan telemetri performa.

Peristiwa navigasi menggunakan API pengaturan waktu Navigasi Level 1 dan Navigasi Level 2. API Level 2 tidak didukung di semua browser, jadi bidang yang lebih baru ini bersifat opsional.

> **Note**
>
> Metrik stempel waktu didasarkan pada DOM. HighResTimestamp Dengan API Level 2, semua pengaturan waktu secara default relatif terhadap `startTime`. Tetapi untuk Level 1, metrik `navigationStart` dikurangi dari metrik timestamp untuk mendapatkan nilai relatif. Semua nilai timestamp dalam milidetik.

Peristiwa navigasi berisi properti berikut.

Nama	Tipe	Deskripsi	Catatan
initiatorType	String	Merupakan jenis sumber daya yang memulai peristiwa performa.	Nilai: "navigasi" Level 1: "navigasi" Level 2: entryData .initiatorType
navigatio nType	String	Merupakan jenis navigasi. Atribut ini tidak diperlukan.	Nilai: Nilai harus berupa salah satu dari berikut ini: • `navigate` adalah navigasi yang dimulai dengan memilih tautan, memasukkan URL

Nama	Tipe	Deskripsi	Catatan
			di bilah alamat browser, pengirima n formulir, atau menginisi alisasi melalui operasi skrip selain `reload` atau `back_forw ard` . • `reload` adalah navigasi melalui operasi muat ulang browser atau `location. reload()` . • `back_forw ard` adalah navigasi melalui operasi penelusur

Nama	Tipe	Deskripsi	Catatan
			an riwayat browser. • `prerender` adalah navigasi yang diprakarsai petunjuk prerender. Untuk informasi selengkapnya, silakan lihat Prerender.
startTime	Nomor	Menunjukkan kapan peristiwa dipicu.	Nilai: 0 Level 1: entryData.navigationStart - entryData.navigationStart Level 2: entryData.startTime

Nama	Tipe	Deskripsi	Catatan
unloadEventStart	Angka	Menunjukkan waktu ketika dokumen sebelumnya di jendela mulai dibongkar setelah peristiwa `unload` dilemparkan.	Nilai: Jika tidak ada dokumen sebelumnya atau jika dokumen sebelumnya atau salah satu pengalihan yang diperlukan tidak berasal dari asal yang sama, nilai yang dikembalikan adalah 0. Level 1: `entryData.unloadEventStart > 0 ? entryData.unloadEventStart - entryData.navigationStart : 0` Level 2: EntryData.

Nama	Tipe	Deskripsi	Catatan
			unloadEvent Start

Nama	Tipe	Deskripsi	Catatan
promptForUnload	Angka	Waktu yang dibutuhkan untuk membongkar dokumen. Dengan kata lain, waktu antara `unloadEventStart` dan `unloadEventEnd`. `UnloadEventEnd` mewakili momen dalam milidetik ketika penanganan peristiwa pembongkaran selesai.	Nilai: Jika tidak ada dokumen sebelumnya atau jika dokumen sebelumnya atau salah satu pengalihan yang diperlukan tidak berasal dari asal yang sama, nilai yang dikembalikan adalah 0. Level 1: EntryData.unloadEventEnd - EntryData.unloadEventStart Level 2: EntryData.unloadEventEnd - EntryData.unloadEventStart

Nama	Tipe	Deskripsi	Catatan
redirectCount	Nomor	Angka yang mewakili jumlah pengalihan sejak navigasi non-pengalihan terakhir di bawah konteks penjelajahan saat ini. Atribut ini tidak diperlukan.	Nilai: Jika tidak ada pengalihan atau jika ada pengalihan yang tidak memiliki asal yang sama dengan dokumen tujuan, nilai yang dikembalikan adalah 0. Level 1: Tidak tersedia Level 2: entryData .redirect Count

Nama	Tipe	Deskripsi	Catatan
redirectStart	Nomor	Waktu ketika pengalihan HTTP pertama dimulai.	Nilai: Jika tidak ada pengalihan atau jika ada pengalihan yang tidak memiliki asal yang sama dengan dokumen tujuan, nilai yang dikembalikan adalah 0. Level 1: `entryData.redirectStart > 0 ? entryData.redirectStart - entryData.navigationStart : 0` Level 2: entryData .redirectStart

Nama	Tipe	Deskripsi	Catatan
redirectTime	Nomor	Waktu yang dibutuhkan untuk pengalihan HTTP. Inilah perbedaan antara `redirectStart` dan `redirectEnd` .	Level 1: : entryData .redirectEnd - entryData .redirectStart Level 2: : entryData .redirectEnd - entryData .redirectStart

Nama	Tipe	Deskripsi	Catatan
workerStart	Nomor	Ini adalah properti antarmuka `PerformanceResourceTiming` . Ini menandai awal operasi thread pekerja. Atribut ini tidak diperlukan.	Nilai: Jika thread Service Worker sudah berjalan, atau segera sebelum memulai thread Service Worker, properti ini mengembalikan waktu segera sebelum pengiriman `FetchEvent` . Ini mengembalikan 0 jika sumber daya tidak dicegat Service Worker. Level 1: Tidak tersedia Level 2: entryData .workerStart

Nama	Tipe	Deskripsi	Catatan
workerTime	Nomor	Jika sumber daya dicegat oleh Service Worker, ini mengembalikan waktu yang diperlukan untuk operasi thread pekerja. Atribut ini tidak diperlukan.	Level 1: Tidak tersedia Level 2: `entryData.workerStart > 0 ? entryData.fetchStart - entryData.workerStart : 0`
fetchStart	Nomor	Waktu ketika browser siap untuk mengambil dokumen menggunakan permintaan HTTP. Ini sebelum memeriksa cache aplikasi apa pun.	Level 1: `: entryData.fetchStart > 0 ? entryData.fetchStart - entryData.navigationStart : 0` Level 2: entryData .fetchStart

Nama	Tipe	Deskripsi	Catatan
domainLookupStart	Angka	Waktu ketika pencarian domain dimulai.	Nilai: Jika koneksi persisten digunakan atau jika informasi disimpan dalam cache atau sumber daya lokal, nilai akan sama dengan `fetchStart`. Level 1: `entryData.domainLookupStart > 0 ? entryData.domainLookupStart - entryData.navigationStart : 0` Level 2: EntryData.domainLookupStart

Nama	Tipe	Deskripsi	Catatan
dns	Nomor	Waktu yang dibutuhkan untuk pencarian domain.	Nilai: Jika sumber daya dan catatan DNS di-cache, nilai yang diharapkan adalah 0. Level 1: EntryData.domainLookupEnd - EntryData.domainLookupStart Level 2: EntryData.domainLookupEnd - EntryData.domainLookupStart
nextHopProtocol	String	Sebuah string yang mewakili protokol jaringan yang digunakan untuk mengambil sumber daya. Atribut ini tidak diperlukan.	Level 1: Tidak tersedia Level 2: EntryData.nextHopProtocol

Nama	Tipe	Deskripsi	Catatan
connectStart	Nomor	Waktu segera sebelum agen pengguna mulai membuat koneksi ke server untuk mengambil dokumen.	Nilai: Jika koneksi persisten RFC2616 digunakan , atau jika dokumen saat ini diambil dari cache aplikasi yang relevan atau sumber daya lokal, atribut ini mengembalikan nilai `domainLookupEnd` . Level 1: `entryData.connectStart > 0 ? entryData.connectStart - entryData.navigationStart : 0`

Nama	Tipe	Deskripsi	Catatan
			Level 2: entryData .connectStart
menghubungkan	Nomor	Mengukur waktu yang diperlukan untuk membuat koneksi transportasi atau untuk melakukan autentikasi SSL. Ini juga termasuk waktu yang diblokir yang diambil ketika ada terlalu banyak permintaan bersamaan yang dikeluarkan oleh browser.	Level 1: entryData .connectEnd - entryData .connectStart Level 2: entryData .connectEnd - entryData .connectStart
secureConnectionStart	Angka	Jika skema URL halaman saat ini adalah "https", atribut ini mengembalikan waktu segera sebelum agen pengguna memulai proses jabat tangan untuk mengamankan koneksi saat ini. Ini mengembalikan 0 jika HTTPS tidak digunakan. Untuk informasi selengkapnya tentang skema URL, silakan lihat representasi URL.	Rumus: EntryData. secureConnectionStart

Nama	Tipe	Deskripsi	Catatan
tlsTime	Nomor	Waktu yang dibutuhkan untuk menyelesaikan jabat tangan SSL.	Level 1: `entryData.secureConnectionStart > 0 ? entryData.connectEnd - entryData.secureConnectionStart : 0` Level 2: `entryData.secureConnectionStart > 0 ? entryData.connectEnd - entryData.secureConnectionStart : 0`

Nama	Tipe	Deskripsi	Catatan
requestStart	Nomor	Waktu segera sebelum agen pengguna mulai meminta sumber daya dari server, atau dari cache aplikasi yang relevan, atau dari sumber daya lokal.	Level 1: `: entryData.requestStart > 0 ? entryData.requestStart - entryData.navigationStart : 0` Level 2: entryData .requestStart
timeToFirstByte	Angka	Waktu yang dibutuhkan untuk menerima byte pertama informasi setelah permintaan dibuat. Waktu ini relatif terhadap `startTime` .	Level 1: entryData .responseStart - entryData .requestStart Level 2: entryData .responseStart - entryData .requestStart

Nama	Tipe	Deskripsi	Catatan
responseStart	Nomor	Waktu segera setelah parser HTTP agen pengguna menerima byte pertama respons dari cache aplikasi yang relevan, atau dari sumber daya lokal, atau dari server.	Level 1: `entryData.responseStart > 0 ? entryData.responseStart - entryData.navigationStart : 0` Level 2: entryData.responseStart

Nama	Tipe	Deskripsi	Catatan
responseTime	String	Waktu yang dibutuhkan untuk menerima respons lengkap dalam bentuk byte dari cache aplikasi yang relevan, atau dari sumber daya lokal, atau dari server.	Level 1: `entryData.responseStart > 0 ? entryData.responseEnd - entryData.responseStart : 0` Level 2: `entryData.responseStart > 0 ? entryData.responseEnd - entryData.responseStart : 0`

Nama	Tipe	Deskripsi	Catatan
domIntera ctive	Nomor	Waktu ketika parser selesai bekerja pada dokumen utama, dan HTML DOM dibangun. Pada saat ini, perubahan `Document. readyState` ini menjadi "interaktif" dan peristiwa `readystatechange` yang sesuai dilemparkan.	Level 1: `entryData .domInter active > 0 ? entryData .domInter active - entryData .navigati onStart : 0` Level 2: entryData .domInter active

Nama	Tipe	Deskripsi	Catatan
domContentLoadedEventStart	Angka	Merupakan nilai waktu yang sama dengan waktu segera sebelum agen pengguna mengaktifkan ContentLoaded peristiwa DOM pada dokumen saat ini. ContentLoaded Peristiwa TheDOM diaktifkan ketika dokumen HTML awal telah dimuat dan diurai sepenuhnya. Pada saat ini, dokumen HTML utama telah selesai diuraikan, browser mulai membangun pohon render, dan subsumber daya masih harus dimuat. Ini tidak menunggu style sheet, citra, dan subframe untuk menyelesaikan pemuatan.	Level 1: `entryData.domContentLoadedEventStart > 0 ? entryData.domContentLoadedEventStart - entryData.navigationStart : 0` Level 2: EntryData.domContentLoadedEventStart

Nama	Tipe	Deskripsi	Catatan
domContentLoaded	Angka	Waktu mulai dan akhir konstruksi pohon render ini ditandai dengan `domContentLoadedEventStart` dan `domContentLoadedEventEnd`. Ini memungkinkan CloudWatch RUM melacak eksekusi. Properti adalah perbedaan antara `domContentLoadedStart` dan `domContentLoadedEnd`. Selama waktu ini, DOM dan CSSOM sudah siap. Properti ini menunggu eksekusi skrip, kecuali skrip asinkron dan dibuat secara dinamis. Jika skrip bergantung pada style sheet, `domContentLoaded` menunggu di style sheet juga. Ini tidak menunggu pada gambar. **Note** Nilai sebenarnya dari `domContentLoadedStart` dan `domContentLoadedEnd` mendekati `domContentLoaded` di panel Jaringan Google Chrome. Ini menunjukkan waktu konstruksi pohon render HTML DOM + CSSOM dari awal proses pemuatan halaman. Dalam kasus metrik navigasi, nilai `domContentLoaded` mewakili perbedaan antara nilai awal dan akhir, yang merupakan waktu yang diperlukan untuk mengunduh subsumber daya dan konstruksi pohon render saja.	Level 2: EntryData.domContentLoadedEventEnd - EntryData.domContentLoadedEventStart Level 2: EntryData.domContentLoadedEventEnd - EntryData.domContentLoadedEventStart

Nama	Tipe	Deskripsi	Catatan
domComplete	Nomor	Waktu segera sebelum browser mengatur kesiapan dokumen saat ini dari dokumen saat ini untuk diselesaikan. Di titik ini, pemuatan subsumber daya, seperti gambar, selesai. Ini termasuk waktu yang dibutuhkan untuk mengunduh konten pemblokiran seperti CSS dan sinkron. JavaScript Ini mendekati `loadTime` di panel Jaringan Google Chrome.	Level 1: `entryData.domComplete > 0 ? entryData.domComplete - entryData.navigationStart : 0` Level 2: entryData .domComplete
domProcessingTime	Angka	Total waktu antara respons dan acara pemuatan dimulai.	Level 1: EntryData . loadEventStart - EntryData .responseEnd Level 2: EntryData . loadEventStart - EntryData .responseEnd

Nama	Tipe	Deskripsi	Catatan
loadEvent Start	Angka	Waktu segera sebelum peristiwa `load` dari dokumen saat ini terpicu.	Level 1: `entryData .loadEventStart > 0 ? entryData.loadEventStart - entryData.navigationStart : 0` Level 2: EntryData . loadEvent Start
loadEvent Time	Angka	Perbedaan antara `loadEventStart` dan `loadEventEnd` . Fungsi atau logika tambahan yang menunggu peristiwa pemuatan ini akan diaktifkan selama waktu ini.	Level 1: EntryData . loadEvent End - EntryData . loadEvent Start Level 2: EntryData . loadEvent End - EntryData . loadEvent Start

Nama	Tipe	Deskripsi	Catatan
durasi	String	Durasi adalah total waktu muat halaman. Ini mencatat waktu untuk mengunduh halaman utama dan semua subsumber daya sinkron, dan juga merender halaman. Sumber daya asinkron seperti skrip terus diunduh nanti. Ini adalah perbedaan antara properti `loadEventEnd` dan properti `startTime` .	Level 1: EntryData . loadEventEnd - EntryData .navigationStart Level 2: entryData .duration
headerSize	Nomor	Mengembalikan perbedaan antara `transferSize` dan `encodedBodySize` . Atribut ini tidak diperlukan.	Level 1: Tidak tersedia Level 2: EntryData .transferSize - EntryData . encodedBodySize Level 2: EntryData .transferSize - EntryData . encodedBodySize

Nama	Tipe	Deskripsi	Catatan
compressionRatio	Nomor	Rasio `encodedBodySize` dan `decodedBodySize`. Nilai `encodedBodySize` adalah ukuran terkompresi dari sumber daya tidak termasuk header HTTP. Nilai `decodedBodySize` adalah ukuran sumber daya yang didekompresi tidak termasuk header HTTP. Atribut ini tidak diperlukan.	Level 1: Tidak tersedia. Level 2: `entryData.encodedBodySize > 0 ? entryData.decodedBodySize / entryData.encodedBodySize : 0`
navigationTimingLevel	Angka	Versi API waktu navigasi.	Nilai: 1 atau 2

Skema peristiwa sumber daya

Peristiwa sumber daya dikumpulkan hanya jika monitor aplikasi mengaktifkan telemetri performa.

Metrik stempel waktu didasarkan pada The DOM typedef. HighResTimeStamp Dengan API Level 2, secara default semua pengaturan waktu relatif terhadap `startTime`. Tetapi untuk API Level 1, metrik `navigationStart` dikurangi dari metrik stempel waktu untuk mendapatkan nilai relatif. Semua nilai timestamp dalam milidetik.

Peristiwa sumber daya yang dihasilkan oleh agen berisi properti berikut.

Nama	Tipe	Deskripsi	Catatan
targetUrl	String	Mengembalikan URL sumber daya.	Formula: entryData .name
initiatorType	String	Merupakan jenis sumber daya yang memulai peristiwa sumber daya performa.	Nilai: "sumber daya" Formula: entryData .initiatorType
durasi	String	Mengembalikan perbedaan antara properti `responseEnd` dan properti `startTime` . Atribut ini tidak diperlukan.	Formula: entryData .duration
transferSize	Nomor	Mengembalikan ukuran (dalam oktet) sumber daya yang diambil, termasuk bidang header respons dan tubuh muat respons. Atribut ini tidak diperlukan.	Formula: entryData .transferSize
fileType	String	Ekstensi berasal dari pola URL target.	

Skema peristiwa catatan terbesar yang memuaskan

Peristiwa catatan terbesar yang memuaskan berisi properti berikut.

Peristiwa ini dikumpulkan hanya jika monitor aplikasi mengaktifkan telemetri performa.

Nama	Penjelasan		
Nilai	Untuk informasi selengkapnya, silakan lihat Web Vitals.		

Peristiwa penundaan input pertama

Peristiwa penundaan input pertama berisi properti berikut.

Peristiwa ini dikumpulkan hanya jika monitor aplikasi mengaktifkan telemetri performa.

Nama	Penjelasan		
Nilai	Untuk informasi selengkapnya, silakan lihat Web Vitals.		

Peristiwa pergeseran tata letak kumulatif

Peristiwa pergeseran tata letak kumulatif berisi properti berikut.

Peristiwa ini dikumpulkan hanya jika monitor aplikasi mengaktifkan telemetri performa.

Nama	Penjelasan		
Nilai	Untuk informasi selengkapnya, silakan lihat Web Vitals.		

Peristiwa HTTP

Peristiwa HTTP dapat berisi properti berikut. Ini akan berisi bidang `Response` atau bidang `Error`, tetapi tidak keduanya.

Peristiwa ini dikumpulkan hanya jika monitor aplikasi mengaktifkan telemetri HTTP.

Nama	Penjelasan
Permintaan	Kolom permintaan mencakup hal berikut ini: • Bidang `Method`, yang dapat memiliki nilai-nilai seperti GET, POST, dan sebagainya.

Nama	Penjelasan
	• URL
Respons	Bidang respons mencakup hal berikut ini: • Status, seperti 2xx, 4xx, atau 5xx • Teks status
Kesalahan	Bidang kesalahan mencakup berikut ini: • Tipe • Pesan • Nama file • Nomor baris • Nomor kolom • Jejak tumpukan

Skema peristiwa jejak X-Ray

Peristiwa ini dikumpulkan hanya jika monitor aplikasi mengaktifkan pelacakan X-Ray.

Untuk informasi tentang skema peristiwa jejak X-Ray, silakan lihat dokumen segmen AWS X-Ray.

Waktu perubahan rute untuk aplikasi satu halaman

Dalam aplikasi multi-halaman tradisional, ketika pengguna meminta konten baru untuk dimuat, pengguna sebenarnya meminta halaman HTML baru dari server. Akibatnya, klien web CloudWatch RUM menangkap waktu muat menggunakan metrik API kinerja reguler.

Namun, aplikasi web satu halaman menggunakan JavaScript dan Ajax untuk memperbarui antarmuka tanpa memuat halaman baru dari server. Pembaruan satu halaman tidak direkam oleh API waktu browser, alih-alih menggunakan waktu perubahan rute.

CloudWatch RUM mendukung pemantauan beban halaman penuh dari server dan pembaruan satu halaman, dengan perbedaan berikut:

- Untuk waktu perubahan rute, tidak ada metrik yang disediakan browser seperti `tlsTime`, `timeToFirstByte`, dan sebagainya.

- Untuk waktu perubahan rute, bidang `initiatorType` adalah `route_change`.

Klien web CloudWatch RUM mendengarkan interaksi pengguna yang dapat menyebabkan perubahan rute, dan ketika interaksi pengguna tersebut direkam, klien web mencatat stempel waktu. Kemudian waktu perubahan rute akan dimulai jika kedua hal berikut benar:

- API riwayat browser (kecuali tombol maju dan mundur browser) digunakan untuk melakukan perubahan rute.
- Perbedaan antara waktu deteksi perubahan rute dan timestamp interaksi pengguna terbaru kurang dari 1000 ms. Ini menghindari penyimpangan data.

Kemudian, setelah waktu perubahan rute dimulai, waktu tersebut selesai jika tidak ada permintaan AJAX dan mutasi DOM yang sedang berlangsung. Kemudian timestamp dari aktivitas yang selesai terakhir akan digunakan sebagai timestamp penyelesaian.

Waktu perubahan rute akan habis jika ada permintaan AJAX yang sedang berlangsung atau mutasi DOM selama lebih dari 10 detik (secara default). Dalam hal ini, klien web CloudWatch RUM tidak akan lagi mencatat waktu untuk perubahan rute ini.

Akibatnya, durasi peristiwa perubahan rute dihitung sebagai berikut:

```
(time of latest completed activity) - (latest user interaction timestamp)
```

Kelola aplikasi Anda yang menggunakan CloudWatch RUM

Gunakan langkah-langkah di bagian ini untuk mengelola penggunaan RUM aplikasi Anda. CloudWatch

Bagaimana cara saya menemukan potongan kode yang sudah saya buat?

Untuk menemukan cuplikan kode CloudWatch RUM yang telah Anda buat untuk aplikasi, ikuti langkah-langkah ini.

Untuk menemukan potongan kode yang telah Anda buat

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Sinyal Aplikasi, RUM.
3. Pilih Tampilan daftar.

4. Di samping nama monitor aplikasi, pilih Lihat JavaScript.
5. Di panel JavaScript Snippet, pilih Salin ke clipboard.

Sunting Aplikasi Anda

Untuk mengubah pengaturan monitor aplikasi, ikuti langkah-langkah berikut. Anda dapat mengubah pengaturan apa pun kecuali nama monitor aplikasi.

Untuk mengedit bagaimana aplikasi Anda menggunakan CloudWatch RUM

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Sinyal Aplikasi, RUM.
3. Pilih Tampilan daftar.
4. Pilih tombol di sebelah nama aplikasi, lalu pilih Tindakan, Sunting.
5. Ubah pengaturan apa pun kecuali nama aplikasi. Untuk informasi selengkapnya tentang pengaturan, silakan lihat Langkah 2: Membuat monitor aplikasi.
6. Setelah selesai, silakan pilih Simpan.

   Mengubah pengaturan mengubah potongan kode. Anda sekarang harus menempelkan potongan kode yang diperbarui ke dalam aplikasi Anda.
7. Setelah cuplikan JavaScript kode dibuat, pilih Salin ke clipboard atau Unduh, lalu pilih Selesai.

   Untuk memulai pemantauan dengan pengaturan baru, Anda memasukkan potongan kode ke dalam aplikasi Anda. Masukkan potongan kode di dalam elemen `<head>` aplikasi Anda, sebelum elemen `<body>` atau tanda `<script>` lainnya.

Berhenti menggunakan CloudWatch RUM atau hapus monitor aplikasi

Untuk berhenti menggunakan CloudWatch RUM dengan aplikasi, hapus cuplikan kode yang dihasilkan RUM dari kode aplikasi Anda.

Untuk menghapus monitor aplikasi RUM, ikuti langkah-langkah ini.

Untuk menghapus monitor aplikasi

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Sinyal Aplikasi, RUM.

3. Pilih Tampilan daftar.
4. Pilih tombol di sebelah nama aplikasi, lalu pilih Tindakan, Hapus.
5. Di kotak konfirmasi, masukkan **`Delete`** lalu pilih Hapus.
6. Jika Anda belum melakukannya, hapus cuplikan kode CloudWatch RUM dari kode aplikasi Anda.

CloudWatch Kuota RUM

CloudWatch RUM memiliki kuota berikut.

Sumber daya	Kuota bawaan
Monitor Aplikasi	20 per akun Anda dapat meminta penambahan kuota.
Tingkat penyerapan RUM	50 PutRumEventspermintaan per detik (TPS). Anda dapat meminta penambahan kuota.

Pemecahan Masalah CloudWatch RUM

Bagian ini berisi tips untuk membantu Anda memecahkan masalah CloudWatch RUM.

Tidak ada data untuk aplikasi saya

Pertama, pastikan bahwa potongan kode telah dimasukkan dengan benar ke dalam aplikasi Anda. Untuk informasi selengkapnya, lihat Langkah 4: Masukkan potongan kode ke dalam aplikasi Anda.

Jika bukan itu masalahnya, maka mungkin belum ada lalu lintas ke aplikasi Anda. Hasilkan beberapa lalu lintas dengan mengakses aplikasi Anda dengan cara yang sama seperti yang dilakukan pengguna.

Data telah berhenti direkam untuk aplikasi saya

Aplikasi Anda mungkin telah diperbarui dan sekarang tidak lagi berisi cuplikan kode CloudWatch RUM. Memeriksa kode aplikasi Anda.

Kemungkinan lain adalah seseorang mungkin telah memperbarui potongan kode tetapi kemudian tidak memasukkan potongan yang diperbarui ke dalam aplikasi. Temukan potongan kode yang benar

saat ini dengan mengikuti petunjuk di Bagaimana cara saya menemukan potongan kode yang sudah saya buat? dan bandingkan dengan potongan kode yang ditempelkan ke aplikasi Anda.

Pemantauan jaringan

Topik di bagian ini menjelaskan kemampuan pemantauan CloudWatch jaringan dan internet yang disediakan oleh Amazon CloudWatch Internet Monitor dan Amazon CloudWatch Network Monitor. Layanan ini membantu Anda mendapatkan visibilitas operasional ke jaringan dan kinerja internet serta ketersediaan aplikasi Anda yang di-host. AWS

- Internet Monitor menggunakan data konektivitas yang AWS menangkap dari jejak jaringan globalnya untuk menghitung dasar kinerja dan ketersediaan untuk lalu lintas yang menghadap ke internet. Anda dapat melihat pandangan global tentang pola lalu lintas dan peristiwa kesehatan, dan dengan mudah menelusuri informasi tentang peristiwa. Anda juga bisa mendapatkan peringatan untuk acara kesehatan internet yang memengaruhi klien aplikasi Anda. Selain itu, Anda dapat menggunakan wawasan yang disediakan Internet Monitor untuk mengeksplorasi potensi peningkatan pengalaman klien Anda, dengan menggunakan Amazon CloudFront atau perutean melalui yang berbeda. Wilayah AWS
- Network Monitor menggunakan pendekatan agen yang dikelola sepenuhnya untuk memungkinkan Anda melacak dan memvisualisasikan latensi dan kehilangan paket untuk koneksi jaringan hybrid. Untuk mengumpulkan pengukuran dan mengaktifkan Network Monitor untuk membuat peringatan peristiwa kesehatan untuk aplikasi Anda, Anda membuat probe yang dikirim dari sumber daya yang dihosting AWS ke alamat IP tujuan lokal. Anda tidak perlu menginstal agen tambahan untuk memantau kinerja jaringan Anda. Seperti halnya Internet Monitor, Anda dapat mengatur peringatan dan ambang batas, mendapatkan informasi untuk membantu Anda memecahkan masalah dengan cepat, dan kemudian mengambil tindakan untuk meningkatkan pengalaman pengguna akhir Anda.

Topik

Menggunakan Amazon CloudWatch Internet Monitor

Amazon CloudWatch Internet Monitor memberikan visibilitas tentang bagaimana masalah internet memengaruhi kinerja dan ketersediaan antara aplikasi yang di-host AWS dan pengguna akhir Anda. Aplikasi ini dapat mengurangi waktu yang Anda butuhkan untuk mendiagnosis masalah internet dari beberapa hari menjadi beberapa menit. Internet Monitor menggunakan data konektivitas yang AWS menangkap dari jejak jaringan globalnya untuk menghitung dasar kinerja dan ketersediaan

untuk lalu lintas yang menghadap ke internet. Ini adalah data yang sama yang AWS digunakan untuk memantau uptime dan ketersediaan internet. Dengan ukuran tersebut sebagai dasar, Monitor Internet meningkatkan kesadaran untuk Anda ketika ada masalah signifikan bagi pengguna akhir (klien) Anda di lokasi geografis yang berbeda di mana aplikasi Anda berjalan.

Di CloudWatch konsol Amazon, Anda dapat melihat pandangan global tentang pola lalu lintas dan peristiwa kesehatan, dan dengan mudah menelusuri informasi tentang peristiwa, di perincian geografis yang berbeda (lokasi). Anda dapat dengan jelas memvisualisasikan dampak, dan mengetahui lokasi dan jaringan klien (ASN, umumnya penyedia layanan internet atau ISP) yang terpengaruh. Jika Internet Monitor menentukan bahwa ketersediaan internet atau masalah kinerja disebabkan oleh ASN tertentu atau oleh AWS jaringan, itu memberikan informasi tersebut.

Fitur-fitur utama dari Monitor Internet

- Monitor Internet menyarankan wawasan dan rekomendasi yang dapat membantu Anda meningkatkan pengalaman pengguna akhir Anda. Anda dapat menjelajahi, dalam waktu dekat, cara meningkatkan latensi yang diproyeksikan aplikasi Anda dengan beralih menggunakan layanan lain, atau dengan mengalihkan lalu lintas ke beban kerja Anda melalui yang berbeda. Wilayah AWS
- Dengan Monitor Internet, Anda dapat dengan cepat mengidentifikasi apa yang memengaruhi performa dan ketersediaan aplikasi Anda, sehingga Anda dapat melacak dan mengatasi masalah.
- Internet Monitor menerbitkan pengukuran internet ke CloudWatch Log dan CloudWatch Metrik, untuk mendukung penggunaan CloudWatch alat dengan informasi kesehatan untuk lokasi dan ASN (penyedia layanan internet) khusus untuk aplikasi Anda. Secara opsional, Anda juga dapat menerbitkan pengukuran internet ke Amazon S3.
- Internet Monitor mengirimkan acara kesehatan ke Amazon EventBridge sehingga Anda dapat mengatur notifikasi. Jika masalah disebabkan oleh AWS jaringan, Anda juga secara otomatis menerima AWS Health Dashboard pemberitahuan dengan langkah-langkah yang AWS diambil untuk mengurangi masalah.

Cara menggunakan Monitor Internet

Untuk menggunakan Internet Monitor, Anda membuat monitor dan mengaitkan sumber daya aplikasi Anda dengannya—VPC, Network Load Balancers, CloudFront distribusi, atau WorkSpaces direktori—untuk memungkinkan Internet Monitor mengetahui di mana lalu lintas yang menghadap ke internet aplikasi Anda berada. Internet Monitor kemudian menerbitkan pengukuran internet dari AWS yang spesifik untuk jaringan kota, yaitu, lokasi klien dan ASN (biasanya penyedia layanan internet atau ISP), di mana klien mengakses aplikasi Anda. Untuk informasi selengkapnya, lihat Bagaimana

Amazon CloudWatch Internet Monitor bekerja. Untuk mulai bekerja dengan Monitor Internet, silakan lihat Memulai dengan Amazon CloudWatch Internet Monitor menggunakan konsol.

Daftar Isi

Didukung Wilayah AWS untuk Amazon CloudWatch Internet Monitor

Wilayah AWS Tempat Amazon CloudWatch Internet Monitor didukung tercantum di bagian ini. Untuk daftar Wilayah saat ini yang didukung oleh Internet Monitor, termasuk Wilayah keikutsertaan, lihat titik akhir dan kuota Amazon CloudWatch Internet Monitor di Referensi Umum Amazon Web Services.

Perhatikan bahwa Internet Monitor menyimpan data untuk monitor hanya Wilayah AWS di mana Anda membuat monitor, meskipun monitor dapat menyertakan sumber daya di beberapa Wilayah.

Nama wilayah (Dukungan keikutsertaan)	Wilayah
Afrika (Cape Town)	`af-south-1`

Nama wilayah (Dukungan keikutsertaan)	Wilayah
Asia Pasifik (Hong Kong)	`ap-east-1`
Asia Pasifik (Hyderabad)	`ap-south-2`
Asia Pasifik (Jakarta)	`ap-southeast-3`
Asia Pasifik (Melbourne)	`ap-southeast-4`
Eropa (Milan)	`eu-south-1`
Eropa (Spanyol)	`eu-south-2`
Eropa (Zürich)	`eu-central-2`
Timur Tengah (Bahrain)	`me-south-1`
Timur Tengah (UEA)	`me-central-1`

Nama wilayah (Dukungan default)	Wilayah
AS Timur (Ohio)	`us-east-2`
AS Timur (Virginia Utara)	`us-east-1`
AS Barat (California Utara)	`us-west-1`
AS Barat (Oregon)	`us-west-2`
Asia Pasifik (Mumbai)	`ap-south-1`
Asia Pasifik (Osaka)	`ap-northeast-3`
Asia Pasifik (Seoul)	`ap-northeast-2`
Asia Pasifik (Singapura)	`ap-southeast-1`
Asia Pasifik (Sydney)	`ap-southeast-2`

Nama wilayah (Dukungan default)	Wilayah
Asia Pasifik (Tokyo)	`ap-northeast-1`
Kanada (Pusat)	`ca-central-1`
Eropa (Frankfurt)	`eu-central-1`
Eropa (Irlandia)	`eu-west-1`
Eropa (London)	`eu-west-2`
Eropa (Paris)	`eu-west-3`
Eropa (Stockholm)	`eu-north-1`
Amerika Selatan (Sao Paulo)	`sa-east-1`

Harga untuk Amazon CloudWatch Internet Monitor

Dengan Amazon CloudWatch Internet Monitor, tidak ada biaya di muka atau komitmen jangka panjang. Penetapan Harga untuk Monitor Internet memiliki dua komponen: biaya sumber daya per pemantauan dan biaya per jaringan kota. Sebuah jaringan kota adalah lokasi tempat klien mengakses sumber daya aplikasi Anda dan lokasi jaringan (ASN, seperti penyedia layanan internet atau ISP) di mana klien mengakses sumber daya.

Anda memilih sebuah persentase lalu lintas untuk dipantau ketika Anda membuat sebuah monitor. Untuk membantu mengontrol tagihan Anda, Anda juga bisa menetapkan sebuah batasan untuk jumlah maksimum jaringan kota untuk dipantau. Anda dapat memperbarui persentase lalu lintas untuk dipantau atau batasan maksimum jaringan kota kapan saja dengan mengubah monitor Anda. 100 jaringan kota pertama (di semua monitor per akun) disertakan. Setelah itu, Anda cukup membayar jumlah tambahan jaringan kota yang sebenarnya yang Anda pantau, hingga jumlah maksimum.

Anda cukup membayar jumlah tambahan jaringan kota yang sebenarnya yang Anda pantau, hingga jumlah maksimum, dengan bebas biaya untuk 100 jaringan kota pertama (di seluruh monitor per akun). Jumlah pembayaran yang sama diterapkan untuk biaya 100 jaringan kota yang dipotong dari tagihan bulanan Anda.

Misalnya, sebuah perusahaan global yang besar dapat memilih untuk memantau 100% lalu lintas yang dapat diakses di internet, dan menetapkan jaringan kota maksimum 50.000, untuk satu monitor dengan satu sumber daya. Dengan asumsi lalu lintas mencapai 50.000 jaringan kota, porsi tagihan tersebut akan menjadi sekitar 2.700 USD/bulan. Untuk perusahaan lain, di wilayah geografis yang lebih sedikit, dengan satu monitor dengan satu sumber daya dan 200 jaringan kota, bagian dari tagihan ini akan menjadi sekitar 13 USD/bulan. Untuk informasi selengkapnya, lihat Memilih batas maksimum jaringan kota.

Anda dapat mencoba berbagai pilihan dengan kalkulator harga. Untuk menjelajahi opsi harga, pada kalkulator Harga untuk CloudWatch halaman, gulir ke bawah ke Internet Monitor. Untuk informasi selengkapnya tentang harga, lihat halaman CloudWatch harga Amazon.

Komponen dan ketentuan untuk Amazon CloudWatch Internet Monitor

Amazon CloudWatch Internet Monitor menggunakan atau referensi berikut ini.

Monitor

Monitor menyertakan sumber daya untuk satu aplikasi yang ingin Anda lihat performa internetnya dan pengukuran ketersediaannya, dan merupakan tempat Anda untuk mendapatkan pengingat peristiwa kondisi kesehatan. Ketika Anda membuat monitor untuk sebuah aplikasi, tambahkan sumber daya untuk aplikasi tersebut untuk menentukan kota (lokasi) untuk dipantau Monitor Internet. Monitor Internet menggunakan pola lalu lintas dari sumber daya aplikasi yang Anda tambahkan sehingga Monitor Internet dapat menerbitkan performa internet dan pengukuran ketersediaan spesifik untuk lokasi itu saja dan ASN (biasanya, penyedia layanan internet atau ISP) yang berkomunikasi dengan aplikasi Anda. Dengan kata lain, sumber daya yang Anda tambahkan menciptakan cakupan jaringan kota yang Anda inginkan untuk dipantau oleh Monitor Internet dan cakupan yang Anda inginkan untuk dipublikasikannya pengukuran.

Sumber daya yang dipantau

Sumber daya yang Anda tambahkan ke monitor adalah sumber daya yang dipantau di Monitor Internet. Sumber daya tersebut adalah:

- Setiap VPC yang Anda tambahkan di sebuah Wilayah adalah sumber daya yang dipantau. Saat Anda menambahkan VPC, Internet Monitor memantau lalu lintas untuk aplikasi apa pun yang menghadap ke internet di VPC, misalnya, aplikasi yang dihosting di instans Amazon EC2, di belakang Network Load Balancer, atau wadah. AWS Fargate
- Setiap Penyeimbang Beban Jaringan yang Anda tambahkan di sebuah Wilayah adalah sumber daya yang dipantau.

- Setiap WorkSpaces direktori yang Anda tambahkan di Wilayah adalah sumber daya yang dipantau.
- Setiap CloudFront distribusi yang Anda tambahkan adalah sumber daya yang dipantau.

Nomor Sistem Otonom (ASN)

Di Internet Monitor, ASN biasanya mengacu pada penyedia layanan internet (ISP), seperti Verizon atau Comcast. ASN adalah penyedia jaringan yang digunakan klien untuk mengakses aplikasi internet Anda. Sistem Otonom (AS) adalah seperangkat prefiks protokol internet (IP) internet yang dapat di-routing yang dimiliki oleh jaringan atau kumpulan jaringan yang semuanya dikelola, dikendalikan, dan diawasi oleh satu organisasi.

Jaringan kota (lokasi dan ASN)

Sebuah jaringan kota adalah lokasi (misalnya saja sebuah kota) tempat klien mengakses sumber daya aplikasi Anda dan ASN, biasanya sebuah penyedia layanan internet (ISP), di mana klien mengakses sumber daya. Untuk membantu mengontrol tagihan Anda, Anda tetapkan batasan untuk jumlah maksimum jaringan kota agar Monitor Internet dapat memantau setiap monitor. Anda cukup membayar jumlah tambahan jaringan kota yang sebenarnya yang Anda pantau, hingga jumlah maksimum. Untuk informasi selengkapnya, silakan lihat Memilih batas maksimum jaringan kota.

Pengukuran internet

Internet Monitor menerbitkan pengukuran internet ke dalam file CloudWatch log di Log setiap lima menit untuk 500 jaringan kota teratas (lokasi klien dan ASN, biasanya penyedia layanan internet atau ISP) di akun Anda. Pengukuran ini mengukur skor performa, skor ketersediaan, byte tertransfer (byte masuk dan byte keluar), dan tempo pulang-pergi untuk jaringan kota aplikasi Anda. Ini adalah pengukuran untuk jaringan kota khusus untuk VPC, Network Load Balancer, distribusi, CloudFront atau direktori Anda. WorkSpaces Secara opsional, Anda dapat memilih untuk menerbitkan pengukuran dan event internet untuk semua jaringan kota yang dipantau (hingga 500.000 batas layanan jaringan kota) ke bucket Amazon S3.

Metrik

Internet Monitor menghasilkan metrik agregat untuk CloudWatch metrik, untuk lalu lintas global ke aplikasi Anda dan lalu lintas global ke masing-masing metrik. Wilayah AWS Untuk informasi selengkapnya, lihat Menggunakan CloudWatch Metrik dengan Amazon CloudWatch Internet Monitor.

peristiwa kondisi kesehatan

Monitor Internet membuat peristiwa kondisi kesehatan untuk mengingatkan Anda tentang masalah tertentu yang memengaruhi aplikasi Anda. Monitor Internet mendeteksi masalah-masalah internet, seperti peningkatan latensi jaringan, di seluruh dunia. Kemudian menggunakan pengukuran internet historisnya dari seluruh jejak infrastruktur AWS global untuk menghitung dampak masalah saat ini pada aplikasi Anda, dan menciptakan peristiwa kesehatan. Monitor Internet, secara default, membuat peristiwa kondisi kesehatan berdasarkan dampak secara keseluruhan dan ambang batas dampak secara lokal. Untuk informasi selengkapnya tentang mengonfigurasi ambang batas, silakan lihat Mengubah ambang batas peristiwa kondisi kesehatan.

Setiap peristiwa kondisi kesehatan mencakup informasi tentang jaringan kota yang terkena dampak. Anda dapat melihat peristiwa kesehatan di CloudWatch konsol, atau dengan menggunakan AWS SDK atau AWS CLI dengan tindakan Internet Monitor API. Internet Monitor juga mengirimkan EventBridge pemberitahuan Amazon untuk acara kesehatan. Untuk informasi selengkapnya, silakan lihat Kapan Monitor Internet membuat dan menyelesaikan peristiwa kondisi kesehatan.

Ambang batas

Monitor Internet membuat peristiwa kondisi kesehatan berdasarkan ambang batas keseluruhan dan ambang batas secara lokal. Anda dapat mengubah ambang batas default dan mengonfigurasi opsi lain, seperti mematikan ambang batas lokal. Untuk informasi selengkapnya tentang mengonfigurasi ambang batas, silakan lihat Mengubah ambang batas peristiwa kondisi kesehatan.

Skor performa dan ketersediaan

Dengan menganalisis data yang AWS dikumpulkan, Internet Monitor dapat mendeteksi kapan kinerja dan ketersediaan aplikasi Anda menurun, dibandingkan dengan perkiraan baseline yang dihitung oleh Internet Monitor. Untuk mempermudah melihat penurunan itu, Monitor Internet melaporkan informasi kepada Anda dalam bentuk skor. Skor performa mewakili persentase perkiraan lalu lintas yang tidak terdapat penurunan performa. Demikian pula, skor ketersediaan mewakili perkiraan persentase lalu lintas yang tidak terdapat penurunan ketersediaan. Untuk informasi selengkapnya, lihat Cara AWS menghitung skor kinerja dan ketersediaan.

Byte ditransfer dan dipantau byte yang ditransfer

Byte ditransfer adalah jumlah total byte masuk dan keluar lalu lintas antara aplikasi di AWS dan jaringan kota (yaitu, lokasi dan ASN, biasanya penyedia layanan internet) di mana klien

mengakses aplikasi. Byte tertransfer yang terpantau adalah metrik serupa, namun hanya mencakup byte untuk lalu lintas yang dipantau saja.

Tempo pulang-pergi

Waktu pulang-pergi (RTT) adalah berapa lama waktu yang dibutuhkan oleh sebuah permintaan dari pengguna klien untuk mendapat respons bagi pengguna tersebut. Ketika RTT dikumpulkan dari seluruh lokasi klien (kota atau geografis lainnya), nilainya diukur oleh seberapa banyak lalu lintas aplikasi Anda berdasarkan setiap lokasi klien.

Bagaimana Amazon CloudWatch Internet Monitor bekerja

Bagian ini memberikan informasi tentang cara kerja Amazon CloudWatch Internet Monitor. Ini termasuk deskripsi tentang bagaimana AWS mengumpulkan data yang digunakannya untuk membantu mendeteksi masalah konektivitas di internet, dan bagaimana skor kinerja dan ketersediaan dihitung.

Daftar Isi

AWS Keuntungannya

Internet Monitor berfokus pada pemantauan hanya pada bagian dari internet yang diakses oleh pengguna AWS sumber daya Anda, alih-alih memantau situs web Anda secara luas dari setiap Wilayah di dunia seperti alat lain. Monitor Internet juga merupakan solusi yang hemat biaya, terjangkau untuk perusahaan besar dan kecil.

Internet Monitor menggunakan probe kuat dan algoritma pendeteksian masalah yang sama yang AWS memanfaatkan secara internal dan memberi tahu Anda tentang masalah konektivitas yang memengaruhi aplikasi Anda dengan membuat acara kesehatan di Internet Monitor. Monitor Internet kemudian memberikan Anda akses ke performa yang dihasilkan dan peta ketersediaan, dengan menyajikan profil lalu lintas yang dibuatnya dari pemirsa aktif Anda, berdasarkan sumber daya aplikasi Anda.

Dengan menggunakan informasi ini, Monitor Internet menunjukkan kepada Anda event yang relevan saja (yaitu, event dari tempat di mana Anda memiliki pemirsa aktif), dan hanya dampak yang disebabkan event itu saja pada volume pemirsa keseluruhan Anda. Jadi, seberapa besar dampak suatu peristiwa, berdasarkan persentase, didasarkan pada total lalu lintas Anda di seluruh dunia.

Internet Monitor menerbitkan ke CloudWatch Log pengukuran internet setiap lima menit untuk 500 jaringan kota teratas (lokasi klien dan ASN, biasanya penyedia layanan internet atau ISP) yang mengirim lalu lintas ke setiap monitor. Secara opsional, Anda dapat memilih untuk menerbitkan pengukuran internet untuk semua jaringan kota yang dipantau (hingga 500.000 batas layanan jaringan kota) ke bucket Amazon S3. Untuk informasi selengkapnya, lihat Menerbitkan pengukuran internet ke Amazon S3 di Amazon CloudWatch Internet Monitor.

Manfaat Monitor Internet termasuk di bawah ini:

- Menggunakan Monitor Internet tidak akan menambah beban atau biaya tambahan pada aplikasi Anda yang di-hosting di AWS.
- Anda tidak perlu menyertakan kode pengukuran performa di sumber daya sisi klien Anda, atau dalam aplikasi Anda.
- Anda bisa mendapatkan visibilitas pada performa dan ketersediaan di internet yang terhubung dengan aplikasi Anda, termasuk informasi "last mile".

Perhatikan bahwa karena Internet Monitor membuat pengukuran berdasarkan AWS sumber daya Anda, Internet Monitor hanya membuat peristiwa yang spesifik untuk lalu lintas aplikasi Anda. Masalah internet global pada umumnya tidak dilaporkan. Selain itu, ketika lokasi layanan adalah Wilayah AWS, pengukuran dan peristiwa yang dipancarkan dirancang untuk mewakili konektivitas di tingkat Regional dan tidak secara akurat mewakili konektivitas antara lokasi pengguna akhir dan Availability Zone.

Bagaimana AWS mengukur masalah konektivitas

Amazon CloudWatch Internet Monitor menggunakan data konektivitas internet antara CloudFront titik kehadiran (POP) yang berbeda Wilayah AWS dan Amazon ke lokasi yang berbeda melalui

jaringan atau Autonomous System Numbers (ASN), biasanya penyedia layanan internet (ISP). Data konektivitas ini digunakan secara internal oleh operator AWS, setiap hari, untuk secara proaktif mendeteksi masalah konektivitas di seluruh internet global.

Untuk setiap orang Wilayah AWS, kami tahu bagian mana dari internet yang berkomunikasi dengan Wilayah dan melakukan hal berikut:

- Kami secara aktif memantau bagian-bagian internet tersebut, dengan jendela gulir 30 hari.
- Kami menggunakan probe jaringan dan protokol tingkat yang lebih tinggi, termasuk probe inbound dan outbound.

Bagaimana AWS menghitung ketersediaan dan RTT

AWS memiliki probe aktif dan pasif yang mengukur latensi (kinerja) pada persentil ke-90 dan jangkauan (ketersediaan) dari setiap Wilayah AWS dan dari layanan ke seluruh internet. CloudFront Pola abnormal dalam konektivitas antara layanan dan lokasi pelanggan dipantau, dan kemudian dilaporkan sebagai peringatan kepada pelanggan.

Waktu pulang-pergi (RTT) adalah berapa lama waktu yang dibutuhkan oleh sebuah permintaan dari pengguna untuk mendapat respons bagi pengguna tersebut. Ketika tempo pulang-pergi dikumpulkan dari seluruh lokasi pengguna akhir, nilainya ditimbang oleh jumlah lalu lintas Anda yang digerakkan oleh setiap lokasi pengguna akhir.

Sebagai contoh, dengan dua lokasi pengguna akhir, satu melayani 90% lalu lintas dengan RTT 5 ms, dan yang lainnya melayani 10% lalu lintas dengan RTT 10 ms, hasilnya kumpulan RTT sebanyak 5,5 ms (yang berasal dari 5 ms * 0,9 + 10 ms * 0,1).

Perhatikan bahwa ada perbedaan sumber daya tentang pengukuran latensi last mile. Untuk pengukuran latensi Monitor Internet, VPC, Network Load Balancer, dan WorkSpaces direktori tidak menyertakan latensi last-mile.

Cara Monitor Internet menghitung skor performa dan ketersediaan

AWS memiliki data historis yang substansif tentang kinerja dan ketersediaan internet antara AWS layanan dan jaringan kota yang berbeda (lokasi dan ASN). Dengan menerapkan analisis statistik pada data, Monitor Internet dapat mendeteksi kapan performa dan ketersediaan untuk aplikasi Anda telah menurun, dibandingkan dengan baseline perkiraan yang Monitor telah perhitungkan. Untuk mempermudah melihat penurunan, informasi tersebut dilaporkan kepada Anda dalam bentuk skor kesehatan: skor performa dan skor ketersediaan.

Skor Kesehatan dihitung pada pedetail wilayah yang berbeda. Pada pedetail wilayah geografis paling detail, kami mengkomputasikan skor kesehatan untuk wilayah geografis, seperti kota

atau area metro, dan ASN (jaringan kota). Kami juga menggabungkan skor kesehatan individu ke angka skor kesehatan keseluruhan untuk aplikasi yang ada di monitor. Jika Anda melihat skor performa atau ketersediaan tanpa memfilter untuk geografi tertentu atau penyedia layanan tertentu, Monitor Internet Monitor akan memberikan skor kesehatan secara keseluruhan.

Skor kesehatan keseluruhan mencakup seluruh aplikasi Anda untuk periode waktu tertentu. Ketika skor performa atau skor ketersediaan untuk pasangan jaringan kota aplikasi Anda di seluruh aplikasi Anda mencapai ambang batas atau turun di bawah ambang batas peristiwa kondisi kesehatan yang sesuai atas performa atau ketersediaan maka Monitor Internet akan memberikan peringatan peristiwa kondisi kesehatan. Secara bawaan, ambang batas adalah 95% untuk performa dan ketersediaan keseluruhan. Monitor Internet juga membuat peristiwa kondisi kesehatan berdasarkan ambang batas lokal — jika opsi ini diaktifkan, seperti halnya default — berdasarkan nilai yang Anda konfigurasikan. Untuk informasi selengkapnya tentang mengonfigurasi ambang batas peristiwa kondisi kesehatan, silakan lihat Mengubah ambang batas peristiwa kondisi kesehatan.

Saat Anda menjelajahi informasi di monitor dan berkas log untuk menyelidiki masalah dan mempelajari lebih lanjut, Anda dapat memfilter berdasarkan kota (lokasi) tertentu, jaringan (ASN atau penyedia layanan internet), atau keduanya. Jadi, Anda dapat menggunakan filter untuk melihat skor kesehatan untuk berbagai kota, ASN, atau pasangan jaringan kota, tergantung pada filter yang Anda pilih.

- Sebuah skor ketersediaan mewakili persentase perkiraan lalu lintas yang tidak terdapat penurunan ketersediaan. Monitor Internet memperkirakan persentase lalu lintas yang mengalami penurunan dari lalu lintas total yang terlihat dan pengukuran metrik ketersediaan. Misalnya, skor ketersediaan 99% untuk pengguna akhir dan pasangan lokasi layanan setara dengan 1% dari lalu lintas yang mengalami penurunan ketersediaan untuk pasangan itu.
- Sebuah skor performa mewakili persentase lalu lintas yang tidak terdapat penurunan performa. Misalnya, skor performa 99% untuk pengguna akhir dan pasangan lokasi layanan setara dengan 1% dari lalu lintas yang mengalami penurunan performa untuk pasangan itu.

Akurasi geolokasi di Monitor Internet

Untuk informasi lokasi, Internet Monitor menggunakan data geolokasi IP yang disediakan oleh. MaxMind Keakuratan informasi lokasi dalam pengukuran Internet Monitor tergantung pada MaxMind keakuratan data.

Apa yang Monitor Internet masukkan ke dalam perhitungan untuk TTFB dan RTT (latensi)

Time to first byte (TTFB) mengacu pada waktu antara ketika klien membuat permintaan dan ketika menerima byte pertama informasi dari server. AWS perhitungan untuk TTFB mengukur waktu

yang berlalu dari Amazon EC2 atau Amazon CloudFront ke node pengukuran Internet Monitor (termasuk mil terakhir node). Artinya, Internet Monitor mengukur waktu dari pengguna ke Wilayah Amazon EC2 untuk TTFB untuk EC2, dan dari pengguna ke TTFB untuk. CloudFront CloudFront

Untuk waktu pulang pergi (RTT), Monitor Internet mencakup waktu dari jaringan kota (yaitu, lokasi klien dan ASN, biasanya penyedia layanan internet), sebagaimana dipetakan oleh alamat IP publik, ke Wilayah AWS. Ini berarti bahwa Monitor Internet tidak memiliki visibilitas last mile bagi pengguna yang mengakses internet dari belakang gateway atau VPN.

Perhatikan bahwa ada perbedaan sumber daya tentang pengukuran latensi last mile. Untuk pengukuran latensi Monitor Internet, VPC, Network Load Balancer, dan WorkSpaces direktori tidak menyertakan latensi last-mile.

Internet Monitor menyertakan informasi TTFB rata-rata di bagian saran pengoptimalan Lalu lintas pada tab Wawasan lalu lintas di CloudWatch dasbor, untuk membantu Anda mengevaluasi opsi untuk pengaturan yang berbeda untuk aplikasi Anda yang dapat meningkatkan kinerja.

Kapan Monitor Internet membuat dan mengatasi peristiwa kondisi kesehatan

Monitor Internet membuat dan mengatasi peristiwa kondisi kesehatan untuk lalu lintas aplikasi yang Anda pantau berdasarkan ambang batas saat ini yang sudah ditetapkan. Monitor Internet memiliki konfigurasi ambang batas default, dan Anda juga dapat mengatur konfigurasi Anda sendiri untuk ambang batas. Monitor Internet memberitahukan dampak keseluruhan masalah konektivitas yang dimiliki aplikasi Anda, dan dampaknya pada area lokal tempat aplikasi Anda memiliki klien, dan akan membuat peristiwa kondisi kesehatan saat ambang batas terlewati.

Internet Monitor menghitung dampak masalah konektivitas pada lokasi klien berdasarkan data historis tentang kinerja internet dan ketersediaan untuk lalu lintas jaringan yang tersedia untuk layanan melalui AWS. Monitor Internet menerapkan informasi yang relevan dengan aplikasi Anda, berdasarkan lokasi geografis untuk ASN dan layanan di mana klien menggunakan aplikasi Anda: pasangan jaringan kota yang terpengaruh. Lokasi ditentukan dari sumber daya yang Anda tambahkan ke monitor Anda. Kemudian Monitor Internet menggunakan analisis statistik untuk mendeteksi kapan performa dan ketersediaan menurun, memengaruhi pengalaman klien di aplikasi Anda.

Skor performa dan ketersediaan yang dihitung oleh Monitor Internet direpresentasikan sebagai persentase lalu lintas yang tidak terdapat penurunan. Dampak adalah kebalikan dari ini: ini adalah representasi dari seberapa banyak sebuah masalah memberikan dampak bagi pelanggan dari pengguna akhir. Jadi jika ada penurunan ketersediaan global sebesar 93%, misalnya, dampak menyesuaikan menjadi 7%.

Ketika skor performa atau ketersediaan untuk pasangan jaringan kota di aplikasi Anda secara global mencapai ambang batas atau turun di bawah ambang batas peristiwa kondisi kesehatan yang sesuai atas performa atau ketersediaan, Monitor Internet akan mengeluarkan sebuah peristiwa kondisi kesehatan. Secara bawaan, ambang batas adalah 95% untuk performa dan ketersediaan. Nilai yang memenuhi ambang batas, atau berada di bawah itu, bersifat kumulatif, sehingga bisa berarti beberapa event yang lebih kecil digabungkan untuk memenuhi persentase ambang batas, atau bahwa ada satu event yang memenuhi atau berada di bawah tingkat ambang batas.

Selama skor performa atau ketersediaan yang memicu sebuah event berada tepat atau berada di bawah persentase ambang peristiwa kondisi kesehatan sesuai mengenai dampak keseluruhan, maka status peristiwa kondisi kesehatan tetap aktif. Ketika skor atau skor gabungan yang memicu status event naik di atas ambang batas, Monitor Internet akan menyelesaikan peristiwa kondisi kesehatan.

Monitor Internet juga membuat peristiwa kondisi kesehatan berdasarkan ambang batas lokal dan persentase lalu lintas keseluruhan yang disebabkan oleh sebuah masalah. Anda dapat mengonfigurasi opsi untuk ambang batas lokal, atau mematikan ambang batas lokal sekalian.

Untuk informasi selengkapnya tentang mengonfigurasi ambang batas peristiwa kondisi kesehatan, silakan lihat Mengubah ambang batas peristiwa kondisi kesehatan.

Penentuan waktu pelaporan peristiwa kondisi kesehatan

Monitor Internet menggunakan agregator untuk mengumpulkan semua sinyal tentang masalah internet, untuk membuat peristiwa kondisi kesehatan di monitor dalam beberapa menit.

Jika memungkinkan, Internet Monitor menganalisis asal mula peristiwa kesehatan, untuk menentukan apakah itu disebabkan oleh AWS atau ASN. Analisis peristiwa kondisi kesehatan berlanjut setelah sebuah event diselesaikan. Monitor Internet dapat memperbarui event dengan informasi baru hingga satu jam.

Cara Monitor Internet bekerja dengan lalu lintas IPv4 dan IPv6

Monitor Internet mengukur kesehatan terhadap jaringan hanya melalui IPv4, dan menunjukkan kepada Anda peristiwa kondisi kesehatan, serta metrik ketersediaan dan performa, jika Anda melayani lalu lintas ke jaringan tersebut melalui famili IP apa pun (IPv4 atau IPv6). Jika Anda melayani lalu lintas dari sumber daya dual-stack, seperti CloudFront distribusi dual-stack, Internet Monitor memunculkan acara kesehatan dan menunjukkan penurunan skor kinerja atau skor ketersediaan hanya jika lalu lintas IPv4 memiliki masalah yang sama untuk sumber daya seperti lalu lintas IPv6.

Perhatikan bahwa metrik Monitor Internet untuk keseluruhan byte yang masuk dan byte yang keluar secara akurat mencerminkan semua lalu lintas internet (IPv4 dan IPv6).

Contoh kasus penggunaan Amazon CloudWatch Internet Monitor

Pada bagian ini, kami menjelaskan beberapa contoh spesifik, dengan tautan ke posting blog dengan detail selengkapnya. Contoh-contoh ini menunjukkan bagaimana Anda dapat menggunakan kemampuan Amazon CloudWatch Internet Monitor untuk membantu Anda memantau aplikasi Anda dan meningkatkan pengalaman pengguna Anda.

Siapkan peringatan dan putuskan tindakan yang akan diambil

Anda dapat menggunakan Monitor Internet untuk mendapatkan wawasan tentang metrik performa internet rata-rata dari waktu ke waktu, dan tentang peristiwa kondisi kesehatan berdasarkan jaringan kota (lokasi klien dan ASN, biasanya penyedia layanan internet). Menggunakan Internet Monitor, Anda dapat mengidentifikasi peristiwa yang memengaruhi pengalaman pengguna akhir untuk aplikasi yang dihosting di Amazon Virtual Private Clouds (VPC), Network Load Balancers, Amazon WorkSpaces, atau Amazon. CloudFront

Setelah Anda membuat monitor, Anda memiliki beberapa opsi untuk mendapat pemberitahuan peristiwa kondisi kesehatan di Monitor Internet. Ini termasuk pemberitahuan berdasarkan CloudWatch Alarm menggunakan metrik acara atau EventBridge aturan Amazon untuk memfilter acara kesehatan. Anda dapat memilih opsi berbeda untuk pemberitahuan atau tindakan berdasarkan alarm, termasuk, misalnya, AWS SMS pemberitahuan atau pembaruan ke grup CloudWatch log.

Untuk melihat contoh dengan panduan terperinci, lihat posting blog berikut: Memperkenalkan Amazon CloudWatch Internet Monitor.

Mengidentifikasi masalah latensi dan meningkatkan TTFB untuk meningkatkan pengalaman gameplay multipemain

Gunakan Monitor Internet untuk membantu Anda mengidentifikasi dengan cepat di mana para pemain game di aplikasi game cloud global yang mengalami masalah latensi secara global, dan memberikan wawasan tentang peningkatan performa. Dengan mengidentifikasi di mana sebagian besar pemain saat ini dengan waktu untuk mendapatkan byte pertama (TTFB) paling lambat, Anda dapat mengetahui cara meningkatkan latensi untuk membuat basis pemain terbesar Anda lebih puas.

Sekarang, ketika Anda siap untuk menyebarkan server EC2 berikutnya untuk game Anda, pilih Wilayah AWS yang disarankan Internet Monitor akan menurunkan TTFB di area dengan latensi tinggi dan sekelompok besar pemain.

Untuk detail tentang pengaturan dan penggunaan Internet Monitor untuk kasus penggunaan ini, lihat posting blog berikut: Menggunakan Amazon CloudWatch Internet Monitor untuk Pengalaman Gaming yang Lebih Baik.

Observabilitas lintas akun Monitor Internet

Dengan observabilitas lintas akun Internet Monitor, Anda dapat memantau aplikasi Anda yang menjangkau beberapa AWS akun dalam satu akun. Wilayah AWS

Anda dapat menggunakan Amazon CloudWatch Observability Access Manager untuk menyiapkan satu atau beberapa AWS akun Anda sebagai akun pemantauan. Anda akan memberikan akun pemantauan dengan kemampuan untuk melihat data di akun sumber Anda dengan membuat sink di akun pemantauan Anda. Sink adalah sumber daya yang mewakili titik lampiran dalam akun pemantauan. Untuk Internet Monitor, titik lampiran sumber daya adalah monitor. Anda menggunakan sink tersebut untuk membuat sebuah tautan dari akun sumber Anda ke akun pemantauan Anda. Untuk informasi selengkapnya, lihat CloudWatch observabilitas lintas akun.

Sumber daya yang dibutuhkan

Untuk fungsionalitas yang tepat dari observabilitas lintas akun CloudWatch Application Insights, pastikan bahwa jenis telemetri berikut dibagikan melalui Observability Access Manager. CloudWatch

- Monitor di Internet Monitor
- Metrik di Amazon CloudWatch
- Grup log di Amazon CloudWatch Logs

Memulai dengan Amazon CloudWatch Internet Monitor menggunakan konsol

Untuk memulai dengan Amazon CloudWatch Internet Monitor, Anda harus membuat monitor di Internet Monitor untuk aplikasi Anda dengan menambahkan AWS sumber daya yang digunakannya dan mengatur beberapa opsi konfigurasi. Bab ini menyediakan prosedur untuk menambahkan sebuah monitor di konsol. Bab ini juga mencakup bagian dengan detail lebih lanjut tentang sumber

daya di Monitor Internet, dan kemudian ada bagian tambahan dengan deskripsi dan batasan untuk berbagai opsi yang dapat atau harus Anda konfigurasikan untuk monitor Anda.

Daftar Isi

Membuat monitor di Amazon CloudWatch Internet Monitor menggunakan konsol

Anda membuat monitor di Amazon CloudWatch Internet Monitor untuk aplikasi Anda dengan menambahkan AWS sumber daya yang digunakannya, dan kemudian mengatur beberapa opsi konfigurasi. Sumber daya yang Anda tambahkan, Amazon Virtual Private Clouds (VPC), Network Load Balancers, CloudFront distribusi, atau WorkSpaces direktori, memberikan informasi untuk Internet Monitor untuk memetakan informasi lalu lintas internet untuk aplikasi Anda. Setelah Anda membuat monitor Anda, tunggu 10-15 menit, dan kemudian Anda dapat menggunakan monitor pada Monitor Internet, atau alat lain, untuk memvisualisasikan dan mengeksplorasi performa dan ketersediaan tentang penggunaan klien Anda. Alat-alat ini memberikan wawasan untuk Anda menggunakan pengukuran lalu lintas aplikasi Anda, dikumpulkan dan dipublikasikan oleh monitor, misalnya, ke CloudWatch Log.

Biasanya, yang termudah adalah dengan membuat satu monitor di Monitor Internet untuk satu aplikasi. Dalam monitor yang sama, Anda dapat mencari dan menyortir pengukuran dan metrik dalam berkas log Monitor Internet berdasarkan lokasi dan ASN yang berbeda (biasanya penyedia layanan internet), atau informasi lainnya. Anda tidak perlu membuat monitor terpisah untuk aplikasi di area yang berbeda, misalnya.

Langkah-langkah di sini memandu Anda selangkah demi selangkah mengatur monitor Anda dengan menggunakan konsol. Untuk melihat contoh penggunaan AWS Command Line Interface

dengan tindakan Internet Monitor API, untuk membuat monitor, melihat peristiwa, dan sebagainya, lihatContoh menggunakan CLI dengan Amazon Internet Monitor CloudWatch .

Membuat sebuah monitor menggunakan konsol

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi kiri, di bawah Pemantauan jaringan, pilih Monitor Internet.
3. Pilih Buat monitor.
4. Untuk nama Monitor, masukkan nama yang ingin Anda gunakan untuk monitor ini di Monitor Internet.
5. Pilih Tambahkan sumber daya, lalu pilih sumber daya untuk menetapkan batasan-batasan pemantauan untuk digunakan oleh Monitor Internet untuk monitor ini.

   > **Note**
   >
   > Waspadai hal-hal berikut:
   >
   > - Untuk menghasilkan output yang berarti dengan Internet Monitor, VPC yang Anda tambahkan harus terhubung ke internet dengan memiliki Internet Gateway yang dikonfigurasi.
   > - Anda dapat menambahkan kombinasi VPC dan CloudFront distribusi, atau Anda dapat menambahkan WorkSpaces direktori, atau Anda dapat menambahkan Network Load Balancers. Anda tidak dapat menambahkan Network Load Balancers atau WorkSpaces direktori bersama dengan jenis sumber daya lainnya.

6. Pilih persentase lalu lintas internet Anda untuk dipantau.
7. Secara opsional, tentukan opsi tambahan di bawah Pengaturan lanjutan.

   - Pada Batasan maksimum jaringan kota, Anda dapat memilih batasan untuk jumlah jaringan kota (lokasi dan ASN, atau penyedia layanan internet) yang akan dipantau oleh Monitor Internet. Anda dapat mengubah batasan maksimum kapan saja dengan mengedit monitor Anda. Lihat Memilih batas maksimum jaringan kota.

     Untuk mengatur ulang ke default, masukkan `500000`.

     Jika Anda menetapkan batasan maksimum jaringan kota, penetapan itu menentukan batas jumlah jaringan kota yang dipantau Monitor Internet untuk aplikasi Anda, terlepas dari persentase lalu lintas yang Anda pilih untuk dipantau.

- Secara opsional, Anda dapat menentukan nama bucket Amazon S3 dan prefiks kustom untuk menerbitkan pengukuran internet ke Amazon S3 untuk semua jaringan kota yang dipantau.

  Internet Monitor menerbitkan 500 pengukuran internet teratas (berdasarkan volume lalu lintas) untuk aplikasi Anda ke CloudWatch Log setiap lima menit. Jika Anda memilih untuk mempublikasikan pengukuran ke S3, pengukuran masih dipublikasikan ke CloudWatch Log. Untuk informasi selengkapnya, lihat Menerbitkan pengukuran internet ke Amazon S3 di Amazon CloudWatch Internet Monitor.
- Secara opsional, Anda dapat menambahkan tanda untuk monitor Anda.

8. Pilih Buat monitor.

Setelah Anda membuat sebuah monitor, Anda dapat mengedit monitor kapan saja, misalnya, untuk mengubah persentase lalu lintas aplikasi, memperbarui batas jaringan kota maksimum atau menambah atau menghapus sumber daya. Anda juga dapat menghapus monitor. Untuk melakukan tugas-tugas ini di konsol Monitor Internet, pilih monitor, lalu pilih opsi di menu Tindakan. Perhatikan bahwa Anda tidak dapat mengubah nama monitor.

Cara melihat dasbor Monitor Internet

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Pemantauan jaringan, lalu Monitor Internet.

   Tab Monitor menampilkan daftar monitor yang telah Anda buat.

Untuk melihat informasi lebih lanjut tentang monitor tertentu, pilih monitor.

Menambahkan sumber daya ke monitor Anda

Saat membuat monitor, Anda mengaitkan sumber daya aplikasi Anda dengannya: Amazon Virtual Private Clouds (VPC), Network Load Balancers, CloudFront distribusi Amazon, atau direktori Amazon. WorkSpaces Kemudian Monitor Internet akan mengetahui di mana lalu lintas yang dapat diakses pada aplikasi Anda dan dimana para klien berada, sehingga Monitor Internet dapat menentukan pengukuran yang relevan bagi monitor Anda untuk dipublikasikan.

Anda dapat menambahkan sumber daya berikut ke monitor di Monitor Internet.

- VPC: Setiap VPC yang Anda tambahkan di sebuah Wilayah adalah sumber daya yang dipantau. Saat Anda menambahkan VPC, Internet Monitor memantau lalu lintas untuk aplikasi apa pun

yang menghadap ke internet di VPC, misalnya, aplikasi yang dihosting di instans Amazon EC2, di belakang Network Load Balancer, atau dalam wadah. AWS Fargate

- Penyeimbang Beban Jaringan (NLB): Setiap NLB yang Anda tambahkan adalah sumber daya yang dipantau.
- CloudFront Distribusi: Setiap CloudFront distribusi yang Anda tambahkan adalah sumber daya yang dipantau.
- WorkSpaces direktori: Setiap WorkSpaces direktori yang Anda tambahkan di Wilayah adalah sumber daya yang dipantau.

Ketika Anda memantau lalu lintas untuk VPC, lalu lintas untuk aplikasi yang di-host pada penyeimbang beban di belakang VPC akan dipantau. Anda dapat memilih untuk memantau lalu lintas untuk penyeimbang beban Penyeimbang Beban Jaringan individu daripada harus memantau VPC dengan beberapa penyeimbang beban. Hal ini dapat berguna, misalnya, jika Anda perlu memahami dan mengonfigurasi fitur-fitur untuk performa atau efisiensi yang lebih baik di tingkat penyeimbang beban. Atau, Anda mungkin memerlukan informasi kepatuhan di tingkat Penyeimbang Beban Jaringan.

Saat Anda menambahkan sumber daya ke monitor di dalam Monitor Internet, waspadai berikut ini:

- Untuk menghasilkan output yang berarti dengan Internet Monitor, VPC yang Anda tambahkan harus terhubung ke internet dengan memiliki Internet Gateway yang dikonfigurasi.
- Anda dapat menambahkan kombinasi Amazon VPC dan CloudFront distribusi, atau Anda dapat menambahkan WorkSpaces direktori, atau Anda dapat menambahkan Network Load Balancers. Anda tidak dapat menambahkan Network Load Balancers atau WorkSpaces direktori bersama dengan jenis sumber daya lainnya.

Ada perbedaan Wilayah untuk Wilayah keikutsertaan yang perlu diingat ketika Anda menambahkan VPC dan NLB sebagai sumber daya. Untuk informasi selengkapnya, lihat Didukung Wilayah AWS untuk Amazon CloudWatch Internet Monitor.

Sebagai tambahan, ada perbedaan sumber daya tentang pengukuran latensi last mile. Untuk pengukuran latensi Monitor Internet, VPC, NLB, dan WorkSpaces direktori tidak menyertakan latensi last-mile.

Memilih persentase lalu lintas aplikasi untuk dipantau

Cakupan yang Anda pilih untuk persentase lalu lintas aplikasi untuk dipantau menentukan berapa banyak jaringan kota (lokasi klien dan ASN, biasanya penyedia layanan internet) yang dipantau untuk aplikasi Anda, hingga batas maksimum jaringan kota pilihan yang juga bisa Anda atur.

Jika Anda memilih untuk memantau kurang dari 100% lalu lintas aplikasi Anda, celah observabilitas mungkin ada dengan monitor Anda. Itu karena jika ada acara kesehatan yang dibuat oleh Amazon CloudWatch Internet Monitor di mana Anda tidak memantau lalu lintas, Anda tidak akan menyadari masalah tersebut. Anda mungkin juga memiliki cakupan yang lebih sedikit atas informasi skor performa dan ketersediaan tentang akses klien ke aplikasi Anda.

Bagian berikut menjelaskan opsi untuk mengeksplorasi pengaturan dan cakupan persentase lalu lintas, dan untuk mendapatkan gambaran tentang dampak peningkatan atau penurunan cakupan.

- Jelajahi perubahan persentase lalu lintas aplikasi Anda
- Lihat jumlah jaringan kota yang dipantau pada pengaturan persentase lalu lintas yang berbeda

Jelajahi perubahan persentase lalu lintas aplikasi Anda

Anda dapat menjelajahi nilai-nilai dimana Anda mungkin ingin mengubah persentase lalu lintas aplikasi Anda, dengan cara melihat jumlah jaringan kota yang dipantau saat Anda mengubah persentase. Prosedur di bagian ini memberikan step-by-step informasi.

Di konsol Monitor Internet, Anda dapat mencoba meningkatkan atau mengurangi persentase lalu lintas aplikasi untuk monitor Anda, dan melihat perkiraan jumlah jaringan kota Anda yang akan dicakup sebagai hasilnya. Dengan opsi ini, Anda dapat dengan cepat melihat bagaimana pengubahan persentase lalu lintas Anda memengaruhi jumlah monitor kota yang dipantau. Opsi ini dapat membantu Anda mendapat perkiraan bagaimana persentase lalu lintas aplikasi yang bagus untuk dipilih untuk aplikasi Anda.

Untuk mengeksplorasi pemantauan cakupan dengan menambah dan menurunkan persentase lalu lintas aplikasi

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi kiri, di bawah Pemantauan jaringan, pilih Monitor Internet.
3. Dalam daftar monitor Anda, pilih sebuah monitor.
4. Pada tab Gambaran umum, di bagian Lalu lintas yang Dipantau, pilih grafik persentase, lalu pilih Perbarui cakupan pemantauan.

5. Pada dialog Jelajahi dan atur cakupan pemantauan lalu lintas, klik panah untuk menambah atau mengurangi persentase lalu lintas yang akan dipantau. Dengan memilih lalu lintas 100%, Anda dapat melihat berapa banyak jaringan kota yang dipantau dengan cakupan penuh untuk memantau aplikasi Anda.
6. Untuk mempelajari lebih lanjut tentang bagaimana jumlah jaringan kota yang dipantau (diperkirakan di sini) dapat memengaruhi biaya Anda, pilih tautan ke kalkulator CloudWatch Harga, lalu gulir ke bawah ke Internet Monitor.
7. Untuk menetapkan persentase lalu lintas baru yang akan dipantau, pilih Perbarui cakupan monitor. Atau, untuk mempertahankan tingkat cakupan saat ini, pilih Batal.

Lihat jumlah jaringan kota yang dipantau pada pengaturan persentase lalu lintas yang berbeda

Anda dapat melihat jumlah jaringan kota yang akan dipantau untuk aplikasi Anda pada persentase lalu lintas aplikasi yang berbeda. Prosedur di bagian ini memberikan step-by-step informasi.

Di konsol Monitor Internet, Anda dapat melihat grafik yang menunjukkan bagaimana cakupan untuk jaringan kota Anda akan berubah pada persentase lalu lintas aplikasi yang berbeda, selama interval waktu yang Anda tentukan. Ini adalah cara cepat untuk memvisualisasikan dan membandingkan cakupan pemantauan untuk aplikasi Anda pada persentase lalu lintas tertentu, semuanya dalam satu grafik.

Untuk melihat grafik persentase lalu lintas aplikasi dan cakupan jaringan kota yang sesuai

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi kiri, di bawah Pemantauan jaringan, pilih Monitor Internet.
3. Dalam daftar monitor Anda, pilih sebuah monitor.
4. Pilih tab Wawasan lalu lintas, dan gulir ke bawah ke grafik lalu lintas Internet.
5. Pada Bandingkan pilihan untuk cakupan lalu lintas, di daftar drop-down, pilih satu atau beberapa persentase. Anda dapat memilih satu atau beberapa persentase lalu lintas aplikasi, dan grafik Total jaringan kota yang dipantau akan diperbarui untuk menampilkan cakupan pemantauan yang disediakan Monitor Internet untuk persentase lalu lintas tersebut. Dengan memilih Jaringan kota dengan Lalu lintas 100%, Anda dapat melihat berapa banyak jaringan kota yang dipantau dengan cakupan penuh untuk memantau aplikasi Anda.

Ingatlah hal berikut:

- Cakupan lalu lintas dihitung berdasarkan jumlah jaringan kota di jam sebelumnya pada lalu lintas aplikasi Anda. Ini berarti bahwa, setelah Anda memilih persentase lalu lintas tertentu untuk dipantau, jaringan kota yang lebih sedikit kemungkinan dipantau untuk aplikasi Anda daripada yang ditampilkan di sini dalam grafik perbandingan cakupan lalu lintas.
- Untuk memastikan bahwa semua lalu lintas aplikasi Anda dipantau, atur `TrafficPercentageToMonitor` ke 100 dan jangan setel `MaxCityNetworksToMonitor`. Atau, Anda dapat mengatur `MaxCityNetworksToMonitor` ke 500.000, yang merupakan batas atas di Monitor Internet.
- Jika Anda menetapkan batasan maksimum jaringan kota, jumlah total jaringan kota yang dipantau tidak akan pernah melebihi batasan tersebut, terlepas dari pilihan persentase lalu lintas aplikasi yang Anda pilih.
- Anda dapat mempelajari lebih lanjut tentang bagaimana jumlah jaringan kota yang dipantau dapat memengaruhi biaya Anda. Pada kalkulator Harga untuk CloudWatch halaman, gulir ke bawah ke Internet Monitor.

Untuk menetapkan persentase lalu lintas baru yang akan dipantau, pada Jelajahi pilihan cakupan lalu lintas lainnya, pilih Perbarui cakupan pemantauan. Pada dialog, pilih persentase lalu lintas, lalu pilih Perbarui cakupan monitor.

Memilih batas maksimum jaringan kota

Amazon CloudWatch Internet Monitor dapat memantau lalu lintas aplikasi Anda untuk beberapa atau semua lokasi di mana klien mengakses sumber daya aplikasi Anda, dan semua ASN (biasanya penyedia layanan internet) yang mereka akses melalui aplikasi Anda—yaitu, jaringan kota untuk lalu lintas internet aplikasi Anda. Pilih persentase lalu lintas aplikasi untuk dipantau saat Anda membuat monitor Anda, yang dapat Anda perbarui kapan saja dengan mengedit monitor.

Selain menetapkan persentase lalu lintas, Anda juga dapat menetapkan batas maksimum untuk jumlah jaringan kota yang dipantau. Bagian ini menjelaskan bagaimana batas jaringan kota dapat membantu Anda mengelola biaya penagihan, dan memberikan informasi dan sebuah contoh untuk membantu Anda menentukan batas yang harus ditetapkan.

Batas maksimum yang Anda tetapkan untuk jumlah jaringan kota membantu memastikan bahwa tagihan Anda dapat diprediksi. Untuk informasi selengkapnya, lihat CloudWatch Harga Amazon. Anda juga dapat mempelajari bagaimana nilai yang berbeda untuk jumlah jaringan kota yang benar-benar dipantau dapat memengaruhi tagihan Anda dengan menggunakan kalkulator harga.

CloudWatch Untuk menjelajahi opsi, pada kalkulator Harga untuk CloudWatch halaman, gulir ke bawah ke Internet Monitor.

Untuk memperbarui monitor Anda dan mengubah batas maksimum jaringan kota, silakan lihat Mengedit atau menghapus sebuah monitor di Monitor Internet.

Cara kerja penagihan dengan batas maksimum jaringan kota

Menetapkan batas maksimum untuk jumlah jaringan kota yang dipantau dapat membantu mencegah biaya tak terduga dalam tagihan Anda. Hal ini berguna, misalnya, jika pola lalu lintas Anda sangat bervariasi. Biaya penagihan meningkat untuk setiap jaringan kota yang dipantau setelah 100 jaringan kota pertama, yang disertakan (di semua monitor per akun). Jika Anda menetapkan batasan maksimum jaringan kota, penetapan itu membatasi jumlah jaringan kota yang dipantau Monitor Internet untuk aplikasi Anda, terlepas dari persentase lalu lintas yang Anda pilih untuk dipantau.

Anda hanya membayar untuk jumlah jaringan kota yang benar-benar dipantau. Batasan maksimum jaringan kota yang Anda pilih memungkinkan Anda menetapkan total batasan yang dapat disertakan saat Monitor Internet memantau lalu lintas dengan monitor Anda. Anda dapat mengubah batasan maksimum kapan saja dengan mengedit monitor Anda.

Untuk menjelajahi opsi, pada kalkulator Harga untuk CloudWatch halaman, gulir ke bawah ke Internet Monitor. Untuk informasi selengkapnya tentang harga Monitor Internet, lihat bagian Monitor Internet di halaman CloudWatch Harga Amazon.

Cara memilih batas maksimum jaringan kota

Untuk membantu Anda memutuskan batasan maksimum jaringan kota untuk dipilih, pertimbangkan berapa banyak lalu lintas yang ingin Anda pantau untuk aplikasi Anda. Metrik Monitor Internet berikut ini dapat membantu Anda menganalisis penggunaan dan cakupan lalu lintas setelah Anda membuat monitor Anda: `CityNetworksMonitored`, `TrafficMonitoredPercent`, dan satu metrik `CityNetworksFor`*`NN`*`PercentTraffic` lagi atau lebih, di mana *`NN`* adalah nilai persentase yang merupakan salah satu dari berikut ini: 25, 50, 90, 95, 99, atau 100. Untuk meninjau definisi metrik ini, dan semua metrik Monitor Internet lainnya, silakan lihat Menggunakan CloudWatch Metrik dengan Amazon CloudWatch Internet Monitor.

Untuk melihat grafik ikhtisar cakupan lalu lintas internet Anda, buka tab Wawasan lalu lintas di CloudWatch dasbor dan, di bagian grafik lalu lintas Internet, pilih opsi untuk opsi Bandingkan untuk cakupan lalu lintas. Grafik yang ditampilkan di bagian tersebut menampilkan jumlah sebenarnya dari jaringan kota yang dipantau untuk aplikasi Anda, dan juga garis grafik untuk persentase lalu lintas

aplikasi yang berbeda yang Anda pilih di daftar drop-down. Untuk mempelajari lebih lanjut, silakan lihat Pengaturan persentase lalu lintas aplikasi Anda.

Untuk menjelajahi opsi Anda secara lebih terperinci, Anda dapat menggunakan metrik Monitor Internet, seperti yang dijelaskan dalam contoh berikut. Contoh-contoh ini menunjukkan bagaimana memilih batas maksimum jaringan kota yang terbaik untuk Anda, tergantung pada luasnya cakupan lalu lintas internet aplikasi yang Anda inginkan. Menggunakan kueri untuk metrik Internet Monitor di CloudWatch Metrik dapat membantu Anda memahami lebih lanjut tentang cakupan lalu lintas internet aplikasi Anda.

Contoh penentuan batas maksimum jaringan kota

Misalnya, Anda telah menetapkan batas maksimum pemantauan sebanyak 100 jaringan kota dan aplikasi Anda diakses oleh klien di 2637 jaringan kota. Di CloudWatch Metrik, Anda akan melihat metrik Internet Monitor berikut ditampilkan:

```
CityNetworksMonitored 100
TrafficMonitoredPercent  12.5
CityNetworksFor90PercentTraffic  2143
CityNetworksFor100PercentTraffic  2637
```

Dari contoh ini, Anda dapat melihat bahwa saat ini Anda sedang memantau 12,5% lalu lintas internet Anda, dengan batas maksimum ditetapkan ke 100 jaringan kota. Jika Anda ingin memantau 90% lalu lintas Anda, metrik berikutnya memberikan informasi tentang itu: `CityNetworksFor90PercentTraffic` menunjukkan bahwa Anda perlu memantau 2.143 jaringan kota untuk cakupan 90%. Untuk melakukan itu, Anda perbarui monitor Anda dan tetapkan batasan maksimum jaringan kota menjadi 2.143.

Sama halnya, katakanlah Anda ingin memiliki 100% pemantauan lalu lintas internet untuk aplikasi Anda. Metrik berikutnya, `CityNetworksFor100PercentTraffic`, menunjukkan bahwa untuk melakukan ini, Anda harus memperbarui monitor Anda untuk menetapkan batas maksimum jaringan kota menjadi 2.637.

Jika Anda sekarang menetapkan batas maksimum menjadi 5.000 jaringan kota, karena itu lebih besar dari 2.637, maka Anda akan melihat metrik berikut ditampilkan:

```
CityNetworksMonitored 2637
TrafficMonitoredPercent  100
CityNetworksFor90PercentTraffic  2143
```

```
CityNetworksFor100PercentTraffic  2637
```

Dari metrik ini, Anda dapat melihat bahwa dengan batas yang lebih tinggi, Anda bisa memantau seluruh 2.637 jaringan kota, yang merupakan 100% dari lalu lintas internet Anda.

Menerbitkan pengukuran internet ke Amazon S3 di Amazon CloudWatch Internet Monitor

Anda dapat memilih agar Amazon CloudWatch Internet Monitor mempublikasikan pengukuran internet ke Amazon S3 untuk lalu lintas yang menghadap internet ke jaringan kota yang dipantau (lokasi klien dan ASN, biasanya penyedia layanan internet) di monitor Anda, hingga batas layanan jaringan kota 500.000. Internet Monitor secara otomatis menerbitkan pengukuran internet ke CloudWatch Log setiap lima menit untuk 500 jaringan kota teratas (berdasarkan volume lalu lintas) untuk setiap monitor. Pengukuran yang diterbitkan ke S3 termasuk 500 teratas yang diterbitkan ke CloudWatch Log.

Anda dapat memilih opsi untuk menerbitkannya ke S3, dan mencantumkan bucket untuk menerbitkan pengukurannya, hingga pada saat Anda membuat atau memperbarui monitor Anda. Bucket harus sudah dibuat di S3 sebelum Anda dapat mencantumkannya di Monitor Internet. Ada batas layanan sebanyak 500.000 jaringan kota untuk pengukuran internet yang dipublikasikan di S3. Monitor Internet menerbitkan pengukuran internet ke S3 sebagai event, serangkaian objek berkas log terkompresi yang disimpan ke dalam bucket.

Saat Anda membuat bucket S3 untuk Internet Monitor untuk mempublikasikan pengukuran, pastikan Anda mengikuti panduan izin yang disediakan oleh CloudWatch Log. Melakukannya memastikan bahwa Internet Monitor dapat mempublikasikan log langsung ke S3, dan itu AWS dapat, jika diperlukan, membuat dan mengubah kebijakan sumber daya yang terkait dengan grup log yang menerima log. Untuk informasi selengkapnya, lihat Log yang dikirim ke CloudWatch Log di Panduan Pengguna CloudWatch Log Amazon.

Berkas log yang dipublikasikan terkompresi. Jika Anda membuka berkas log menggunakan konsol Amazon S3, berkas log akan terdekompresi dan event pengukuran internet akan ditampilkan. Jika Anda mengunduh berkas-berkasnya, Anda harus mendekompresi semuanya untuk melihat event-event.

Anda juga dapat menjalankan kueri pengukuran internet di berkas log menggunakan Amazon Athena. Amazon Athena adalah layanan kueri interaktif yang memudahkan analisis data di Amazon S3 dengan menggunakan SQL standar. Untuk informasi selengkapnya, lihat Menggunakan Amazon Athena untuk menjalankan kueri pengukuran internet di berkas log Amazon S3.

Menggunakan sebuah monitor di Monitor Internet

Ada beberapa cara untuk menggunakan monitor Amazon CloudWatch Internet Monitor setelah Anda membuatnya: misalnya, Anda dapat melihat informasi di CloudWatch dasbor, mendapatkan informasi dengan menggunakan AWS Command Line Interface, dan mengatur peringatan kesehatan.

Monitor Anda memberikan informasi tentang aplikasi dan preferensi konfigurasi Anda sehingga Monitor Internet dapat menyesuaikan pengukuran dan metrik untuk menerbitkannya dalam bentuk event untuk Anda. Monitor Internet mengumpulkan pengukuran dari jejak infrastruktur global untuk AWS. Pengukuran ini merupakan jumlah performa jaringan dan informasi ketersediaan yang sangat besar, dari seluruh dunia. Dengan menggunakan informasi dari sumber daya yang Anda tambahkan untuk aplikasi Anda, Monitor Internet menerbitkan pengukuran performa dan ketersediaan untuk Anda yang tercakup ke jaringan kota (yakni, lokasi klien dan ASN, biasanya penyedia layanan internet atau ISP) dimana aplikasi Anda aktif. Jadi, pengukuran dan metrik di dasbor Monitor Internet dan di CloudWatch Log — tentang ketersediaan, kinerja, byte yang dipantau yang ditransfer, dan waktu pulang-pergi — khusus untuk lokasi klien dan ASN Anda.

Monitor Internet juga menentukan kapan ada anomali dalam performa dan ketersediaan. Secara default, Internet Monitor melapisi lalu lintas Anda dengan pengukuran ketersediaan dan kinerja yang AWS telah dikumpulkan untuk setiap pasangan sumber-tujuan di lokasi klien Anda, untuk menentukan kapan ada penurunan kinerja atau ketersediaan yang signifikan. Ketika ada degradasi yang signifikan untuk lokasi dan cakupan aplikasi Anda, Monitor Internet menghasilkan peristiwa kondisi kesehatan, dan menerbitkan informasi tentang masalah tersebut ke monitor Anda.

Setelah Anda membuat sebuah monitor, Anda dapat menggunakannya untuk mengakses atau mendapat pemberitahuan tentang informasi yang disediakan Monitor Internet, dengan cara-cara sebagai berikut:

- Gunakan CloudWatch dasbor untuk melihat dan menjelajahi kinerja, ketersediaan, dan acara kesehatan; jelajahi data historis aplikasi Anda; dan dapatkan wawasan tentang cara baru untuk mengonfigurasi aplikasi Anda untuk kinerja yang lebih baik. Untuk mempelajari selengkapnya, silakan lihat di bawah ini:
  - Melacak kinerja dan ketersediaan real-time di Amazon CloudWatch Internet Monitor (tab Ikhtisar)
  - Memfilter dan melihat data historis di Amazon CloudWatch Internet Monitor (tab penjelajah sejarah)
  - Mendapatkan wawasan untuk meningkatkan kinerja aplikasi di Amazon CloudWatch Internet Monitor (tab wawasan lalu lintas)

- Konfigurasikan ambang batas peristiwa kondisi kesehatan untuk mengubah apa yang memicu Monitor Internet untuk memberikan peristiwa kondisi kesehatan di aplikasi Anda. Anda dapat mengonfigurasi ambang batas keseluruhan dan ambang batas lokal (jaringan kota). Untuk mempelajari selengkapnya, silakan lihat Mengubah ambang batas peristiwa kondisi kesehatan.
- Gunakan AWS CLI perintah dengan tindakan Internet Monitor API untuk melihat informasi profil lalu lintas, melihat pengukuran, daftar peristiwa kesehatan, dan sebagainya. Untuk mempelajari selengkapnya, lihat Contoh menggunakan CLI dengan Amazon Internet Monitor CloudWatch .
- Gunakan CloudWatch alat standar, seperti CloudWatch Contributor Insights, CloudWatch Metrics explorer, dan CloudWatch Logs Insights untuk memvisualisasikan data. CloudWatch Untuk mempelajari selengkapnya, silakan lihat Menjelajahi data Anda dengan CloudWatch alat dan antarmuka kueri Internet Monitor.
- Gunakan Athena dengan log S3 untuk mengakses dan menganalisis pengukuran internet di Monitor Internet untuk aplikasi Anda, jika Anda mengaktifkan publikasi pengukuran ke S3.
- Buat EventBridge notifikasi Amazon untuk mengingatkan Anda ketika Internet Monitor menentukan ada acara kesehatan. Untuk mempelajari selengkapnya, lihat Menggunakan Amazon CloudWatch Internet Monitor dengan Amazon EventBridge.
- Menerima AWS Health Dashboard pemberitahuan secara otomatis, ketika Internet Monitor menentukan bahwa masalah disebabkan oleh AWS jaringan. Pemberitahuan mencakup langkah-langkah yang AWS diambil untuk mengurangi masalah.

Mengedit atau menghapus sebuah monitor di Monitor Internet

Menggunakan menu Tindakan, Anda dapat mengedit atau menghapus monitor di Amazon CloudWatch Internet Monitor setelah Anda membuatnya. Misalnya, Anda dapat mengedit monitor untuk melakukan hal berikut:

- Mengubah persentase lalu lintas aplikasi untuk dipantau
- Menetapkan atau memperbarui batas maksimum jaringan kota
- Mengubah ambang batas peristiwa kondisi kesehatan untuk skor ketersediaan atau skor performa
- Menambah atau menghapus sumber daya
- Mengaktifkan atau memperbarui publikasi event ke Amazon S3

Anda juga dapat menghapus sebuah monitor. Perhatikan bahwa Anda tidak dapat mengubah nama monitor setelah Anda membuatnya.

Untuk membuat perubahan pada monitor atau menghapus monitor, gunakan salah satu prosedur berikut.

Untuk mengedit monitor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi kiri, di bawah Pemantauan jaringan, pilih Monitor Internet.
3. Pilih monitor Anda, lalu pilih menu tindakan.
4. Pilih Perbarui monitor.
5. Buat pembaruan yang diinginkan. Misalnya, untuk mengubah persentase lalu lintas yang akan dipantau, pada Lalu lintas aplikasi untuk dipantau, pilih atau masukkan persentase.
6. Pilih Perbarui.

Untuk menghapus monitor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi kiri, di bawah Pemantauan jaringan, pilih Monitor Internet.
3. Pilih monitor Anda, lalu pilih menu tindakan.
4. Pilih Disable (Nonaktifkan).
5. Pilih menu Action lagi, lalu pilih Delete.

Untuk informasi selengkapnya tentang opsi yang dapat Anda perbarui, silakan lihat berikut ini:

- Untuk mempelajari lebih lanjut tentang sumber daya yang Anda tambahkan di Monitor Internet, silakan lihat Menambahkan sumber daya ke monitor Anda.
- Untuk mempelajari selengkapnya tentang persentase lalu lintas aplikasi, silakan lihat Memilih persentase lalu lintas aplikasi untuk dipantau.
- Untuk informasi selengkapnya tentang mengubah ambang batas untuk peristiwa kondisi kesehatan, silakan lihat Mengubah ambang batas peristiwa kondisi kesehatan.
- Untuk mempelajari lebih lanjut tentang batas maksimum jaringan kota, silakan lihat Memilih batas maksimum jaringan kota.
- Untuk mempelajari lebih lanjut tentang pemilihan untuk menerbitkan event ke S3, silakan lihat Menerbitkan pengukuran internet ke Amazon S3 di Amazon CloudWatch Internet Monitor.

Tambahkan atau buat monitor Amazon CloudWatch Internet Monitor dengan Amazon VPC

Saat Anda membuat VPC Amazon Virtual Private Cloud di AWS Management Console, Anda dapat memilih untuk juga mengatur pemantauan untuk itu di Amazon CloudWatch Internet Monitor. Anda dapat menambahkan VPC ke monitor yang ada, atau Anda dapat memilih untuk membuat monitor baru untuk VPC di konsol VPC Amazon.

Dengan menggunakan Monitor Internet dengan VPC Anda, Anda dapat melihat dan mengevaluasi pengukuran dan metrik tentang ketersediaan, performa, byte tertransfer yang dipantau, dan tempo pulang-pergi yang spesifik ke lokasi klien dan ASN di aplikasi Anda (biasanya penyedia layanan internet). Monitor Internet juga memberitahukan kapan terjadi anomali dalam performa dan ketersediaan dan membuat peristiwa kondisi kesehatan di monitor Anda, yang dapat Anda pilih agar mendapat pemberitahuan. Untuk mempelajari lebih lanjut tentang cara menggunakan monitor untuk mengelola dan meningkatkan pengalaman klien Anda dengan aplikasi Anda, silakan lihat Menggunakan sebuah monitor di Monitor Internet.

> **Important**
>
> Untuk membuat monitor, atau menambahkan VPC ke monitor yang ada, Anda harus memiliki izin yang benar. Untuk informasi selengkapnya, lihat Identity and Access Management untuk Amazon CloudWatch Internet Monitor.

Tambahkan VPC ke monitor yang ada

Anda dapat memilih untuk meminta Amazon CloudWatch Internet Monitor menambahkan VPC baru ke monitor yang ada untuk Anda saat Anda membuat VPC di. AWS Management Console Setelah Anda menambahkan VPC, tunggu beberapa menit, dan kemudian metrik untuk VPC akan mulai ditampilkan di konsol Monitor Internet.

Anda dapat mengedit monitor kapan saja, untuk menghapus VPC atau menambahkan VPC lain atau sumber daya lainnya. Anda juga dapat mengubah persentase lalu lintas yang Anda pantau, atau membuat perubahan lainnya. Jika Anda memilih untuk menghapus VPC dari monitor, lalu lintas dari klien ke VPC tersebut tidak lagi dipantau oleh Monitor Internet.

Untuk mempelajari lebih lanjut tentang memperbarui monitor, lihatMengedit atau menghapus sebuah monitor di Monitor Internet.

Buat monitor untuk VPC

Jika Anda memilih untuk membuat monitor untuk VPC, pemandu Buat monitor akan memandu Anda selangkah demi selangkah. Anda menambahkan VPC sebagai sumber daya yang dipantau saat Anda membuat monitor. Jika suka, Anda juga dapat memilih persentase lalu lintas klien yang ingin Anda pantau untuk aplikasi Anda (defaultnya adalah 100%).

Anda dapat mempelajari lebih lanjut dengan meninjau informasi di Membuat monitor di Amazon CloudWatch Internet Monitor menggunakan konsol.

Penetapan Harga

Dengan Amazon CloudWatch Internet Monitor, Anda hanya membayar untuk apa yang Anda gunakan. Penetapan Harga untuk Monitor Internet memiliki dua komponen: biaya sumber daya per pemantauan dan biaya per jaringan kota. Jaringan kota adalah lokasi di mana klien mengakses sumber daya aplikasi Anda dari dan jaringan (ASN, seperti penyedia layanan internet atau ISP) yang klien mengakses sumber daya melalui.

Untuk informasi selengkapnya, termasuk contoh harga, lihat Harga untuk Amazon CloudWatch Internet Monitor

Berhenti memantau VPC

Jika Anda ingin berhenti memantau sumber daya VPC Anda dengan Internet Monitor, lakukan hal berikut di konsol Internet Monitor:

Untuk menghapus sumber daya dari monitor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi kiri, di bawah Pemantauan jaringan, pilih Monitor Internet.
3. Pilih monitor Anda, lalu pilih menu tindakan.
4. Pilih Perbarui monitor.
5. Pada Sumber daya tambahan, pilih Hapus sumber daya.
6. Pilih VPC untuk dihapus, lalu pilih Hapus.
7. Pilih Perbarui.

Tambahkan atau buat monitor Amazon CloudWatch Internet Monitor dengan CloudFront

Di dasbor metrik untuk distribusi di CloudFront konsol Amazon, Anda dapat mengatur pemantauan tambahan untuk distribusi di Amazon CloudWatch Internet Monitor. Anda dapat menambahkan distribusi ke monitor yang ada, atau Anda dapat membuat monitor baru untuk distribusi.

Dengan menggunakan Internet Monitor dengan CloudFront distribusi Anda, Anda dapat melihat dan mengevaluasi pengukuran dan metrik tentang ketersediaan, kinerja, byte yang dipantau yang ditransfer, dan waktu pulang-pergi yang spesifik untuk lokasi klien dan ASN aplikasi Anda (biasanya penyedia layanan internet). Monitor Internet juga memberitahukan kapan terjadi anomali dalam performa dan ketersediaan dan membuat peristiwa kondisi kesehatan di monitor Anda, yang dapat Anda pilih agar mendapat pemberitahuan. Untuk mempelajari lebih lanjut tentang cara menggunakan monitor untuk mengelola dan meningkatkan pengalaman klien Anda dengan aplikasi Anda, silakan lihat Menggunakan sebuah monitor di Monitor Internet.

> **Important**
>
> Untuk membuat monitor, atau menambahkan distribusi ke monitor yang ada, Anda harus memiliki izin yang benar. Untuk informasi selengkapnya, lihat Identity and Access Management untuk Amazon CloudWatch Internet Monitor.

Tambahkan distribusi ke monitor yang ada

Anda dapat memilih agar Internet Monitor menambahkan distribusi ke monitor yang ada langsung dari dasbor CloudFront metrik di AWS Management Console. Setelah Anda menambahkan distribusi, tunggu beberapa menit, dan kemudian metrik untuk distribusi akan mulai ditampilkan di konsol Monitor Internet.

Anda dapat mengedit monitor kapan saja, untuk menghapus distribusi atau menambahkan distribusi lain atau sumber daya lainnya. Anda juga dapat mengubah persentase lalu lintas yang Anda pantau, atau membuat perubahan lainnya. Jika Anda memilih untuk menghapus distribusi dari monitor, lalu lintas dari klien ke distribusi itu tidak lagi dipantau oleh Internet Monitor.

Untuk mempelajari lebih lanjut tentang memperbarui monitor, lihatMengedit atau menghapus sebuah monitor di Monitor Internet.

Buat monitor untuk distribusi

Jika Anda memilih untuk membuat monitor untuk distribusi, wizard Buat monitor memandu Anda melalui langkah-langkahnya. Anda menambahkan distribusi sebagai sumber daya yang dipantau saat Anda membuat monitor. Jika suka, Anda juga dapat memilih persentase lalu lintas klien yang ingin Anda pantau untuk aplikasi Anda (defaultnya adalah 100%).

Anda dapat mempelajari lebih lanjut dengan meninjau informasi di Membuat monitor di Amazon CloudWatch Internet Monitor menggunakan konsol.

Penetapan Harga

Dengan Amazon CloudWatch Internet Monitor, Anda hanya membayar untuk apa yang Anda gunakan. Penetapan Harga untuk Monitor Internet memiliki dua komponen: biaya sumber daya per pemantauan dan biaya per jaringan kota. Jaringan kota adalah lokasi di mana klien mengakses sumber daya aplikasi Anda dari dan jaringan (ASN, seperti penyedia layanan internet atau ISP) yang klien mengakses sumber daya melalui.

Untuk informasi selengkapnya, termasuk contoh harga, lihat Harga untuk Amazon CloudWatch Internet Monitor

Berhenti memantau distribusi

Jika Anda ingin berhenti memantau sumber daya distribusi Anda dengan Internet Monitor, lakukan hal berikut di konsol Internet Monitor:

Untuk menghapus sumber daya dari monitor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi kiri, di bawah Pemantauan jaringan, pilih Monitor Internet.
3. Pilih monitor Anda, lalu pilih menu tindakan.
4. Pilih Perbarui monitor.
5. Pada Sumber daya tambahan, pilih Hapus sumber daya.
6. Pilih distribusi yang akan dihapus, lalu pilih Hapus.
7. Pilih Perbarui.

Contoh menggunakan CLI dengan Amazon Internet Monitor CloudWatch

Bagian ini mencakup contoh untuk menggunakan operasi AWS Command Line Interface dengan Amazon CloudWatch Internet Monitor.

Sebelum memulai, pastikan Anda masuk untuk menggunakan AWS akun yang sama yang memiliki Amazon Virtual Private Clouds (VPC), Network Load Balancers, CloudFront distribusi Amazon, atau WorkSpaces direktori Amazon yang ingin Anda pantau. AWS CLI Monitor Internet tidak mendukung pengaksesan sumber daya di seluruh akun. Untuk informasi selengkapnya tentang penggunaan AWS CLI, lihat Referensi AWS CLI Perintah. Untuk informasi selengkapnya tentang penggunaan tindakan API dengan Amazon CloudWatch Internet Monitor, lihat Panduan Referensi API Amazon CloudWatch Internet Monitor.

Topik

Membuat sebuah monitor

Saat Anda membuat monitor di Monitor Internet, beri nama dan kaitkan sumber daya dengan monitor untuk menunjukkan di mana lalu lintas internet aplikasi Anda berada. Anda tentukan persentase lalu lintas yang merincikan berapa banyak lalu lintas aplikasi Anda yang terpantau. Hal itu juga menentukan jumlah jaringan kota, yaitu lokasi klien dan ASN, biasanya penyedia layanan internet atau ISP, yang dipantau. Anda juga dapat memilih untuk menetapkan batasan jumlah maksimum jaringan kota untuk memantau sumber daya aplikasi Anda, untuk membantu mengontrol tagihan Anda. Untuk informasi selengkapnya, lihat Memilih batas maksimum jaringan kota.

Terakhir, Anda dapat memilih apakah Anda ingin menerbitkan semua pengukuran internet untuk aplikasi Anda ke Amazon S3. Pengukuran internet untuk 500 jaringan kota teratas (berdasarkan volume lalu lintas) secara otomatis dipublikasikan ke CloudWatch Log oleh Internet Monitor, tetapi Anda dapat memilih untuk mempublikasikan semua pengukuran ke S3 juga.

Untuk membuat monitor dengan AWS CLI, Anda menggunakan `create-monitor` perintah. Perintah berikut membuat monitor yang memantau 100% lalu lintas namun menetapkan batas maksimum jaringan kota sebanyak 10.000, menambahkan sumber daya VPC, dan memilih untuk menerbitkan pengukuran internet ke Amazon S3.

> **Note**
>
> Internet Monitor menerbitkan ke CloudWatch Log pengukuran internet setiap lima menit untuk 500 jaringan kota teratas (lokasi klien dan ASN, biasanya penyedia layanan internet atau ISP) yang mengirim lalu lintas ke setiap monitor. Secara opsional, Anda dapat memilih untuk menerbitkan pengukuran internet untuk semua jaringan kota yang dipantau (hingga 500.000 batas layanan jaringan kota) ke bucket Amazon S3. Untuk informasi selengkapnya, lihat Menerbitkan pengukuran internet ke Amazon S3 di Amazon CloudWatch Internet Monitor.

```
aws internetmonitor --create-monitor monitor-name "TestMonitor" \
    --traffic-percentage-to-monitor 100 \
    --max-city-networks-to-monitor 10000 \
    --resources "arn:aws:ec2:us-east-1:111122223333:vpc/vpc-11223344556677889" \
    --internet-measurements-log-delivery
 S3Config="{BucketName=MyS3Bucket,LogDeliveryStatus=ENABLED}"
```

```
{
"Arn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/TestMonitor",
 "Status": "ACTIVE"
}
```

> **Note**
>
> Anda tidak dapat mengubah nama monitor.

Melihat detail monitor

Untuk melihat informasi tentang monitor dengan AWS CLI, Anda menggunakan `get-monitor` perintah.

```
aws internetmonitor get-monitor --monitor-name "TestMonitor"
```

```
{
    "ClientLocationType": "city",
    "CreatedAt": "2022-09-22T19:27:47Z",
    "ModifiedAt": "2022-09-22T19:28:30Z",
    "MonitorArn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/TestMonitor",
    "MonitorName": "TestMonitor",
    "ProcessingStatus": "OK",
    "ProcessingStatusInfo": "The monitor is actively processing data",
    "Resources": [
        "arn:aws:ec2:us-east-1:111122223333:vpc/vpc-11223344556677889"
    ],
    "MaxCityNetworksToMonitor": 10000,
    "Status": "ACTIVE"
}
```

Membuat daftar peristiwa kondisi kesehatan

Ketika performa menurun untuk lalu lintas internet aplikasi Anda, Monitor Internet membuat peristiwa kondisi kesehatan di monitor Anda. Untuk melihat daftar peristiwa kesehatan saat ini dengan AWS CLI, gunakan `list-health-events` perintah

```
aws internetmonitor list-health-events --monitor-name "TestMonitor"
```

```
{
    "HealthEvents": [
        {
            "EventId": "2022-06-20T01-05-05Z/latency",
            "Status": "RESOLVED",
            "EndedAt": "2022-06-20T01:15:14Z",
            "ServiceLocations": [
                {
                    "Name": "us-east-1"
                }
            ],
            "PercentOfTotalTrafficImpacted": 1.21,
            "ClientLocations": [
                {
                    "City": "Lockport",
                    "PercentOfClientLocationImpacted": 60.370000000000005,
                    "PercentOfTotalTraffic": 2.01,
                    "Country": "United States",
                    "Longitude": -78.6913,
```

```
                    "AutonomousSystemNumber": 26101,
                    "Latitude": 43.1721,
                    "Subdivision": "New York",
                    "NetworkName": "YAHOO-BF1"
                }
            ],
            "StartedAt": "2022-06-20T01:05:05Z",
            "ImpactType": "PERFORMANCE",
            "EventArn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/
TestMonitor/health-event/2022-06-20T01-05-05Z/latency"
        },
        {
            "EventId": "2022-06-20T01-17-56Z/latency",
            "Status": "RESOLVED",
            "EndedAt": "2022-06-20T01:30:23Z",
            "ServiceLocations": [
                {
                    "Name": "us-east-1"
                }
            ],
            "PercentOfTotalTrafficImpacted": 1.29,
            "ClientLocations": [
                {
                    "City": "Toronto",
                    "PercentOfClientLocationImpacted": 75.32,
                    "PercentOfTotalTraffic": 1.05,
                    "Country": "Canada",
                    "Longitude": -79.3623,
                    "AutonomousSystemNumber": 14061,
                    "Latitude": 43.6547,
                    "Subdivision": "Ontario",
                    "CausedBy": {
                        "Status": "ACTIVE",
                        "Networks": [
                            {
                                "AutonomousSystemNumber": 16509,
                                "NetworkName": "Amazon.com"
                            }
                        ],
                        "NetworkEventType": "AWS"
                    },
                    "NetworkName": "DIGITALOCEAN-ASN"
                },
                {
```

```
                    "City": "Lockport",
                    "PercentOfClientLocationImpacted": 22.91,
                    "PercentOfTotalTraffic": 2.01,
                    "Country": "United States",
                    "Longitude": -78.6913,
                    "AutonomousSystemNumber": 26101,
                    "Latitude": 43.1721,
                    "Subdivision": "New York",
                    "NetworkName": "YAHOO-BF1"
                },
                {
                    "City": "Hangzhou",
                    "PercentOfClientLocationImpacted": 2.88,
                    "PercentOfTotalTraffic": 0.7799999999999999,
                    "Country": "China",
                    "Longitude": 120.1612,
                    "AutonomousSystemNumber": 37963,
                    "Latitude": 30.2994,
                    "Subdivision": "Zhejiang",
                    "NetworkName": "Hangzhou Alibaba Advertising Co.,Ltd."
                }
            ],
            "StartedAt": "2022-06-20T01:17:56Z",
            "ImpactType": "PERFORMANCE",
            "EventArn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/
TestMonitor/health-event/2022-06-20T01-17-56Z/latency"
        },
        {
            "EventId": "2022-06-20T01-34-20Z/latency",
            "Status": "RESOLVED",
            "EndedAt": "2022-06-20T01:35:04Z",
            "ServiceLocations": [
                {
                    "Name": "us-east-1"
                }
            ],
            "PercentOfTotalTrafficImpacted": 1.15,
            "ClientLocations": [
                {
                    "City": "Lockport",
                    "PercentOfClientLocationImpacted": 39.45,
                    "PercentOfTotalTraffic": 2.01,
                    "Country": "United States",
                    "Longitude": -78.6913,
```

```
                    "AutonomousSystemNumber": 26101,
                    "Latitude": 43.1721,
                    "Subdivision": "New York",
                    "NetworkName": "YAHOO-BF1"
                },
                {
                    "City": "Toronto",
                    "PercentOfClientLocationImpacted": 29.770000000000003,
                    "PercentOfTotalTraffic": 1.05,
                    "Country": "Canada",
                    "Longitude": -79.3623,
                    "AutonomousSystemNumber": 14061,
                    "Latitude": 43.6547,
                    "Subdivision": "Ontario",
                    "CausedBy": {
                        "Status": "ACTIVE",
                        "Networks": [
                            {
                                "AutonomousSystemNumber": 16509,
                                "NetworkName": "Amazon.com"
                            }
                        ],
                        "NetworkEventType": "AWS"
                    },
                    "NetworkName": "DIGITALOCEAN-ASN"
                },
                {
                    "City": "Hangzhou",
                    "PercentOfClientLocationImpacted": 2.88,
                    "PercentOfTotalTraffic": 0.7799999999999999,
                    "Country": "China",
                    "Longitude": 120.1612,
                    "AutonomousSystemNumber": 37963,
                    "Latitude": 30.2994,
                    "Subdivision": "Zhejiang",
                    "NetworkName": "Hangzhou Alibaba Advertising Co.,Ltd."
                }
            ],
            "StartedAt": "2022-06-20T01:34:20Z",
            "ImpactType": "PERFORMANCE",
            "EventArn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/
TestMonitor/health-event/2022-06-20T01-34-20Z/latency"
        }
    ]
```

```
}
```

Lihat peristiwa kondisi kesehatan spesifik

Untuk melihat informasi lebih rinci tentang peristiwa kondisi kesehatan spesifik dengan CLI, jalankan perintah `get-health-event` dengan nama monitor Anda dan ID peristiwa kondisi kesehatan.

```
aws internetmonitor get-monitor --monitor-name "TestMonitor" --event-id "health-event/
TestMonitor/2021-06-03T01:02:03Z/latency"
```


```
{
    "EventId": "2022-06-20T01-34-20Z/latency",
    "Status": "RESOLVED",
    "EndedAt": "2022-06-20T01:35:04Z",
    "ServiceLocations": [
        {
            "Name": "us-east-1"
        }
    ],
    "EventArn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/TestMonitor/
health-event/2022-06-20T01-34-20Z/latency",
    "LastUpdatedAt": "2022-06-20T01:35:04Z",
    "ClientLocations": [
        {
            "City": "Lockport",
            "PercentOfClientLocationImpacted": 39.45,
            "PercentOfTotalTraffic": 2.01,
            "Country": "United States",
            "Longitude": -78.6913,
            "AutonomousSystemNumber": 26101,
            "Latitude": 43.1721,
            "Subdivision": "New York",
            "NetworkName": "YAHOO-BF1"
        },
        {
            "City": "Toronto",
            "PercentOfClientLocationImpacted": 29.770000000000003,
            "PercentOfTotalTraffic": 1.05,
            "Country": "Canada",
            "Longitude": -79.3623,
            "AutonomousSystemNumber": 14061,
            "Latitude": 43.6547,
            "Subdivision": "Ontario",
```

```
            "CausedBy": {
                "Status": "ACTIVE",
                "Networks": [
                    {
                        "AutonomousSystemNumber": 16509,
                        "NetworkName": "Amazon.com"
                    }
                ],
                "NetworkEventType": "AWS"
            },
            "NetworkName": "DIGITALOCEAN-ASN"
        },
        {
            "City": "Shenzhen",
            "PercentOfClientLocationImpacted": 4.07,
            "PercentOfTotalTraffic": 0.61,
            "Country": "China",
            "Longitude": 114.0683,
            "AutonomousSystemNumber": 37963,
            "Latitude": 22.5455,
            "Subdivision": "Guangdong",
            "NetworkName": "Hangzhou Alibaba Advertising Co.,Ltd."
        },
        {
            "City": "Hangzhou",
            "PercentOfClientLocationImpacted": 2.88,
            "PercentOfTotalTraffic": 0.7799999999999999,
            "Country": "China",
            "Longitude": 120.1612,
            "AutonomousSystemNumber": 37963,
            "Latitude": 30.2994,
            "Subdivision": "Zhejiang",
            "NetworkName": "Hangzhou Alibaba Advertising Co.,Ltd."
        }
    ],
    "StartedAt": "2022-06-20T01:34:20Z",
    "ImpactType": "PERFORMANCE",
    "PercentOfTotalTrafficImpacted": 1.15
}
```


Tampilkan daftar monitor

Untuk melihat daftar semua monitor di akun Anda dengan CLI, jalankan perintah `list-monitors`.

```
aws internetmonitor list-monitors
```

```
{
    "Monitors": [
        {
            "MonitorName": "TestMonitor",
            "ProcessingStatus": "OK",
            "Status": "ACTIVE"
        }
    ],
    "NextToken": " zase12"
}
```

Mengedit monitor

Untuk memperbarui informasi tentang monitor Anda dengan menggunakan CLI, gunakan perintah `update-monitor` dan tentukan nama monitor yang akan diperbarui. Anda dapat memperbarui persentase lalu lintas untuk dipantau, batasan jumlah maksimum jaringan kota untuk dipantau, menambah atau menghapus sumber daya yang Monitor Internet gunakan untuk memantau lalu lintas, dan mengubah status monitor dari `ACTIVE` menjadi `INACTIVE`, atau sebaliknya. Perhatikan bahwa Anda tidak dapat mengubah nama monitor.

Respons untuk call `update-monitor` menampilkan hanya `MonitorArn` dan `Status`.

Contoh berikut menunjukkan cara penggunaan perintah `update-monitor` untuk mengubah jumlah maksimum jaringan kota untuk dipantau menjadi `50000`:

```
aws internetmonitor update-monitor --monitor-name "TestMonitor" --max-city-networks-to-
monitor 50000
```

```
{
    "MonitorArn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/TestMonitor",
    "Status": " ACTIVE "
}
```

Contoh berikut menunjukkan cara menambahkan dan menghapus sumber daya:

```
aws internetmonitor update-monitor --monitor-name "TestMonitor" \
    --resources-to-add "arn:aws:ec2:us-east-1:111122223333:vpc/vpc-11223344556677889" \
    --resources-to-remove "arn:aws:ec2:us-east-1:111122223333:vpc/vpc-2222444455556666"
```

```
{
    "MonitorArn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/TestMonitor",
    "Status": "ACTIVE"
}
```

Contoh berikut menunjukkan cara penggunaan perintah `update-monitor` untuk mengubah jumlah status monitor menjadi `INACTIVE`:

```
aws internetmonitor update-monitor --monitor-name "TestMonitor" --status "INACTIVE"
```

```
{
    "MonitorArn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/TestMonitor",
    "Status": "INACTIVE"
}
```

Hapus monitor

Anda dapat menghapus monitor dengan CLI dengan menggunakan perintah `delete-monitor`. Pertama, Anda harus mengatur monitor menjadi tidak aktif. Untuk melakukan itu, gunakan perintah `update-monitor` untuk mengubah status menjadi `INACTIVE`. Konfirmasikan bahwa monitor tidak aktif dengan menggunakan perintah `get-monitor` dan periksa statusnya.

Ketika status monitor `INACTIVE`, maka Anda dapat menggunakan CLI untuk menjalankan perintah `delete-monitor` untuk menghapus monitor. Respons untuk panggilan `delete-monitor` yang berhasil adalah kosong.

```
aws internetmonitor delete-monitor --monitor-name "TestMonitor"
```

```
{}
```

Pantau dan optimalkan dengan dasbor Monitor Internet

Informasi di bagian ini menjelaskan cara memfilter dan melihat informasi di dasbor Amazon CloudWatch Internet Monitor untuk memvisualisasikan dan mendapatkan wawasan tentang lalu lintas dan penyiapan internet AWS aplikasi Anda.

Setelah Anda membuat monitor untuk memantau kinerja dan ketersediaan internet aplikasi Anda, Amazon CloudWatch Internet Monitor menerbitkan CloudWatch log yang berisi pengukuran internet

untuk pasangan jaringan lokasi klien (jaringan kota), dan menerbitkan CloudWatch metrik agregat untuk lalu lintas ke aplikasi Anda, dan ke setiap lokasi tepi. Wilayah AWS Anda dapat memfilter, menjelajahi, dan mendapatkan saran berorientasi tindakan dari informasi ini dari Monitor Internet dengan beberapa cara yang berbeda.

Untuk memulai, di CloudWatch konsol, di bawah Pemantauan jaringan, pilih Monitor Internet.

Bagian ini terutama menjelaskan cara memfilter dan melihat metrik Internet Monitor menggunakan. AWS Management Console Atau, Anda dapat menggunakan operasi Internet Monitor API dengan AWS CLI atau SDK untuk bekerja secara langsung dengan peristiwa Internet Monitor yang disimpan dalam file CloudWatch Log. Untuk informasi selengkapnya, silakan lihat Penggunan monitor dan informasi pengukuran. Untuk informasi selengkapnya tentang penggunaan operasi API, lihat Contoh menggunakan CLI dengan Amazon Internet Monitor CloudWatch dan Referensi API Amazon CloudWatch Internet Monitor.

Terdapat tiga tab di dasbor Monitor Internet:

- Pada tab Gambaran umum, Anda dapat melihat informasi performa dan ketersediaan terkini dan berdasarkan riwayat tentang aplikasi Anda, dan peristiwa kondisi kesehatan yang memengaruhi lokasi klien Anda.
- Pada tab berikutnya, penjelajah berdasarkan riwayat, Anda dapat memfilter berdasarkan lokasi, ASN, tanggal, dan sebagainya, dan memvisualisasikan metrik untuk lalu lintas internet Anda dari waktu ke waktu, menggunakan grafik.
- Pada tab Wawasan lalu lintas, tak hanya dapat melihat informasi mengenai lalu lintas terpantau paling atas yang diringkas dalam beberapa cara yang dapat dikustomisasi, Anda juga bisa mendapat pengaturan teroptimasi untuk meningkatkan performa untuk lokasi dan pasangan ASN yang berbeda. Internet Monitor memprediksi peningkatan kinerja aplikasi Anda, berdasarkan pola lalu lintas dan kinerja masa lalu, ketika Anda mengubah cara Anda merutekan lalu lintas atau AWS sumber daya yang Anda gunakan. Anda juga dapat melihat sebuah grafik untuk membandingkan berapa banyak jaringan kota yang disertakan dalam cakupan pemantauan Anda, berdasarkan persentase lalu lintas aplikasi yang Anda pilih untuk monitor Anda.

Selain itu, karena Internet Monitor menghasilkan dan menerbitkan file log dengan pengukuran tentang lalu lintas Anda, Anda dapat menggunakan CloudWatch alat lain di konsol untuk lebih memvisualisasikan data yang diterbitkan oleh Internet Monitor, termasuk Wawasan CloudWatch Kontributor, CloudWatch Metrik, dan Wawasan Log. CloudWatch Untuk informasi selengkapnya, lihat Menjelajahi data Anda dengan CloudWatch alat dan antarmuka kueri Internet Monitor.

Pelajari cara penggunaan Monitor Internet untuk menjelajahi pengukuran performa dan ketersediaan Anda di bagian berikut.

Topik

Melacak kinerja dan ketersediaan real-time di Amazon CloudWatch Internet Monitor (tab Ikhtisar)

Gunakan tab Ikhtisar di CloudWatch konsol, di bawah Monitor Internet, untuk mendapatkan tampilan kinerja dan ketersediaan tingkat tinggi untuk lalu lintas yang dilacak monitor Anda. Tab ini juga menampilkan peta gambaran umum lalu lintas internet, dengan klaster lalu lintas yang dapat membantu Anda memvisualisasikan lalu lintas global aplikasi Anda, serta lokasi serta dampak peristiwa kondisi kesehatan.

Skor Kesehatan

Grafik Skor Kesehatan menunjukkan informasi kinerja dan ketersediaan untuk lalu lintas global Anda. AWS memiliki data historis yang substansif tentang kinerja internet dan ketersediaan untuk lalu lintas jaringan antara lokasi geografis untuk ASN dan AWS layanan yang berbeda. Internet Monitor menggunakan data konektivitas yang AWS telah diambil dari jejak jaringan globalnya untuk menghitung dasar kinerja dan ketersediaan untuk lalu lintas internet. Ini adalah data yang sama yang kami gunakan AWS untuk memantau uptime dan ketersediaan internet kami sendiri.

Dengan pengukuran tersebut sebagai baseline, Monitor Internet dapat mendeteksi kapan performa dan ketersediaan aplikasi Anda menurun, dibandingkan dengan baseline-nya. Untuk mempermudah melihat penurunan, kami melaporkan informasi tersebut kepada Anda dalam bentuk skor kesehatan dan skor ketersediaan. Untuk informasi selengkapnya, lihat Menjelajahi data Anda dengan CloudWatch alat dan antarmuka kueri Internet Monitor.

Grafik Skor Kesehatan mencakup peristiwa kondisi kesehatan yang terjadi selama periode waktu yang Anda pilih. Ketika ada sebuah peristiwa kondisi kesehatan, terdapat penurunan pada performa atau garis ketersediaan pada grafik. Jika Anda memilih sebuah event, Anda akan melihat detail lebih lanjut dan garis pita muncul pada grafik, dengan informasi tanggal dan waktu yang menunjukkan berapa lama event berlangsung.

Anda juga dapat melihat metrik ini dengan mengakses berkas log secara langsung untuk setiap titik data. Di menu Tindakan, pilih Lihat CloudWatch Log.

Gambaran umum lalu lintas internet

Peta gambaran umum lalu lintas Internet menunjukkan kepada Anda lalu lintas internet dan peristiwa kondisi kesehatan yang spesifik di lokasi dan ASN tempat para pengguna Anda mengakses aplikasi Anda. Negara-negara yang berwarna abu-abu di peta adalah negara-negara yang menyertakan lalu lintas untuk aplikasi Anda.

Setiap lingkaran pada peta menunjukkan peristiwa kondisi kesehatan di suatu area, untuk jangka waktu yang Anda pilih. Internet Monitor menciptakan peristiwa kesehatan ketika mendeteksi masalah, pada ambang tertentu, dengan konektivitas antara salah satu sumber daya Anda yang dihosting AWS dan jaringan kota tempat pengguna mengakses aplikasi Anda. Memilih lingkaran di peta akan menampilkan detail lebih lanjut tentang peristiwa kondisi kesehatan di lokasi tersebut. Selain itu, untuk klaster dengan peristiwa kondisi kesehatan, Anda dapat melihat informasi terperinci dalam tabel event Kesehatan di bawah peta.

Perhatikan bahwa Monitor Internet mengeluarkan peristiwa kondisi kesehatan di monitor ketika Monitor Internet mendeteksi bahwa suatu event memiliki dampak global yang signifikan di aplikasi Anda. Jika tidak ada peristiwa kondisi kesehatan yang melebihi ambang batas untuk dampak pada lalu lintas di lokasi klien dalam periode waktu yang Anda pilih, maka peta kosong. Untuk informasi selengkapnya, silakan lihat Kapan Monitor Internet membuat dan menyelesaikan peristiwa kondisi kesehatan.

Ubah ambang batas peristiwa kondisi kesehatan

Anda dapat mengonfigurasi beberapa opsi seputar bagaimana dan kapan Monitor Internet membuat peristiwa kondisi kesehatan untuk aplikasi Anda. Pilih Perbarui ambang batas untuk membuat perubahan.

Anda dapat mengubah ambang batas keseluruhan yang memicu Monitor Internet untuk membuat sebuah peristiwa kondisi kesehatan. Angka default untuk ambang batas peristiwa kondisi kesehatan, baik skor performa maupun skor ketersediaan, adalah 95%. Artinya, ketika skor performa atau skor ketersediaan secara keseluruhan pada aplikasi Anda turun ke 95% atau lebih rendah dari itu maka Monitor Internet akan membuat peristiwa kondisi kesehatan. Untuk ambang batas keseluruhan, peristiwa kondisi kesehatan dapat dipicu oleh satu masalah besar, atau kombinasi dari beberapa masalah yang lebih kecil.

Anda juga dapat mengubah ambang batas lokal — yaitu jaringan kota —, dikombinasikan dengan persentase tingkat dampak keseluruhan, kombinasi keduanya akan memicu terbuatnya peristiwa

kondisi kesehatan. Dengan menetapkan ambang batas yang memunculkan peristiwa kondisi kesehatan saat skor turun di bawah ambang batas pada satu atau beberapa jaringan kota (lokasi dan ASN, biasanya ISP), Anda bisa mendapatkan wawasan tentang kapan ada masalah di lokasi yang lalu lintas yang lebih rendah misalnya.

Opsi ambang batas lokal tambahan berkolaborasi dengan ambang batas lokal untuk skor ketersediaan atau skor performa. Faktor kedua adalah persentase lalu lintas Anda secara keseluruhan yang harus terdampak sebelum Monitor Internet membuat peristiwa kondisi kesehatan berdasarkan ambang batas lokal.

Dengan mengonfigurasi opsi ambang batas untuk lalu lintas keseluruhan dan lalu lintas lokal, Anda dapat menyempurnakan seberapa sering peristiwa kondisi kesehatan dibuat, agar selaras dengan penggunaan aplikasi dan kebutuhan Anda. Ketahuilah bahwa ketika Anda menetapkan ambang batas lokal menjadi lebih rendah, biasanya akan ada lebih banyak peristiwa kondisi kesehatan yang muncul, tergantung pada aplikasi Anda dan nilai konfigurasi ambang batas lainnya yang Anda tetapkan.

Singkatnya, Anda dapat mengonfigurasi ambang batas peristiwa kondisi kesehatan—untuk skor performa, skor ketersediaan, atau keduanya—dengan cara berikut:

- Pilih ambang batas global yang berbeda yang memicu peristiwa kondisi kesehatan.
- Pilih ambang batas lokal yang berbeda yang memicu peristiwa kondisi kesehatan. Dengan opsi ini, Anda juga dapat mengubah persentase dampak pada keseluruhan aplikasi Anda yang harus dilampaui sebelum Monitor Internet memunculkan sebuah event.
- Pilih nonaktifkan pemicu peristiwa kondisi kesehatan berdasarkan ambang batas lokal, atau aktifkan opsi ambang batas lokal.

Anda juga dapat mengonfigurasi opsi untuk skor performa, skor ketersediaan, atau keduanya. Anda dapat mengonfigurasi kombinasi opsi, atau hanya salah satunya.

Untuk memperbarui ambang batas dan opsi konfigurasi lainnya untuk skor performa, skor ketersediaan, atau keduanya, lakukan hal berikut:

Untuk mengubah opsi konfigurasi ambang batas

1. Di AWS Management Console, navigasikan ke CloudWatch, dan kemudian, di panel navigasi kiri, pilih Internet Monitor.
2. Pada tab Gambaran umum, di bagian Timeline peristiwa kondisi kesehatan, pilih Perbarui ambang batas.

3. Pada halaman dialog yang terbuka, pilih nilai dan opsi baru yang Anda inginkan untuk ambang batas dan opsi lain yang memicu Monitor Internet untuk membuat peristiwa kondisi kesehatan. Anda dapat melakukan langkah-langkah berikut:
   - Pilih nilai baru untuk ambang batas skor Ketersediaan, Ambang batas skor performa, atau keduanya.

     Grafik ada di bagian yang setiap pengaturan menampilkan pengaturan ambang batas saat ini dan skor peristiwa kondisi kesehatan terkini yang sebenarnya atas skor ketersediaan atau skor performa di aplikasi Anda Dengan melihat nilai-nilai yang serupa, Anda bisa mendapatkan gambaran nilai-nilai yang mungkin ingin Anda ubah ambang batasnya.

     Tip: Untuk melihat grafik yang lebih besar dan mengubah jangka waktu, pilih ekspander di pojok kanan atas grafik.
   - Pilih untuk mengaktifkan atau menonaktifkan ambang batas lokal untuk ketersediaan atau performa, atau keduanya. Saat sebuah opsi diaktifkan, Anda dapat mengatur ambang batas dan tingkat dampak saat Anda ingin Monior Internet membuat peristiwa kondisi kesehatan.
4. Setelah Anda mengonfigurasi opsi ambang batas, simpan pembaruan Anda dengan memilih Perbarui ambang batas peristiwa kondisi kesehatan.

Untuk mempelajari lebih lanjut tentang cara kerja peristiwa kondisi kesehatan, silakan lihat Kapan Monitor Internet membuat dan mengatasi peristiwa kondisi kesehatan.

Tabel peristiwa kondisi kesehatan

Tabel event Kesehatan mencantumkan lokasi klien yang telah terpengaruh oleh peristiwa kondisi kesehatan, beserta informasi tentang event tersebut. Kolom berikut sudah termasuk ke dalam tabel.

	Deskripsi
Lokasi klien	Lokasi pengguna akhir yang terkena dampak event, yang mengalami peningkatan latensi atau penurunan ketersediaan. Untuk mempelajari lebih lanjut tentang keakurasian lokasi klien di Monitor Internet,

	Deskripsi
	silakan lihat Informasi dan akurasi geolokasi di Monitor Internet.
Dampak lalu lintas	Seberapa besar dampak yang disebabkan oleh event tersebut, saat peningkatan latensi atau berkurangnya ketersediaan. Untuk latensi, ini adalah persentase dari berapa banyak latensi yang meningkat selama acara dibandingkan dengan kinerja tipikal untuk lalu lintas, dari lokasi klien ini ke AWS lokasi ini menggunakan jaringan klien ini.
Jaringan klien	Jaringan yang dilalui lalu lintas. Biasanya, ini adalah penyedia layanan internet (ISP) atau Nomor Sistem Otonom (ASN) untuk lalu lintas jaringan.
AWS lokasi	AWS Lokasi untuk lalu lintas jaringan, yang dapat berupa Wilayah AWS atau lokasi tepi internet.

	Deskripsi
Jenis dampak	Jenis dampak peristiwa kondisi kesehatan . peristiwa kondisi kesehatan biasanya disebabkan oleh peningkatan latensi (masalah performa) atau keterjangkauan (masalah ketersediaan). Anda mungkin juga dapat mengklik jenis dampak untuk melihat penyebab kerusakan . Jika memungkinkan, Internet Monitor menganalisis asal mula peristiwa kesehatan , untuk menentukan apakah itu disebabka n oleh AWS atau ASN (penyedia layanan internet). Perhatikan bahwa analisis ini berlanjut setelah event diselesaikan. Monitor Internet dapat memperbarui event dengan informasi baru hingga satu jam.

Jika Anda memilih salah satu lokasi klien di tabel event Kesehatan, Anda dapat melihat detail lebih lanjut tentang peristiwa kondisi kesehatan di lokasi tersebut. Misalnya, Anda dapat melihat kapan event dimulai, kapan event berakhir, dan dampak lalu lintas lokal.

Visualisasi jalur jaringan

Analisis gangguan yang lengkap memiliki jalur jaringan lengkap pada Visualisasi jalur jaringan. Jalur lengkap menunjukkan kepada Anda setiap node di sepanjang jalur jaringan untuk aplikasi Anda untuk acara kesehatan, antara AWS lokasi dan klien, untuk pasangan lokasi-klien.

Jika Monitor Internet menemukan penyebab gangguan, penyebab gangguan ditandai dengan lingkaran merah putus-putus. Kerusakan dapat disebabkan oleh ASN, biasanya penyedia layanan internet (ISP), atau penyebabnya bisa jadi AWS. Jika ada beberapa penyebab gangguan, beberapa simpul akan dilingkari.

Memfilter dan melihat data historis di Amazon CloudWatch Internet Monitor (tab penjelajah sejarah)

Gunakan tab Historical explorer di CloudWatch konsol, di bawah Internet Monitor, untuk memfilter dan melihat data untuk aplikasi Anda yang ada di CloudWatch Log. Internet Monitor menerbitkan pengukuran ke CloudWatch Log khusus untuk aplikasi Anda untuk ketersediaan, kinerja, byte yang dipantau yang ditransfer (atau jumlah koneksi klien, hanya untuk WorkSpaces direktori), dan waktu pulang-pergi untuk jaringan kota yang dipantau di. Wilayah AWS

> **Note**
>
> Internet Monitor menerbitkan pengukuran internet ke CloudWatch Log setiap lima menit untuk 500 (berdasarkan volume lalu lintas) jaringan kota teratas (yaitu, lokasi klien dan ASN, biasanya penyedia layanan internet atau ISP) yang mengirim lalu lintas ke setiap monitor. Secara opsional, Anda dapat memilih untuk menerbitkan pengukuran internet untuk semua jaringan kota yang dipantau (hingga 500.000 batas layanan jaringan kota) ke bucket Amazon S3. Untuk informasi selengkapnya, lihat Menerbitkan pengukuran internet ke Amazon S3 di Amazon CloudWatch Internet Monitor.

Untuk mulai menjelajahi data aplikasi Anda, pilihlah sebuah periode waktu. Kemudian, pilih lokasi geografis tertentu, seperti kota, dan filter lainnya (opsional). Monitor Internet menerapkan filter ke data dalam log pengukuran internet yang telah menerbitkan jaringan kota untuk lalu lintas aplikasi Anda. Kemudian Anda dapat melihat grafik data yang menunjukkan skor kinerja, skor ketersediaan, byte yang dipantau yang ditransfer (untuk VPC, Network Load Balancers, dan CloudFront distribusi) atau jumlah koneksi klien (untuk WorkSpaces direktori), dan waktu pulang-pergi (RTT) untuk aplikasi Anda dari waktu ke waktu.

Tabel Semua event di bawah grafik menunjukkan kepada Anda peristiwa kondisi kesehatan yang filter Anda perlihatkan lagi atas lalu lintas aplikasi Anda, dengan informasi tentang setiap event. Tabel ini termasuk kolom berikut.

	Deskripsi
event dimulai	Waktu ketika peristiwa kondisi kesehatan dimulai.

	Deskripsi
Status	Apakah event tersebut masih aktif atau sudah terselesaikan.
Lokasi klien	Lokasi pengguna akhir yang terkena dampak event, yang mengalami peningkatan latensi atau penurunan performa. Untuk mempelajari lebih lanjut tentang keakurasian lokasi klien di Monitor Internet, silakan lihat Informasi dan akurasi geolokasi di Monitor Internet.
Dampak lalu lintas	Dampak tertimbang event di lokasi peristiwa kondisi kesehatan. Yaitu, misalnya, dampak pada latensi, dibandingkan dengan kinerja tipikal untuk lalu lintas dari lokasi klien ke AWS lokasi melalui ASN klien, biasanya penyedia layanan internet (ISP). Demikian pula, untuk acara yang memengaruhi ketersediaan, Anda melihat dampaknya pada ketersediaan dibandingkan dengan ketersediaan umum untuk lokasi klien untuk AWS lokasi di atas ASN klien.
Durasi event	Berapa lama event tersebut berlangsung. Monitor Internet menyudahi peristiwa kondisi kesehatan ketika event tidak lagi memengaruhi lebih dari 5% (total) lokasi klien aplikasi Anda.
ISP Klien	ASN, biasanya penyedia layanan internet (ISP), adalah operator untuk lalu lintas jaringan.
Lokasi layanan	Lokasi layanan tempat lalu lintas jaringan berasal, yang dapat berupa lokasi tepi internet Wilayah AWS atau internet.

Atau, Anda dapat melihat pengukuran di aplikasi Anda dengan mengakses log secara langsung untuk setiap titik data. Di menu Tindakan, pilih Lihat CloudWatch Log. Perhatikan bahwa karena peristiwa pengukuran dipublikasikan ke akun Anda saat dibuat, Anda juga dapat membuat CloudWatch dasbor atau alarm lain berdasarkan peristiwa tersebut. Untuk informasi selengkapnya, silakan lihat Mendapatkan wawasan untuk meningkatkan kinerja aplikasi di Amazon CloudWatch Internet Monitor (tab wawasan lalu lintas) dan Membuat alarm dengan Amazon CloudWatch Internet Monitor.

Selain menjelajahi dan menganalisis pengukuran dan metrik Monitor Internet, serta membuat dasbor dan alarm berdasarkan pengukuran dan metrik, Anda dapat menggunakan Monitor Internet untuk membantu Anda memahami cara-cara yang dapat meningkatkan performa di aplikasi Anda. Tab Wawasan lalu lintas memiliki beberapa cara untuk membantu Anda menjelajahi opsi. Untuk informasi selengkapnya, silakan lihat Saran optimisasi pada tab lWawasan lalu lintas. Selain itu, Anda dapat melihat contoh spesifik di Bab Kasus Penggunaan Monitor Internet.

Mendapatkan wawasan untuk meningkatkan kinerja aplikasi di Amazon CloudWatch Internet Monitor (tab wawasan lalu lintas)

Gunakan tab Traffic insights di CloudWatch konsol, di bawah Internet Monitor, untuk melihat informasi ringkasan untuk lalu lintas teratas (berdasarkan volume) untuk aplikasi Anda. Anda dapat memfilter dan mengurutkan lalu lintas aplikasi Anda dengan berbagai cara. Kemudian, gulir ke bawah, dan pilih kombinasi pengaturan yang berbeda untuk aplikasi Anda untuk melihat apa yang disarankan Monitor Internet sebagai alternatif terbaik untuk mendapatkan performa waktu untuk mendapat byte pertama (TTFB) paling cepat.

Internet Monitor menerbitkan ke CloudWatch Log pengukuran internet setiap lima menit untuk 500 (berdasarkan volume lalu lintas) jaringan kota teratas (yaitu, lokasi klien dan ASN, biasanya penyedia layanan internet atau ISP) yang mengirim lalu lintas ke setiap monitor. Secara opsional, Anda dapat memilih untuk menerbitkan pengukuran internet untuk semua jaringan kota yang dipantau (hingga 500.000 batas layanan jaringan kota) ke bucket Amazon S3. Untuk informasi selengkapnya, lihat Menerbitkan pengukuran internet ke Amazon S3 di Amazon CloudWatch Internet Monitor.

Ringkasan lalu lintas teratas

Anda dapat memulai dengan melihat ringkasan tingkat tinggi dari keseluruhan lalu lintas dan performa aplikasi Anda, selama periode waktu tertentu, yang difilter berdasarkan lokasi klien. Anda juga dapat melihat performa aplikasi Anda untuk lokasi klien teratas (atau terbawah) berdasarkan volume lalu lintas, difilter dan diurutkan dengan berbagai cara. Misalnya, Anda

bisa mengurutkan berdasarkan pedetail wilayah (yaitu, kota, daerah, negara, atau area metro), berdasarkan total lalu lintas, waktu untuk mendapat byte pertama (TTFB) rata-rata, dan faktor lainnya.

Untuk mempelajari lebih lanjut tentang keakurasian lokasi klien di Monitor Internet, silakan lihat Informasi dan akurasi geolokasi di Monitor Internet.

> **Note**
>
> Filter yang Anda gunakan berlaku untuk seluruh halaman, sehingga filter akan memengaruhi jaringan kota mana yang termasuk ke dalam grafik dan informasi ringkasan untuk lalu lintas total, dan juga jaringan kota mana yang termasuk dalam bagian Saran optimisasi lalu lintas yang berikutnya.

Saran optimisasi lalu lintas

Bagian saran optimisasi lalu lintas menampilkan serangkaian jaringan kota yang dipantau yang difilter (filter lokasi dan ASN, penyedia layanan internet) untuk lalu lintas Anda, bersama dengan total lalu lintas klien untuk masing-masing. Entri dalam tabel didasarkan pada filter yang Anda pilih untuk lalu lintas aplikasi Anda untuk wawasan Lalu Lintas di bagian atas halaman. Bawaannya adalah 10 kota teratas berdasarkan volume lalu lintas. Anda biasanya akan melihat lebih dari 10 baris dalam tabel, karena ada entri untuk setiap pasangan jaringan kota yang unik. Artinya, ada satu baris untuk setiap kombinasi lokasi (kota) dan ASN (penyedia jaringan) tempat klien mengakses aplikasi Anda—seperti, Dallas, Texas, AS, dan Comcast, misalnya.

> **Note**
>
> Untuk melihat saran pengoptimalan lalu lintas untuk semua jaringan kota yang dipantau, Anda dapat menjalankan kueri langsung di Wawasan. CloudWatch Untuk contoh kueri yang tidak memasukkan filter pedetail wilayah geografis yang membatasi daftar jaringan kota di halaman ini, silakan lihat Menggunakan Wawasan CloudWatch Log dengan Amazon CloudWatch Internet Monitor.

Di bagian ini, pilih opsi yang berbeda: Amazon EC2, CloudFront, atau keduanya. Ini memungkinkan Anda melihat nilai prediksi waktu rata-rata ke byte pertama (TTFB) untuk klien saat Anda menggunakan aplikasi Anda dengan layanan tersebut di AWS Wilayah yang berbeda,

dibandingkan dengan TTFB saat ini. Untuk informasi selengkapnya tentang perhitungan TTFB, silakan lihat AWS perhitungan untuk TTFB dan latensi.

Dengan memilih opsi yang berbeda, dan kemudian melihat hasilnya di tabel, Anda dapat mulai merencanakan pengaturan dan deployment yang dapat meningkatkan performa untuk klien Anda. Perlu Anda perhatikan bahwa Anda mungkin akan melihat tanda hubung (-) dalam sebuah kolom alih-alih nilai, saat tidak ada data yang tersedia untuk ditampilkan. Untuk meninjau contoh spesifik tentang cara meningkatkan kinerja, lihat Menggunakan Monitor CloudWatch Internet Amazon untuk Pengalaman Bermain Game yang Lebih Baik.

Misalnya, untuk memulai, untuk jaringan kota tertentu (lokasi klien dan pasangan ASN), bereksperimen dengan memilih EC2 atau CloudFront opsi, atau keduanya. Untuk setiap jaringan kota yang tercantum dalam tabel, Internet Monitor menunjukkan kepada Anda potensi peningkatan kinerja TTFB, berdasarkan pilihan perutean lalu lintas (melalui yang spesifik Wilayah AWS) dengan opsi itu, dibandingkan dengan pengaturan saat ini. (Perhatikan bahwa, untuk kelengkapan, tabel juga memasukkan rute yang sudah dioptimisasi.) Misalnya, Anda mungkin melihat prediksi TTFB rata-rata yaitu 50ms dengan menggunakan perutean EC2 melalui `us-east-1` dibandingkan dengan pengaturan Anda saat ini dengan TTFB 100 ms di mana Anda menggunakan perutean EC2 melalui `us-west-2`. Jadi, Anda dapat mempertimbangkan perutean melalui `us-west-2`.

Sebagai contoh lain, Anda mungkin memilih EC2, dan kemudian melihat bahwa itu tidak membuat perbedaan kinerja yang terukur untuk satu lokasi klien dan ASN, tetapi kemudian perhatikan bahwa ketika Anda memilih CloudFront dengan Wilayah yang sama, itu sedikit menurunkan TTFB. Ini menunjukkan bahwa jika Anda menambahkan CloudFront distribusi di depan aplikasi Anda, itu dapat menghasilkan peningkatan kinerja dan mungkin patut dicoba, untuk lokasi klien dan ASN ini.

Menjelajahi data Anda dengan CloudWatch alat dan antarmuka kueri Internet Monitor

Selain memvisualisasikan kinerja dan ketersediaan aplikasi Anda dengan dasbor Amazon CloudWatch Internet Monitor, ada beberapa metode yang dapat Anda gunakan untuk menyelam lebih dalam ke data yang dihasilkan Internet Monitor untuk Anda. Metode ini termasuk menggunakan CloudWatch alat dengan data Monitor Internet yang disimpan dalam file CloudWatch Log dan menggunakan antarmuka kueri Internet Monitor. Alat yang dapat Anda gunakan termasuk Wawasan CloudWatch Log, CloudWatch Metrik, Wawasan CloudWatch Kontributor, dan Amazon Athena.

Anda dapat menggunakan beberapa atau semua alat ini, beserta dasbornya, untuk menjelajahi data Monitor Internet, tergantung pada kebutuhan Anda.

Internet Monitor menggabungkan CloudWatch metrik tentang lalu lintas ke aplikasi Anda dan masing-masing Wilayah AWS, dan mencakup data seperti dampak lalu lintas total, ketersediaan, dan waktu pulang pergi. Data ini dipublikasikan ke CloudWatch Log dan juga tersedia untuk digunakan dengan antarmuka query Internet Monitor. Detail tentang geo-pedetail wilayah dan aspek lain dari informasi yang tersedia untuk dijelajahi masing-masing bervariasi.

Amazon CloudWatch Internet Monitor menerbitkan data untuk monitor Anda pada interval 5 menit, dan kemudian membuat data tersedia dalam beberapa cara. Tabel berikut mencantumkan skenario untuk mengakses data Monitor Internet, dan menjelaskan fitur data yang dikumpulkan untuk masing-masing data.

Fitur	CloudWatch Log	Ekspor ke S3	Antarmuka kueri	CloudWatch dasbor
Diaktifkan secara default	Ya	Tidak	Ya	Ya
Jumlah jaringan kota tempat data dikumpulkan	500 Teratas (lihat catatan di bawah)	Semua	Semua	Semua
Retensi data	Pengguna terkendali	Pengguna terkendali	30 hari	30 hari
Geo-pedetail wilayah tempat data dikumpulkan	Semua (jaringan kota, jaringan metro+, jaringan daerah+, jaringan negara +)	Jaringan kota	Semua (jaringan kota, jaringan metro+, jaringan daerah+, jaringan negara +)	Semua (jaringan kota, jaringan metro+, jaringan daerah+, jaringan negara +)
Cara menjalankan kueri dan memfilter data	Menggunakan Wawasan CloudWatch Log dengan Amazon	Menggunakan Amazon Athena untuk menjalankan kueri pengukura	Menggunakan antarmuka kueri Amazon CloudWatch Internet Monitor	Pantau dan optimalkan dengan dasbor Monitor Internet

Fitur	CloudWatch Log	Ekspor ke S3	Antarmuka kueri	CloudWatch dasbor
	CloudWatch Internet Monitor	n internet di berkas log Amazon S3		

Catatan: 500 pengukuran teratas diambil untuk jaringan kota; 250 teratas untuk jaringan metro+, 100 teratas untuk jaringan daerah+, 50 teratas untuk jaringan negara+).

Bab ini menjelaskan cara menanyakan dan menjelajahi data Anda dengan menggunakan CloudWatch alat atau antarmuka kueri Internet Monitor, bersama dengan contoh untuk setiap metode.

Daftar Isi

Menggunakan Wawasan CloudWatch Log dengan Amazon CloudWatch Internet Monitor

Amazon CloudWatch Internet Monitor menerbitkan pengukuran terperinci ketersediaan dan waktu pulang-pergi ke CloudWatch Log, dan Anda dapat menggunakan kueri Wawasan CloudWatch Log untuk memfilter subset log untuk kota atau geografi tertentu (lokasi klien), ASN klien (ISP), dan lokasi sumber. AWS

Untuk mempelajari lebih lanjut tentang keakurasian lokasi klien di Monitor Internet, silakan lihat Informasi dan akurasi geolokasi di Monitor Internet.

Contoh di bagian ini dapat membantu Anda membuat kueri Wawasan CloudWatch Log untuk mempelajari lebih lanjut tentang pengukuran dan metrik lalu lintas aplikasi Anda sendiri. Jika Anda menggunakan contoh ini di Wawasan CloudWatch Log, ganti *`MonitorName dengan nama monitor`* Anda sendiri.

Lihat saran optimisasi lalu lintas

Pada tab Wawasan lalu lintas di Monitor Internet, Anda dapat melihat saran optimisasi lalu lintas, yang difilter berdasarkan lokasi. Untuk melihat informasi yang sama yang ditampilkan di bagian Saran pengoptimalan lalu lintas pada tab itu, tetapi tanpa filter granularitas lokasi, Anda dapat menggunakan kueri Wawasan CloudWatch Log berikut.

1. Di AWS Management Console, navigasikan ke Wawasan CloudWatch Log.
2. Untuk Grup Log, pilih `/aws/internet-monitor/`*`monitorName`*`/byCity` dan`/aws/internet-monitor/`*`monitorName`*`/byCountry`, lalu tentukan rentang waktu.
3. Tambahkan kueri berikut, dan kemudian jalankan kueri.

```
fields @timestamp,
clientLocation.city as @city, clientLocation.subdivision as @subdivision,
 clientLocation.country as @country,
`trafficInsights.timeToFirstByte.currentExperience.serviceName` as @serviceNameField,
concat(@serviceNameField, ` (`, `serviceLocation`, `)`) as @currentExperienceField,
concat(`trafficInsights.timeToFirstByte.ec2.serviceName`, ` (`,
 `trafficInsights.timeToFirstByte.ec2.serviceLocation`, `)`) as @ec2Field,
`trafficInsights.timeToFirstByte.cloudfront.serviceName` as @cloudfrontField,
concat(`clientLocation.networkName`, ` (AS`, `clientLocation.asn`, `)`) as @networkName
| filter ispresent(`trafficInsights.timeToFirstByte.currentExperience.value`)
| stats avg(`trafficInsights.timeToFirstByte.currentExperience.value`) as @averageTTFB,
avg(`trafficInsights.timeToFirstByte.ec2.value`) as @ec2TTFB,
avg(`trafficInsights.timeToFirstByte.cloudfront.value`) as @cloudfrontTTFB,
sum(`bytesIn` + `bytesOut`) as @totalBytes,
latest(@ec2Field) as @ec2,
latest(@currentExperienceField) as @currentExperience,
latest(@cloudfrontField) as @cloudfront,
count(*) by @networkName, @city, @subdivision, @country
| display @city, @subdivision, @country, @networkName, @totalBytes, @currentExperience,
 @averageTTFB, @ec2, @ec2TTFB, @cloudfront, @cloudfrontTTFB
| sort @totalBytes desc
```

Lihat ketersediaan internet dan RTT (p50, p90, dan p95)

Untuk melihat ketersediaan internet dan waktu pulang-pergi (p50, p90, dan p95) untuk lalu lintas, Anda dapat menggunakan kueri Wawasan Log berikut. CloudWatch

Geografi pengguna akhir: Chicago, IL, Amerika Serikat

Jaringan pengguna akhir (ASN): AS7018

AWS lokasi layanan: Wilayah AS Timur (Virginia N.)

Untuk melihat log, lakukan hal berikut:

1. Di AWS Management Console, navigasikan ke Wawasan CloudWatch Log.
2. Untuk Grup Log, pilih `/aws/internet-monitor/`*`monitorName`*`/byCity` dan`/aws/internet-monitor/`*`monitorName`*`/byCountry`, lalu tentukan rentang waktu.
3. Tambahkan kueri berikut, dan kemudian jalankan kueri.

Kueri menampilkan semua data performa untuk pengguna yang terhubung dari AS7018 di Chicago, IL menuju Wilayah AS Timur (Virginia Utara) selama periode waktu yang dipilih.

```
fields @timestamp,
internetHealth.availability.experienceScore as availabilityExperienceScore,
internetHealth.availability.percentageOfTotalTrafficImpacted as
 percentageOfTotalTrafficImpacted,
internetHealth.performance.experienceScore as performanceExperienceScore,
internetHealth.performance.roundTripTime.p50 as roundTripTimep50,
internetHealth.performance.roundTripTime.p90 as roundTripTimep90,
internetHealth.performance.roundTripTime.p95 as roundTripTimep95
 | filter clientLocation.country == `United States`
 and clientLocation.city == `Chicago`
 and serviceLocation == `us-east-1`
 and clientLocation.asn == 7018
```

Untuk informasi selengkapnya, lihat Menganalisis data CloudWatch log dengan Wawasan Log.

Menggunakan Wawasan Kontributor dengan Amazon Internet Monitor CloudWatch

CloudWatch Contributor Insights dapat membantu Anda mengidentifikasi lokasi dan jaringan klien teratas (ASN atau penyedia layanan internet) untuk aplikasi Anda. Gunakan contoh aturan Contributor Insights berikut untuk memulai aturan yang berguna dengan Amazon CloudWatch Internet Monitor. Untuk informasi selengkapnya, lihat Membuat Aturan Wawasan Kontributor.

Untuk mempelajari lebih lanjut tentang keakurasian lokasi klien di Monitor Internet, silakan lihat Informasi dan akurasi geolokasi di Monitor Internet.

> **Note**
>
> Monitor Internet menerbitkan data setiap lima menit, jadi setelah Anda mengatur aturan Wawasan Kontributor, Anda harus menyesuaikan periode menjadi lima menit untuk melihat grafik.

Lihat lokasi teratas dan ASN yang terkena dampak ketersediaan

Untuk melihat lokasi klien dan ASN teratas yang dipengaruhi oleh penurunan ketersediaan, Anda dapat menggunakan aturan Wawasan Kontributor berikut di editor Sintaks. Ganti *nama monitor* dengan nama monitor Anda sendiri.

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "AggregateOn": "Sum",
    "Contribution": {
        "Filters": [
            {
                "Match": "$.clientLocation.city",
                "IsPresent": true
            }
        ],
        "Keys": [
            "$.clientLocation.city",
            "$.clientLocation.networkName"
        ],
        "ValueOf": "$.awsInternetHealth.availability.percentageOfTotalTrafficImpacted"
    },
    "LogFormat": "JSON",
    "LogGroupNames": [
        "/aws/internet-monitor/monitor-name/byCity"
    ]
}
```

Lihat lokasi dan ASN teratas yang terkena dampak ketersediaan

Untuk melihat lokasi klien dan ASN teratas yang terdampak oleh peningkatan tempo pulang pergi (latensi), Anda dapat menggunakan aturan Wawasan Kontributor berikut di editor Sintaks. Ganti *nama monitor* dengan nama monitor Anda sendiri.

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "AggregateOn": "Sum",
    "Contribution": {
        "Filters": [            {
                "Match": "$.clientLocation.city",
                "IsPresent": true
            }
        ],
        "Keys": [
            "$.clientLocation.city",
            "$.clientLocation.networkName"
        ],
        "ValueOf": "$.awsInternetHealth.performance.percentageOfTotalTrafficImpacted"
    },
    "LogFormat": "JSON",
    "LogGroupNames": [
        "/aws/internet-monitor/monitor-name/byCity"
    ]
}
```

Lihat lokasi klien dan ASN teratas yang terdampak oleh persentase total lalu lintas

Untuk melihat lokasi klien dan ASN teratas yang terdampak penurunan ketersediaan, Anda dapat menggunakan aturan Wawasan Kontributor berikut di editor Sintaks. Ganti *nama monitor* dengan nama monitor Anda sendiri.

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "AggregateOn": "Sum",
    "Contribution": {
        "Filters": [
```

```
            {
                "Match": "$.clientLocation.city",
                "IsPresent": true
            }
        ],
        "Keys": [
            "$.clientLocation.city",
            "$.clientLocation.networkName"
        ],
        "ValueOf": "$.percentageOfTotalTraffic"
    },
    "LogFormat": "JSON",
    "LogGroupNames": [
        "/aws/internet-monitor/monitor-name/byCity"
    ]
}
```

Menggunakan CloudWatch Metrik dengan Amazon CloudWatch Internet Monitor

Amazon CloudWatch Internet Monitor menerbitkan metrik ke akun Anda, termasuk metrik untuk performa, ketersediaan, waktu pulang pergi, dan throughput (byte per detik), yang dapat Anda lihat di Metrik di konsol. CloudWatch CloudWatch Untuk menemukan semua metrik untuk monitor Anda, di dasbor CloudWatch Metrik, lihat namespace khusus. `AWS/InternetMonitor`

Metrik dikumpulkan di semua lalu lintas internet ke VPC Anda, Network Load Balancer, CloudFront distribusi, atau WorkSpaces direktori di monitor, dan ke semua lalu lintas ke masing-masing Wilayah AWS dan lokasi tepi internet yang dipantau. Wilayah ditentukan oleh lokasi layanan, yang dapat berupa semua lokasi atau Wilayah tertentu, seperti `us-east-1`.

Catatan: jaringan kota adalah lokasi klien dan ASN (biasanya penyedia layanan internet atau ISP).

Monitor Internet menyediakan metrik berikut.

Metrik	Deskripsi
PerformanceScore	Skor performa mewakili persentase perkiraan lalu lintas yang tidak terdapat penurunan performa.

Metrik	Deskripsi
AvailabilityScore	Sebuah skor ketersediaan mewakili persentase perkiraan lalu lintas yang tidak terdapat penurunan ketersediaan.
BytesIn	Byte yang masuk melalui lalu lintas internet aplikasi Anda di semua jaringan kota aplikasi.
BytesOut	Byte yang keluar melalui lalu lintas internet aplikasi Anda di semua jaringan kota aplikasi.
BytesInMonitored	Byte yang masuk melalui lalu lintas internet aplikasi Anda di jaringan kota yang dipantau.
BytesOutMonitored	Byte yang keluar melalui lalu lintas internet aplikasi Anda di jaringan kota yang dipantau.
Waktu pulang pergi (RTT)	Waktu pulang-pergi antara Wilayah AWS, ASN (biasanya penyedia layanan internet atau ISP), dan lokasi (seperti kota) khusus untuk VPC, Network Load Balancer, distribusi, atau direktori Anda. CloudFront WorkSpaces
CityNetworksMonitored	Jumlah jaringan kota yang dipantau oleh Monitor Internet atas lalu lintas internet aplikasi Anda. Jumlahnya tidak pernah lebih dari batas atas yang Anda tetapkan sebagai jumlah maksimum jaringan kota untuk monitor.
TrafficMonitoredPercent	Persentase total lalu lintas internet aplikasi untuk monitor ini diwakilkan (termasuk) oleh jaringan kota yang dipantau Monitor Internet. Akan kurang dari 100 (yaitu, kurang dari 100%) jika klien mengakses aplikasi Anda di lebih banyak jaringan kota daripada batas maksimum jaringan kota yang telah Anda tetapkan untuk dipantau.

Metrik	Deskripsi
CityNetworksFor100 PercentTraffic	Jumlah yang harus Anda tetapkan untuk batas maksimum jaringan kota jika Anda ingin memantau 100% lalu lintas internet aplikasi di Monitor Internet.
CityNetworksFor99 PercentTraffic	Jumlah yang harus Anda tetapkan untuk batas maksimum jaringan kota jika Anda ingin memantau 99% lalu lintas internet aplikasi di Monitor Internet.
CityNetworksFor95 PercentTraffic	Jumlah yang harus Anda tetapkan untuk batas maksimum jaringan kota jika Anda ingin memantau 95% lalu lintas internet aplikasi di Monitor Internet.
CityNetworksFor90 PercentTraffic	Jumlah yang harus Anda tetapkan untuk batas maksimum jaringan kota jika Anda ingin memantau 90% lalu lintas internet aplikasi di Monitor Internet.
CityNetworksFor75 PercentTraffic	Jumlah yang harus Anda tetapkan untuk batas maksimum jaringan kota jika Anda ingin memantau 75% lalu lintas internet aplikasi di Monitor Internet.
CityNetworksFor50 PercentTraffic	Jumlah yang harus Anda tetapkan untuk batas maksimum jaringan kota jika Anda ingin memantau 50% lalu lintas internet aplikasi di Monitor Internet.
CityNetworksFor25 PercentTraffic	Jumlah yang harus Anda tetapkan untuk batas maksimum jaringan kota jika Anda ingin memantau 25% lalu lintas internet aplikasi di Monitor Internet.

> **Note**
>
> Untuk melihat contoh penggunaan beberapa metrik ini guna membantu menentukan nilai batas maksimum jaringan kota yang akan dipilih untuk monitor Anda, silakan lihat Memilih nilai batas maksimum jaringan kota.

Untuk informasi selengkapnya, lihat Gunakan CloudWatch metrik Amazon.

Menggunakan Amazon Athena untuk menjalankan kueri pengukuran internet di berkas log Amazon S3

Anda dapat menggunakan Amazon Athena untuk menanyakan dan melihat pengukuran internet yang diterbitkan Amazon CloudWatch Internet Monitor ke bucket Amazon S3. Ada opsi di Monitor Internet untuk menerbitkan pengukuran internet untuk aplikasi Anda ke bucket S3 atas lalu lintas yang dapat diakses internet untuk jaringan kota yang dipantau (lokasi klien dan ASN, biasanya penyedia layanan internet atau ISP). Terlepas dari apakah Anda memilih untuk mempublikasikan pengukuran ke S3, Internet Monitor secara otomatis menerbitkan pengukuran internet ke CloudWatch Log setiap lima menit untuk 500 jaringan kota teratas (berdasarkan volume lalu lintas) untuk setiap monitor.

Bab ini mencakup langkah-langkah cara membuat tabel di Athena untuk pengukuran internet yang terletak di berkas log S3, dan kemudian memberikan contoh kueri untuk melihat tampilan pengukuran yang berbeda. Misalnya, Anda dapat menjalankan kueri untuk 10 jaringan kota teratas yang terdampak berdasarkan dampak latensi.

Menggunakan Amazon Athena untuk membuat tabel untuk pengukuran internet di Monitor Internet

Untuk mulai menggunakan Athena dengan berkas log S3 Monitor Internet Anda, pertama-tama buatlah tabel untuk pengukuran internet.

Ikuti langkah-langkah dalam prosedur ini untuk membuat tabel di Athena berdasarkan berkas log S3. Kemudian, Anda dapat menjalankan kueri Athena di sebuah tabel, misalnya contoh kueri pengukuran internet ini, untuk mendapatkan informasi tentang pengukuran Anda.

Untuk membuat tabel Athena

1. Buka konsol Athena di https://console.aws.amazon.com/athena/.
2. Di editor kueri Athena, masukkan pernyataan kueri untuk menghasilkan tabel dengan pengukuran internet di Monitor Internet. Ganti nilai untuk parameter LOCATION dengan lokasi bucket S3 tempat pengukuran internet disimpan di Monitor Internet Anda.

```
CREATE EXTERNAL TABLE internet_measurements (
    version INT,
    timestamp INT,
    clientlocation STRING,
    servicelocation STRING,
    percentageoftotaltraffic DOUBLE,
    bytesin INT,
    bytesout INT,
    clientconnectioncount INT,
    internethealth STRING,
    trafficinsights STRING
)
PARTITIONED BY (year STRING, month STRING, day STRING)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION
's3://bucket_name/bucket_prefix/AWSLogs/account_id/internetmonitor/AWS_Region/'
TBLPROPERTIES ('skip.header.line.count' = '1');
```

3. Masukkan sebuah pernyataan untuk membuat partisi untuk membaca data. Misalnya, kueri berikut menciptakan partisi tunggal untuk tanggal dan lokasi yang ditentukan:

```
ALTER TABLE internet_measurements
ADD PARTITION (year = 'YYYY', month = 'MM', day = 'dd')
LOCATION
's3://bucket_name/bucket_prefix/AWSLogs/account_id/internetmonitor/AWS_Region/YYYY/
MM/DD';
```

4. Pilih Jalankan.

Contoh pernyataan Athena untuk pengukuran internet

Berikut ini adalah contoh dari pernyataan untuk menghasilkan tabel:

```
CREATE EXTERNAL TABLE internet_measurements (
    version INT,
    timestamp INT,
    clientlocation STRING,
    servicelocation STRING,
    percentageoftotaltraffic DOUBLE,
    bytesin INT,
    bytesout INT,
    clientconnectioncount INT,
```

```
    internethealth STRING,
    trafficinsights STRING
)
PARTITIONED BY (year STRING, month STRING, day STRING)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION 's3://internet-measurements/TestMonitor/AWSLogs/1111222233332/internetmonitor/
us-east-2/'
TBLPROPERTIES ('skip.header.line.count' = '1');
```

Berikut ini adalah contoh dari pernyataan untuk membuat partisi untuk membaca data:

```
ALTER TABLE internet_measurements
ADD PARTITION (year = '2023', month = '04', day = '07')
LOCATION 's3://internet-measurements/TestMonitor/AWSLogs/1111222233332/internetmonitor/
us-east-2/2023/04/07/'
```

Contoh kueri Amazon Athena untuk digunakan dengan pengukuran internet di Internet Monitor

Bagian ini mencakup contoh kueri yang bisa digunakan di Amazon Athena untuk mendapatkan informasi tentang pengukuran internet aplikasi Anda yang dipublikasikan di Amazon S3.

Jalankan kueri untuk 10 lokasi klien dan ASN teratas yang paling terdampak (berdasarkan persentase lalu lintas total)

Jalankan kueri Athena ini untuk menampilkan 10 jaringan kota yang terdampak (berdasarkan persentase lalu lintas total) — yaitu, lokasi klien dan ASN, biasanya penyedia layanan internet.

```
SELECT json_extract_scalar(clientLocation, '$.city') as city,
    json_extract_scalar(clientLocation, '$.networkname') as networkName,
    sum(percentageoftotaltraffic) as percentageoftotaltraffic
FROM internet_measurements
GROUP BY json_extract_scalar(clientLocation, '$.city'),
    json_extract_scalar(clientLocation, '$.networkname')
ORDER BY percentageoftotaltraffic desc
limit 10
```

Jalankan kueri untuk 10 lokasi klien dan ASN teratas Anda yang terdampak (berdasarkan ketersediaan)

Jalankan kueri Athena ini untuk menampilkan 10 jaringan kota yang terdampak (berdasarkan persentase lalu lintas total) — yaitu, lokasi klien dan ASN, biasanya penyedia layanan internet.

```
SELECT json_extract_scalar(clientLocation, '$.city') as city,
    json_extract_scalar(clientLocation, '$.networkname') as networkName,
    sum(
        cast(
            json_extract_scalar(
                internetHealth,
                '$.availability.percentageoftotaltrafficimpacted'
            )
        as double )
    ) as percentageOfTotalTrafficImpacted
FROM internet_measurements
GROUP BY json_extract_scalar(clientLocation, '$.city'),
    json_extract_scalar(clientLocation, '$.networkname')
ORDER BY percentageOfTotalTrafficImpacted desc
limit 10
```

Jalankan kueri untuk 10 lokasi klien dan ASN teratas Anda yang terdampak (berdasarkan latensi)

Jalankan kueri Athena ini untuk menampilkan 10 jaringan kota yang terdampak (berdasarkan latensi) — yaitu, lokasi klien dan ASN, biasanya penyedia layanan internet.

```
SELECT json_extract_scalar(clientLocation, '$.city') as city,
    json_extract_scalar(clientLocation, '$.networkname') as networkName,
    sum(
        cast(
            json_extract_scalar(
                internetHealth,
                '$.performance.percentageoftotaltrafficimpacted'
            )
        as double )
    ) as percentageOfTotalTrafficImpacted
FROM internet_measurements
GROUP BY json_extract_scalar(clientLocation, '$.city'),
    json_extract_scalar(clientLocation, '$.networkname')
ORDER BY percentageOfTotalTrafficImpacted desc
limit 10
```

Lalu lintas kueri menyoroti lokasi klien dan ASN Anda

Jalankan kueri Athena ini untuk menampilkan lalu lintas yang disorot, termasuk skor ketersediaan, skor performa, dan waktu untuk mendapat byte pertama untuk jaringan kota Anda—yaitu, lokasi klien dan ASN, biasanya penyedia layanan internet.

```
SELECT json_extract_scalar(clientLocation, '$.city') as city,
    json_extract_scalar(clientLocation, '$.subdivision') as subdivision,
    json_extract_scalar(clientLocation, '$.country') as country,
    avg(cast(json_extract_scalar(internetHealth, '$.availability.experiencescore') as
 double)) as availabilityScore,
    avg(cast(json_extract_scalar(internetHealth, '$.performance.experiencescore') as
 double)) performanceScore,
    avg(cast(json_extract_scalar(trafficinsights,
 '$.timetofirstbyte.currentexperience.value') as double)) as averageTTFB,
    sum(bytesIn) as bytesIn,
    sum(bytesOut) as bytesOut,
    sum(bytesIn + bytesOut) as totalBytes
FROM internet_measurements
where json_extract_scalar(clientLocation, '$.city') != 'N/A'
GROUP BY
json_extract_scalar(clientLocation, '$.city'),
    json_extract_scalar(clientLocation, '$.subdivision'),
    json_extract_scalar(clientLocation, '$.country')
ORDER BY totalBytes desc
limit 100
```

Untuk informasi selengkapnya tentang penggunaan Athena, silakan lihat Panduan Pengguna Amazon Athena.

Menggunakan antarmuka kueri Amazon CloudWatch Internet Monitor

Opsi untuk memahami lebih lanjut tentang lalu lintas internet untuk AWS aplikasi Anda adalah dengan menggunakan antarmuka kueri Amazon CloudWatch Internet Monitor. Untuk menggunakan antarmuka kueri, buatlah sebuah kueri dengan filter data yang Anda pilih, lalu jalankan kueri untuk menampilkan subset data Monitor Internet Anda. Menjelajahi data yang ditampilkan kueri akan memberikan Anda wawasan tentang bagaimana aplikasi Anda berjalan di internet.

Anda dapat menjalankan kueri dan menjelajahi semua metrik yang ditangkap Monitor Internet dengan monitor Anda, termasuk skor ketersediaan dan skor performa, byte tertransfer, tempo pulang pergi, dan waktu untuk mendapat byte pertama (TTFB).

Monitor Internet menggunakan antarmuka kueri untuk menyediakan data yang dapat Anda jelajahi di dasbor konsol Monitor Internet. Dengan menggunakan opsi pelacakan di dasbor—pada tab Penjelajah Riwayat atau tab Wawasan lalu lintas—Anda dapat menjalankan kueri dan memfilter data internet untuk aplikasi Anda.

Jika Anda ingin lebih banyak fleksibilitas untuk menjelajahi dan memfilter data Anda daripada yang disediakan dasbor, Anda dapat menggunakan antarmuka kueri sendiri, dengan menggunakan operasi Internet Monitor API dengan AWS Command Line Interface atau dengan AWS SDK. Bagian ini memperkenalkan jenis-jenis kueri yang dapat Anda gunakan dengan antarmuka kueri, dan filter yang dapat Anda tentukan untuk membuat satu subset data, untuk mendapatkan wawasan tentang lalu lintas internet untuk aplikasi Anda.

Topik

Cara menggunakan antarmuka kueri

Buatlah sebuah kueri dengan antarmuka kueri dengan memilih jenis kueri, kemudian tentukan nilai-nilai filter, untuk menampilkan subset tertentu yang diinginkan atas data berkas log Anda. Kemudian, Anda dapat mengerjakan subset data tersebut untuk memfilter dan mengurutkan lebih lanjut, membuat laporan, dan sebagainya.

Proses kueri bekerja seperti ini:

1. Saat Anda menjalankan sebuah kueri, Monitor Internet menampilkan `query ID` yang unik untuk kueri. Bagian ini menjelaskan jenis-jenis kueri yang tersedia, dan opsi untuk memfilter data dalam kueri. Untuk memahami cara kerjanya, Anda juga dapat meninjau bagian contoh kueri.
2. Anda menentukan ID kueri dengan nama monitor Anda dengan operasi GetQueryResultsAPI untuk mengembalikan hasil data kueri. Setiap jenis query menampilkan satu set bidang data yang berbeda. Untuk mempelajari selengkapnya, silakan lihat Mendapatkan hasil kueri.

Antarmuka kueri memberikan tiga jenis kueri berikut. Setiap jenis kueri menampilkan sekumpulan informasi yang berbeda tentang lalu lintas Anda dari berkas log, sebagaimana yang ditunjukkan.

- Pengukuran: Memberikan skor ketersediaan, skor kinerja, total lalu lintas, dan waktu pulang pergi, dengan interval 5 menit.
- Lokasi teratas: Memberikan skor ketersediaan, skor kinerja, lalu lintas total, dan informasi time to first byte (TTFB), untuk lokasi teratas dan kombinasi ASN yang Anda pantau, berdasarkan volume lalu lintas.

- Detail lokasi teratas: Menyediakan TTFB untuk Amazon CloudFront, konfigurasi Anda saat ini, dan konfigurasi Amazon EC2 berkinerja terbaik, dengan interval 1 jam.

Dengan masing-masing tipe kueri ini, Anda dapat memfilter data dengan lebih baik dengan menentukan satu atau beberapa dari kriteria berikut ini:

- AWS lokasi: Untuk AWS lokasi, Anda dapat menentukan CloudFront atau Wilayah AWS, sepertius-east-2,us-west-2, dan sebagainya.
- ASN: Tentukan sebuah ASN, biasanya merupakan penyedia layanan internet (ISP).
- Lokasi klien: Untuk lokasi, tentukan sebuah kota, metro, subdivisi, atau negara.
- Geo: Tentukan geo untuk beberapa kueri. Ini diperlukan untuk kueri-kueri yang menggunakan jenis kueri Top locations, tetapi tidak diizinkan untuk jenis kueri lainnya. Untuk memahami kapan harus menentukan geo parameter filter, silakan lihat bagian contoh kueri.

Operator yang dapat Anda gunakan untuk memfilter data Anda adalah EQUALS dan NOT_EQUALS. Untuk detail tentang memfilter parameter, lihat operasi FilterParameterAPI.

Untuk melihat detail tentang operasi antarmuka kueri, lihat operasi API berikut di Panduan Referensi API Amazon CloudWatch Internet Monitor:

- Untuk membuat dan menjalankan kueri, lihat operasi StartQueryAPI.
- Untuk menghentikan kueri, lihat operasi StopQueryAPI.
- Untuk mengembalikan data kueri yang telah Anda buat, lihat operasi GetQueryResultsAPI.
- Untuk mengambil status kueri, lihat operasi GetQueryStatusAPI.

Contoh kueri

Untuk membuat kueri yang dapat Anda gunakan untuk mengambil kumpulan data yang difilter dari file log monitor, Anda menggunakan operasi StartQueryAPI. Tentukan jenis kueri dan parameter filter untuk kueri tersebut. Kemudian, ketika Anda menggunakan operasi API antarmuka kueri di Monitor Internet untuk mendapatkan hasil kueri setelah menggunakan kueri, maka Anda mendapatkan subset data yang ingin Anda kerjakan.

Untuk mengilustrasikan cara kerja jenis kueri dan parameter filter, mari kita lihat beberapa contoh.

Contoh 1

Katakanlah Anda ingin mengambil semua data berkas log monitor Anda untuk negara tertentu, kecuali untuk satu kota. Contoh berikut ini menunjukkan parameter filter untuk kueri yang bisa Anda buat dengan operasi `StartQuery` untuk skenario ini.

```
{
   MonitorName: "TestMonitor"
   StartTime: "2023-07-12T20:00:00Z"
   EndTime: "2023-07-12T21:00:00Z"
   QueryType: "MEASUREMENTS"
   FilterParameters: [
      {
       Field: "country",
       Operator: "EQUALS",
       Values: ["Germany"]
      },
      {
       Field: "city",
       Operator: "NOT_EQUALS",
       Values: ["Berlin"]
      },
    ]
}
```

Contoh 2

Contoh lainnya, katakanlah Anda ingin melihat lokasi teratas Anda berdasarkan area metropolitan. Anda dapat menggunakan contoh query berikut untuk skenario ini.

```
{
   MonitorName: "TestMonitor"
   StartTime: "2023-07-12T20:00:00Z"
   EndTime: "2023-07-12T21:00:00Z"
   QueryType: "TOP_LOCATIONS"
   FilterParameters: [
      {
       Field: "geo",
       Operator: "EQUALS",
       Values: ["metro"]
      },
    ]
}
```

Contoh 3

Sekarang katakanlah Anda ingin melihat kombinasi jaringan kota teratas di area metro Los Angeles. Untuk melakukan hal itu, tentukan `geo=city`, kemudian atur `metro` ke Los Angeles. Sekarang kueri akan menampilkan jaringan kota teratas di area metro Los Angeles alih-alih jaringan metro teratas+ secara keseluruhan.

Berikut contoh kueri yang dapat Anda gunakan:

```
{
   MonitorName: "TestMonitor"
   StartTime: "2023-07-12T20:00:00Z"
   EndTime: "2023-07-12T21:00:00Z"
   QueryType: "TOP_LOCATIONS"
   FilterParameters: [
      {
       Field: "geo",
       Operator: "EQUALS",
       Values: ["city"]
      },
      {
       Field: "metro",
       Operator: "EQUALS",
       Values: ["Los Angeles"]
      }
    ]
}
```

Contoh 4

Terakhir, katakanlah Anda ingin memperoleh data TTFB untuk subdivisi tertentu (misalnya, negara bagian AS).

Berikut adalah contoh kueri untuk skenario ini.

```
{
   MonitorName: "TestMonitor"
   StartTime: "2023-07-12T20:00:00Z"
   EndTime: "2023-07-12T21:00:00Z"
   QueryType: "TOP_LOCATION_DETAILS"
   FilterParameters: [
      {
       Field: "subdivision",
```

```
        Operator: "EQUALS",
        Values: ["California"]
       },
     ]
}
```

Mendapatkan hasil kueri

Setelah menentukan kueri, Anda dapat mengembalikan serangkaian hasil dengan kueri dengan menjalankan operasi Internet Monitor API lainnya, GetQueryResults. Saat menjalankan `GetQueryResults`, tentukan ID kueri untuk kueri yang telah Anda tetapkan, bersama dengan nama monitor Anda. `GetQueryResults` akan mengambil data untuk kueri yang telah ditentukan ke dalam rangkaian hasil.

Saat Anda menjalankan sebuah kueri, pastikan kueri telah selesai berjalan sebelum Anda menggunakan `GetQueryResults` untuk melihat hasilnya. Anda dapat menentukan apakah kueri telah selesai dengan menggunakan operasi GetQueryStatusAPI. Ketika `Status` atas kueri adalah `SUCCEEDED`, Anda dapat melanjutkan dengan meninjau hasilnya.

Saat kueri Anda selesai, Anda bisa menggunakan informasi berikut untuk membantu Anda meninjau hasilnya. Setiap jenis kueri yang Anda gunakan untuk membuat kueri harus menyertakan serangkaian bidang data yang unik dari berkas log, seperti yang dijelaskan dalam daftar berikut:

Pengukuran

Jenis kueri `measurements` menampilkan data berikut:

`timestamp, availability, performance, bytes_in, bytes_out, rtt_p50, rtt_p90, rtt_p95`

Lokasi teratas

Jenis kueri `top locations` mengelompokkan data berdasarkan lokasi, dan menyediakan data rata-rata selama periode waktu. Data yang ditampilkan termasuk hal-hal berikut:

`aws_location, city, metro, subdivision, country, asn, availability, performance, bytes_in, bytes_out, current_fbl, best_ec2, best_ec2_region, best_cf_fbl`

Perhatikan bahwa `city`, `metro`, dan `subdivision` hanya ditampilkan jika Anda memilih jenis lokasi untuk bidang `geo` tersebut. Bidang lokasi berikut ini ditampilkan, tergantung pada jenis lokasi yang Anda tentukan untuk `geo`:

```
city = city, metro, subdivision, country
metro = metro, subdivision, country
subdivision = subdivision, country
country = country
```

Detail lokasi teratas

Jenis kueri `top locations details` menampilkan data yang dikelompokkan jam demi jam. Kueri tersebut menampilkan data berikut:

`timestamp, current_service, current_fbl, best_ec2_fbl, best_ec2_region, best_cf_fbl`

Saat Anda menjalankan operasi API `GetQueryResults`, Monitor Internet akan menampilkan respons sebagai berikut:

- Sebuah array string data yang berisikan hasil yang kueri tampilkan. Informasi ditampilkan dalam array yang selaras dengan bidang `Fields`, juga ditampilkan oleh panggilan API. Dengan menggunakan bidang `Fields`, Anda dapat mengurai informasi dari repositori `Data` dan kemudian memfilter lebih lanjut atau mengurutkannya untuk tujuan Anda.
- Sebuah bidang array terdiri dari bidang-bidang tempat kueri menampilkan data (pada respon bidang `Data`). Setiap item dalam array adalah pasangan nama-datatype, seperti `availability_score-float`.

Pemecahan Masalah

Jika kesalahan dikembalikan saat Anda menggunakan operasi API antarmuka kueri, verifikasi bahwa Anda memiliki izin yang diperlukan untuk menggunakan Amazon CloudWatch Internet Monitor. Secara spesifik, pastikan bahwa Anda memiliki izin berikut:

```
internetmonitor:StartQuery
internetmonitor:GetQueryStatus
internetmonitor:GetQueryResults
internetmonitor:StopQuery
```

Izin ini disertakan dalam AWS Identity and Access Management kebijakan yang disarankan untuk menggunakan dasbor Internet Monitor di konsol. Untuk informasi selengkapnya, lihat Izin IAM untuk Amazon Internet Monitor CloudWatch .

Membuat alarm dengan Amazon CloudWatch Internet Monitor

Anda dapat membuat CloudWatch alarm Amazon berdasarkan metrik Amazon CloudWatch Internet Monitor, seperti yang Anda bisa untuk metrik Amazon CloudWatch lainnya.

Misalnya, Anda dapat membuat alarm berdasarkan metrik `PerformanceScore` di Monitor Internet, dan mengonfigurasinya untuk mengirimkan notifikasi ketika metrik turun lebih rendah dari nilai yang Anda pilih. Anda mengonfigurasi alarm untuk metrik Internet Monitor mengikuti pedoman yang sama seperti metrik lainnya CloudWatch .

Berikut ini adalah contoh metrik Monitor Internet yang mungkin Anda pilih untuk membuat alarm:

- PerformanceScore
- AvailabilityScore
- RoundtripTime

Untuk melihat semua metrik yang tersedia untuk Monitor Internet, silakan lihat Menggunakan CloudWatch Metrik dengan Amazon CloudWatch Internet Monitor.

Prosedur berikut memberikan contoh pengaturan alarm PerformanceScoredengan menavigasi ke metrik di dasbor. CloudWatch Kemudian, Anda mengikuti CloudWatch langkah-langkah standar untuk membuat alarm berdasarkan ambang batas yang Anda pilih, dan mengatur pemberitahuan atau memilih opsi lain.

Untuk membuat alarm PerformanceScoredi CloudWatch Metrik

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pilih Metrik, lalu pilih Semua metrik.
3. Filter Monitor Internet dengan memilih `AWS/InternetMonitor`.
4. Pilih MeasurementSource, MonitorName.
5. Dalam daftar, pilih PerformanceScore.
6. Pada GraphedMetricstab, di bawah Tindakan, pilih ikon lonceng untuk membuat alarm berdasarkan ambang batas statis.

Sekarang, ikuti CloudWatch langkah-langkah standar untuk memilih opsi untuk alarm. Misalnya, Anda dapat memilih untuk diberi tahu oleh pesan Amazon SNS PerformanceScorejika berada di bawah nomor ambang tertentu. Atau, atau sebagai tambahan, Anda dapat menambahkan alarm ke dasbor.

Ingatlah hal berikut:

- Metrik Monitor Internet biasanya dihitung dan dipublikasikan dalam waktu 20 menit.
- Ketika Anda membuat alarm berdasarkan metrik Monitor Internet, pastikan bahwa Anda memperhitungkan adanya penundaan singkat sebelum publikasi saat Anda mengatur periode lookback alarm. Kami menyarankan Anda mengonfigurasi Periode Evaluasi dengan periode lookback minimal 25 menit.

Untuk mempelajari lebih lanjut tentang menggunakan CloudWatch alarm dengan Internet Monitor, lihat posting blog berikut: Menggunakan Amazon CloudWatch Internet Monitor untuk meningkatkan pengamatan internet.

Untuk informasi selengkapnya tentang opsi saat Anda membuat CloudWatch alarm, lihatBuat CloudWatch alarm berdasarkan ambang statis.

Menggunakan Amazon CloudWatch Internet Monitor dengan Amazon EventBridge

Peristiwa kesehatan yang dibuat Amazon CloudWatch Internet Monitor untuk masalah jaringan dipublikasikan dengan Amazon EventBridge, sehingga Anda dapat mengirim pemberitahuan tentang degradasi apa pun dalam pengalaman pengguna akhir untuk aplikasi Anda.

Untuk digunakan EventBridge untuk bekerja dengan acara kesehatan Internet Monitor, ikuti panduan di sini.

Untuk mengatur aturan untuk Internet Monitor di EventBridge

1. Dalam AWS Management Console, di EventBridge, pilih Aturan, lalu masukkan nama dan deskripsi. Buat aturan pada bus peristiwa Default.
2. Langkah 2, pilih Lainnya untuk sumber event, dan kemudian, pada Pola event, cocokkan sumber berikut.

```
{
  "source": ["aws.internetmonitor"]
}
```

3. Pada Langkah 3, untuk target, pilih Grup AWS Layanan dan CloudWatch Log, lalu pilih grup log yang ada atau buat yang baru.

4. Tambahkan tanda apapun yang diinginkan, lalu buat aturannya. Ini harus mengisi Grup CloudWatch Log yang Anda pilih dengan acara dari EventBridge.

Untuk informasi selengkapnya tentang cara kerja EventBridge aturan dengan pola peristiwa, lihat pola EventBridge peristiwa Amazon di Panduan EventBridge Pengguna Amazon.

Memecahkan masalah kesalahan CloudWatch akses log dan metrik

Untuk mendukung beberapa fitur, Amazon CloudWatch Internet Monitor harus berinteraksi dengan CloudWatch sumber daya Amazon tertentu, termasuk log dan metrik. Jika Internet Monitor tidak dapat mengakses CloudWatch sumber daya yang memerlukan akses ke, Internet Monitor menetapkan kode status `FAULT_ACCESS_CLOUDWATCH` untuk monitor.

Ada beberapa alasan mengapa monitor Anda bisa memiliki status `FAULT_ACCESS_CLOUDWATCH`. Bagian berikut mencantumkan kemungkinan penyebab kesalahan ini, dan langkah-langkah pemecahan masalah yang disarankan.

Internet Monitor tidak dapat mengakses CloudWatch log di akun Anda

Monitor Internet menerbitkan log diagnostik tentang lalu lintas aplikasi yang monitor Anda lacak. Ini menerbitkan log ini ke grup log di CloudWatch Log di lokasi berikut:`/aws/internet-monitor/`*`monitor_name`*`/[byCity|byMetro|bySubdivision|byCountry]`. Internet Monitor tidak dapat mengakses grup log ini.

Status kesalahan dan solusi potensial:

- PutLogEvents kesalahan pelambatan: Layanan Internet Monitor mungkin telah dibatasi ketika mencoba mempublikasikan log monitor Anda. CloudWatch Tinjau batas throttling yang ditetapkan untuk akun Anda, dan, jika perlu, mintalah peningkatan batas.
- Grup log tidak ditemukan: Nonaktifkan, lalu aktifkan kembali monitor Anda. Mengaktifkan monitor akan memulai ulang pembuatan grup log, yang mungkin akan memperbaiki masalahnya.
- PutLogEvents akses ditolak kesalahan: Hubungi AWS dukungan untuk bantuan.
- PutLogEvents kesalahan tidak diketahui atau umum: Hubungi AWS dukungan untuk bantuan.

Internet Monitor tidak dapat mengakses CloudWatch metrik di akun Anda

Internet Monitor memberikan CloudWatch metrik spesifik tentang lalu lintas aplikasi yang dilacak oleh monitor. Terjadi kesalahan saat Internet Monitor mencoba mengirimkan metrik ini. CloudWatch

Status kesalahan dan solusi potensial:

- PutMetricData kesalahan pelambatan: Layanan Internet Monitor mungkin telah dibatasi ketika mencoba mempublikasikan metrik monitor Anda. CloudWatch Tinjau batas throttling yang ditetapkan untuk akun Anda, dan, jika perlu, mintalah peningkatan batas.
- PutMetricData akses ditolak kesalahan: Hubungi AWS dukungan untuk bantuan.
- PutMetricData kesalahan tidak diketahui atau umum: Hubungi AWS dukungan untuk bantuan.

Perlindungan data dan privasi data dengan Amazon CloudWatch Internet Monitor

Model tanggung jawab AWS bersama berlaku untuk perlindungan data dan privasi data di Amazon CloudWatch Internet Monitor. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS cloud. Anda harus bertanggung jawab untuk memelihara kendali atas konten yang di-hosting di infrastruktur ini. Untuk informasi selengkapnya tentang privasi data, silakan lihat Pertanyaan Umum Privasi Data. Untuk informasi tentang perlindungan data di Eropa, lihat Model Tanggung Jawab AWS Bersama dan posting blog GDPR di Blog AWS Keamanan. Untuk informasi selengkapnya tentang mematuhi persyaratan GDPR, silakan lihat Pusat Peraturan Perlindungan Data Umum (GDPR).

Kami sangat merekomendasikan agar Anda tidak memasukkan informasi identifikasi yang sensitif, misalnya nomor akun, alamat email, atau informasi pribadi lain Anda atau para pengguna, ke dalam bidang formulir kosong. Data apa pun yang Anda masukkan ke Amazon CloudWatch Internet Monitor atau layanan lain mungkin disertakan dalam log diagnostik.

Identity and Access Management untuk Amazon CloudWatch Internet Monitor

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

> **Important**
>
> Sumber daya Monitor Internet berubaha pada tanggal 24 Februari 2023

Jika Anda membuat kebijakan IAM yang memasukkan sumber daya Monitor Internet sebelum 24 Februari 2023, perhatikan perubahan berikut pada sumber daya dan jenis sumber daya Monitor Internet.

- HealthEventssumber daya diubah namanya menjadi HealthEvent.
- Format ARN dan Regex untuk sumber daya diperbarui. HealthEvent
- Format ARN dan Regex untuk sumber daya Monitor diperbarui.
- Izin tingkat sumber daya untuk GetHealthEventtindakan sekarang hanya didukung pada jenis sumber daya. HealthEvent Izin tidak didukung pada sumber daya Monitor.
- TagResource, UntagResource, dan ListTagsForResourceuntuk jenis sumber daya Monitor diperbarui agar diperlukan.

Untuk melihat informasi selengkapnya tentang tindakan, sumber daya, dan kunci kondisi yang dapat Anda tentukan dalam kebijakan untuk mengelola akses ke AWS sumber daya di Monitor Internet, lihat Tindakan, sumber daya, dan kunci kondisi untuk Amazon CloudWatch Internet Monitor.

Daftar Isi

Bagaimana Amazon CloudWatch Internet Monitor bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke Monitor Internet, pelajari fitur IAM apa yang tersedia untuk digunakan dengan Monitor Internet.

Untuk melihat tabel yang menampilkan tampilan tingkat tinggi serupa tentang cara kerja AWS layanan dengan sebagian besar fitur IAM, lihat AWS layanan yang bekerja dengan IAM di Panduan Pengguna IAM.

Fitur IAM yang dapat Anda gunakan dengan Amazon CloudWatch Internet Monitor

Fitur IAM	Dukungan Monitor Internet
Kebijakan berbasis identitas	Ya
Kebijakan berbasis sumber daya	Tidak
Tindakan kebijakan	Ya
Sumber daya kebijakan	Ya
kunci-kunci persyaratan kebijakan (spesifik layanan)	Ya
ACL	Tidak
ABAC (tanda dalam kebijakan)	Parsial
Kredensial sementara	Ya
Izin pengguna utama	Ya
Peran layanan	Tidak
Peran terkait layanan	Ya

Kebijakan berbasis identitas untuk Monitor Internet

Mendukung kebijakan berbasis identitas	Ya

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 pengguna dan peran, di sumber daya mana, dan dengan ketentuan apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan tindakan dan sumber daya yang diizinkan atau ditolak, serta ketentuan terkait jenis tindakan yang diizinkan atau ditolak. Anda tidak

dapat menentukan pengguna utama dalam kebijakan berbasis identitas karena kebijakan ini berlaku untuk pengguna atau peran yang dilampiri kebijakan. Untuk mempelajari semua elemen yang dapat Anda gunakan dalam kebijakan JSON, silakan lihat Referensi elemen kebijakan JSON IAM dalam Panduan Pengguna IAM.

Kebijakan berbasis sumber daya dalam Monitor Internet

Mendukung kebijakan berbasis sumber daya	Tidak

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.

Tindakan kebijakan untuk Monitor Internet

Mendukung tindakan kebijakan	Ya

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

Menyertakan tindakan dalam suatu kebijakan untuk memberikan izin melakukan operasi terkait.

Untuk melihat daftar tindakan Monitor Internet, lihat Tindakan yang ditentukan oleh Amazon CloudWatch Internet Monitor di Referensi Otorisasi Layanan.

Tindakan kebijakan di Monitor Internet menggunakan prefiks berikut sebelum tindakan:

```
internetmonitor
```

Untuk menetapkan secara spesifik beberapa tindakan dalam satu pernyataan, pisahkan tindakan-tindakan tersebut dengan koma.

```
"Action": [
      "internetmonitor:action1",
      "internetmonitor:action2"
         ]
```

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

```
"Action": "internetmonitor:Describe*"
```

Sumber daya kebijakan untuk Monitor Internet

Mendukung sumber daya kebijakan	Ya

Dalam Referensi Otorisasi Layanan, Anda dapat melihat informasi berikut yang terkait dengan Monitor Internet.

- Untuk melihat daftar jenis sumber daya Internet Monitor dan ARNnya, lihat Sumber daya yang ditentukan oleh Amazon CloudWatch Internet Monitor.
- Untuk mempelajari tindakan yang dapat Anda tentukan dengan ARN dari setiap sumber daya, lihat Tindakan yang ditentukan oleh Amazon CloudWatch Internet Monitor.

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

Elemen kebijakan JSON `Resource` menentukan objek atau beberapa 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 di tingkat sumber daya, misalnya operasi pencantuman, gunakan wildcard (*) untuk mengindikasikan bahwa pernyataan tersebut berlaku bagi semua sumber daya.

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

kunci-kunci persyaratan kebijakan untuk Monitor Internet

Mendukung kunci kondisi kebijakan spesifik layanan	Ya

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

Elemen `Condition` (atau blok `Condition`) memungkinkan Anda menentukan kondisi di mana suatu pernyataan akan diterapkan. Elemen `Condition` bersifat opsional. Anda dapat membuat ekspresi kondisional yang menggunakan operator kondisi, misalnya sama dengan atau kurang dari, untuk mencocokkan kondisi dalam kebijakan dengan nilai-nilai yang diminta.

Jika Anda menentukan beberapa elemen `Condition` dalam satu pernyataan, atau beberapa kunci dalam satu elemen `Condition`, AWS akan mengevaluasinya dengan menggunakan operasi `AND` logis. Jika Anda menentukan beberapa nilai untuk satu kunci kondisi, AWS mengevaluasi kondisi menggunakan `OR` operasi logis. Semua kondisi harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan kondisi. Misalnya, Anda dapat memberikan izin kepada pengguna IAM untuk mengakses sumber daya hanya jika izin tersebut mempunyai tag yang sesuai dengan nama pengguna IAM mereka. Untuk informasi selengkapnya, lihat Elemen kebijakan IAM: variabel dan tag di Panduan Pengguna IAM.

AWS mendukung kunci kondisi global dan kunci kondisi khusus layanan. Untuk melihat semua kunci kondisi AWS global, lihat kunci konteks kondisi AWS global di Panduan Pengguna IAM.

Untuk melihat daftar kunci kondisi Monitor Internet, lihat Kunci kondisi untuk Amazon CloudWatch Internet Monitor di Referensi Otorisasi Layanan. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan kunci kondisi, lihat Tindakan yang ditentukan oleh Amazon CloudWatch Internet Monitor.

ACL di Monitor Internet

Mendukung ACL	Tidak

Daftar kontrol akses (ACL) mengontrol pengguna utama (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun kebijakan-kebijakan tersebut tidak menggunakan format dokumen kebijakan JSON.

ABAC dengan Monitor Internet

Mendukung ABAC (tanda dalam kebijakan)	Parsial

Monitor Internet memiliki dukungan parsial untuk tanda dalam kebijakan. Dukungan parsial Ini mendukung pemberian tanda untuk satu sumber daya, monitor.

Untuk menggunakan tag dengan Internet Monitor, gunakan AWS Command Line Interface atau AWS SDK. Penandaan untuk Monitor Internet tidak didukung dengan. AWS Management Console

Untuk mempelajari selengkapnya tentang penggunaan tanda dalam kebijakan secara umum, tinjau informasi berikut.

Kontrol akses berbasis atribut (ABAC) adalah strategi otorisasi yang menentukan izin berdasarkan atribut. Dalam AWS, atribut ini disebut tag. Anda dapat melampirkan tag ke entitas IAM (pengguna atau peran) dan ke banyak AWS sumber daya. Pemberian tanda ke entitas dan sumber daya adalah langkah pertama dari ABAC. Kemudian, rancanglah kebijakan ABAC untuk mengizinkan operasi saat tag milik pengguna utama cocok dengan tag yang ada di sumber daya yang ingin diakses.

ABAC sangat berguna di lingkungan yang berkembang dengan cepat dan berguna di situasi di mana pengelolaan kebijakan menjadi rumit.

Untuk mengendalikan akses berdasarkan tag, berikan informasi tentang tag di elemen kondisi dari kebijakan dengan menggunakan kunci kondisi `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`.

Jika sebuah layanan mendukung ketiga kunci kondisi untuk setiap jenis sumber daya, nilainya adalah Ya untuk layanan tersebut. Jika suatu layanan mendukung ketiga kunci kondisi hanya untuk beberapa jenis sumber daya, nilainya adalah Parsial.

Untuk informasi selengkapnya tentang ABAC, lihat Apa itu ABAC? di Panduan Pengguna IAM. Untuk melihat tutorial yang menguraikan langkah-langkah pengaturan ABAC, silakan lihat Menggunakan kontrol akses berbasis atribut (ABAC) di Panduan Pengguna IAM.

Menggunakan kredensial temporer dengan Monitor Internet

Mendukung kredensial sementara	Ya

Beberapa Layanan AWS tidak berfungsi saat Anda masuk menggunakan kredensi sementara. Untuk informasi tambahan, termasuk yang Layanan AWS bekerja dengan kredensi sementara, lihat Layanan AWS yang bekerja dengan IAM di Panduan Pengguna IAM.

Anda menggunakan kredensi sementara jika Anda masuk AWS Management Console menggunakan metode apa pun kecuali nama pengguna dan kata sandi. Misalnya, ketika Anda mengakses AWS menggunakan tautan masuk tunggal (SSO) perusahaan Anda, proses tersebut secara otomatis membuat kredensi sementara. Anda juga akan membuat kredensial sementara secara otomatis saat masuk ke konsol sebagai pengguna dan kemudian beralih peran. Untuk informasi selengkapnya tentang cara beralih peran, lihat Beralih peran (konsol) di Panduan Pengguna IAM.

Anda dapat membuat kredenal sementara secara manual menggunakan API AWS CLI atau AWS . Anda kemudian dapat menggunakan kredensi sementara tersebut untuk mengakses. AWS AWS merekomendasikan agar Anda secara dinamis menghasilkan kredensi sementara alih-alih menggunakan kunci akses jangka panjang. Untuk informasi selengkapnya, silakan lihat Kredensial keamanan temporer di IAM.

Izin-izin pengguna utama lintas layanan untuk Monitor Internet

Mendukung sesi akses maju (FAS)	Ya

Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Saat Anda menggunakan beberapa layanan, Anda mungkin melakukan tindakan yang kemudian memulai 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, silakan lihat Meneruskan sesi akses.

Peran layanan untuk Monitor Internet

Mendukung peran layanan	Tidak

Peran layanan adalah sebuah peran IAM yang diambil oleh sebuah layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, silakan lihat Membuat sebuah peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

Peran terkait layanan untuk Monitor Internet

Mendukung peran yang terkait layanan	Ya

Peran terkait layanan adalah jenis peran layanan yang ditautkan ke. Layanan AWS Layanan dapat menggunakan 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.

Untuk informasi selengkapnya tentang peran terkait layanan untuk Monitor Internet, silakan lihat Peran terkait layanan untuk Amazon Internet Monitor CloudWatch .

Untuk detail tentang membuat atau mengelola peran terkait layanan secara umum di AWS, lihat AWS layanan yang bekerja dengan IAM. Temukan sebuah layanan dalam tabel yang memiliki `Yes` di kolom Peran terkait layanan. Pilih tautan Ya untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

AWS kebijakan terkelola untuk Amazon CloudWatch Internet Monitor

Kebijakan AWS terkelola adalah kebijakan mandiri yang dibuat dan dikelola oleh AWS. AWS Kebijakan terkelola dirancang untuk memberikan izin bagi banyak kasus penggunaan umum sehingga Anda dapat mulai menetapkan izin kepada pengguna, grup, dan peran.

Perlu diingat bahwa kebijakan AWS terkelola mungkin tidak memberikan izin hak istimewa paling sedikit untuk kasus penggunaan spesifik Anda karena tersedia untuk digunakan semua pelanggan. AWS Kami menyarankan Anda untuk mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola pelanggan yang khusus untuk kasus penggunaan Anda.

Anda tidak dapat mengubah izin yang ditentukan dalam kebijakan AWS terkelola. Jika AWS memperbarui izin yang ditentukan dalam kebijakan AWS terkelola, pembaruan akan memengaruhi semua identitas utama (pengguna, grup, dan peran) yang dilampirkan kebijakan tersebut. AWS kemungkinan besar akan memperbarui kebijakan AWS terkelola saat baru Layanan AWS diluncurkan atau operasi API baru tersedia untuk layanan yang ada.

Untuk informasi selengkapnya, lihat Kebijakan yang dikelola AWS dalam Panduan Pengguna IAM.

AWS kebijakan terkelola: CloudWatchInternetMonitorServiceRolePolicy

Kebijakan ini dilampirkan ke peran terkait layanan yang diberi nama AWSServiceRoleForInternetMonitoruntuk memungkinkan Internet Monitor mengakses sumber daya di akun Anda, seperti sumber daya Amazon Virtual Private Cloud atau Network Load Balancer, sehingga Anda dapat memilihnya saat membuat monitor. Untuk informasi selengkapnya, lihat Peran terkait layanan untuk Amazon Internet Monitor CloudWatch .

Izin IAM untuk Amazon Internet Monitor CloudWatch

Untuk mengakses tindakan untuk bekerja dengan monitor dan data di Amazon CloudWatch Internet Monitor, pengguna harus memiliki izin yang benar.

Untuk informasi selengkapnya tentang keamanan di Amazon CloudWatch, lihatManajemen identitas dan akses untuk Amazon CloudWatch.

Izin untuk akses hanya-baca di Amazon Internet Monitor CloudWatch

Untuk mengakses tindakan hanya-baca agar berfungsi dengan monitor dan data di Amazon CloudWatch Internet Monitor, pengguna harus masuk sebagai pengguna atau peran yang memiliki izin berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "internetmonitor:Get*",
                "internetmonitor:List*",
                "internetmonitor:StartQuery",
                "internetmonitor:StopQuery",
                "logs:DescribeLogGroups",
```

```
                "logs:GetQueryResults",
                "logs:StartQuery",
                "logs:StopQuery"
                ],
            "Resource": "*"
        }
    ]
}
```

Izin untuk akses penuh di Amazon CloudWatch Internet Monitor

Untuk membuat monitor di Amazon CloudWatch Internet Monitor, dan untuk memiliki akses penuh ke tindakan untuk bekerja dengan monitor dan data di Internet Monitor, pengguna harus masuk dengan pengguna atau peran yang memiliki izin berikut:

- Izin-izin untuk membuat sebuah peran terkait layanan yang terkait dengan Monitor Internet. Untuk informasi selengkapnya, lihat Peran terkait layanan untuk Amazon Internet Monitor CloudWatch .
- Izin-izin untuk tindakan yang memungkinkan akses penuh untuk bekerja dengan monitor dan data di Monitor Internet.

> **Note**
>
> Jika Anda membuat sebuah kebijakan izin berbasis identitas yang lebih ketat, pengguna yang memiliki kebijakan tersebut mungkin tidak akan memiliki akses penuh untuk membuat dan bekerja dengan monitor dan data yang ada di Monitor Internet.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "internetmonitor:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
```

```
            "Resource": "arn:aws:iam::*:role/aws-service-role/
internetmonitor.amazonaws.com/AWSServiceRoleForInternetMonitor",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "internetmonitor.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/
internetmonitor.amazonaws.com/AWSServiceRoleForInternetMonitor"
        },
        {
            "Action": [
                "ec2:DescribeVpcs",
                "elasticloadbalancing:DescribeLoadBalancers",
                "workspaces:DescribeWorkspaceDirectories",
                "cloudfront:GetDistribution"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Peran terkait layanan untuk Amazon Internet Monitor CloudWatch

Amazon CloudWatch Internet Monitor menggunakan peran AWS Identity and Access Management terkait layanan (IAM). Sebuah peran terkait layanan adalah jenis peran peran IAM yang unik yang terkait langsung ke Monitor Internet. Peran terkait layanan telah ditentukan sebelumnya oleh Internet Monitor dan mencakup semua izin yang diperlukan layanan untuk memanggil AWS layanan lain atas nama Anda.

Monitor Internet merincikan izin-izin peran terkait layanan, dan kecuali disebutkan sebaliknya, yang hanya Monitor Internet saja yang dapat menjalankan peran tersebut. Izin-izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, serta bahwa kebijakan izin tidak dapat dilampirkan ke entitas IAM lainnya.

Anda dapat menghapus peran tersebut hanya setelah pertama kali menghapus sumber dayanya yang terkait. Pembatasan ini melindungi sumber daya Monitor Internet Anda karena Anda tidak dapat secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, silakan lihat layanan AWS yang bisa digunakan dengan IAM dan carilah layanan yang memiliki opsi Ya di kolom Peran terkait layanan. Pilih Ya dengan sebuah tautan untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Izin-izin peran terkait layanan untuk Monitor Internet

Internet Monitor menggunakan peran terkait layanan bernama. AWSServiceRoleForInternetMonitor Peran ini memungkinkan Internet Monitor untuk mengakses sumber daya di akun Anda, seperti sumber daya Amazon Virtual Private Cloud, CloudFront distribusi Amazon, WorkSpaces direktori Amazon, dan Network Load Balancer, sehingga Anda dapat memilihnya saat membuat monitor.

Peran terkait layanan ini menggunakan kebijakan terkelola. `CloudWatchInternetMonitorServiceRolePolicy`

Peran AWSServiceRoleForInternetMonitorterkait layanan mempercayai layanan berikut untuk mengambil peran:

- `internetmonitor.amazonaws.com`

Untuk melihat izin kebijakan ini, lihat CloudWatchInternetMonitorServiceRolePolicydi Referensi Kebijakan AWS Terkelola.

Membuat sebuah peran terkait layanan untuk Monitor Internet

Anda tidak perlu membuat peran terkait layanan secara manual untuk Monitor Internet. Pertama kali Anda membuat monitor, Internet Monitor menciptakan AWSServiceRoleForInternetMonitoruntuk Anda.

Untuk informasi selengkapnya, silakan lihat Membuat peran terkait layanan dalam Panduan Pengguna IAM.

Mengedit sebuah peran terkait layanan untuk Monitor Internet

Setelah Monitor Internet menciptakan sebuah peran terkait layanan di akun Anda, Anda tidak dapat mengubah nama peran tersebut karena beragam entitas dapat mengacu pada peran tersebut. Anda

dapat mengedit deskripsi peran menggunakan IAM. Untuk informasi selengkapnya, silakan lihat Menyunting peran terkait layanan dalam Panduan Pengguna IAM.

Menghapus sebuah peran terkait layanan untuk Monitor Internet

Jika Anda tidak lagi perlu menggunakan sebuah fitur atau layanan yang memerlukan sebuah peran terkait layanan, kami merekomendasikan Anda untuk menghapus peran tersebut. Dengan begitu, Anda tidak memiliki entitas yang tidak digunakan yang tidak dipantau atau dipelihara secara aktif. Akan tetapi, Anda harus membersihkan sumber daya peran terkait layanan sebelum menghapusnya secara manual.

Setelah menghapus sumber daya dari monitor di Internet Monitor dan kemudian menghapus monitor, Anda dapat menghapus peran terkait layanan. AWSServiceRoleForInternetMonitor

> **Note**
>
> Jika layanan Monitor Internet menggunakan peran tersebut ketika Anda mencoba menghapusnya, penghapusan mungkin gagal. Jika hal tersebut terjadi, tunggu beberapa menit lalu coba lagi.

Untuk menghapus peran terkait layanan secara manual menggunakan IAM

Gunakan konsol IAM, the AWS CLI, atau AWS API untuk menghapus peran AWSServiceRoleForInternetMonitorterkait layanan. Untuk informasi selengkapnya, silakan lihat Menghapus peran terkait layanan di Panduan Pengguna IAM.

Pembaruan pada peran terkait layanan di Monitor Internet

Untuk pembaruan AWSServiceRoleForInternetMonitor, kebijakan AWS terkelola untuk peran terkait layanan Internet Monitor, lihat CloudWatch pembaruan kebijakan AWS terkelola. Untuk peringatan otomatis tentang perubahan kebijakan terkelola CloudWatch, berlangganan umpan RSS di halaman Riwayat CloudWatch dokumen.

Kuota di Amazon CloudWatch Internet Monitor

Amazon CloudWatch Internet Monitor memiliki kuota berikut.

Sumber daya	Kuota bawaan
Monitor per Wilayah	50
Sumber daya per monitor	50
Hari-hari untuk menyelesaikan peristiwa kondisi kesehatan di Monitor Internet dipertahankan	400

Menggunakan Monitor CloudWatch Jaringan Amazon

Amazon CloudWatch Network Monitor memberikan visibilitas ke kinerja jaringan yang menghubungkan aplikasi yang AWS dihosting ke tujuan lokal dan memungkinkan Anda mengidentifikasi sumber penurunan kinerja jaringan dalam beberapa menit. Monitor Jaringan sepenuhnya dikelola oleh AWS. Oleh karena itu, Anda tidak perlu menginstal agen tambahan untuk memantau performa jaringan Anda. Anda dapat memvisualisasikan kehilangan paket dan latensi koneksi jaringan hybrid Anda dengan cepat, mengatur peringatan dan ambang batas, dan kemudian mengambil tindakan-tindakan untuk meningkatkan pengalaman jaringan pengguna akhir Anda.

Monitor Jaringan ditujukan untuk operator jaringan dan developer aplikasi yang menginginkan wawasan waktu-nyata tentang performa jaringan.

Fitur utama

- Gunakan Monitor Jaringan untuk membandingkan lingkungan jaringan hybrid Anda yang berubah dengan metrik kehilangan paket dan latensi waktu-nyata yang terjadi terus-menerus.
- Ketika Anda terhubung dengan menggunakanAWS Direct Connect, Network Monitor dengan cepat mendiagnosis degradasi jaringan dengan menulis Indikator AWS Kesehatan Jaringan ke akun Anda CloudWatch . Metrik ini memberikan skor probabilistik untuk menentukan apakah degradasi jaringan terjadi dalam AWS.
- Monitor Jaringan menyediakan pemantauan tanpa friksi dengan pendekatan agen terkelola sepenuhnya, yang berarti Anda tidak perlu menginstal agen baik di VPC maupun on-premise. Anda hanya perlu menentukan subnet VPC dan alamat IP on-premise untuk memulai.

- Network Monitor menerbitkan metrik ke CloudWatch Metrik. Anda dapat membuat dasbor untuk melihat metrik-metrik Anda dan membuat ambang batas dan alarm yang dapat ditindaklanjuti pada metrik-metrik tertentu untuk aplikasi Anda.

Untuk detail selengkapnya, lihat the section called “Cara kerja Monitor Jaringan”.

Terminologi dan komponen Monitor Jaringan

- Monitor — Monitor memperlihatkan sumber daya yang ingin Anda lihat pengukuran kinerja dan ketersediaan jaringannya, dan yang ingin Anda dapatkan peringatan peristiwa kesehatannya. Saat Anda membuat monitor untuk sebuah aplikasi, Anda menambahkan sumber daya yang di-host oleh AWS sebagai sumber jaringan. Monitor Jaringan kemudian membuat daftar semua probe yang mungkin antara sumber daya yang di-host oleh AWS dan alamat IP tujuan Anda.
- Probe — Probe adalah lalu lintas yang dikirim dari sumber daya yang di-host AWS ke alamat IP tujuan on-premise Anda. Metrik Network Monitor ditulis ke CloudWatch akun Anda untuk setiap probe yang dikonfigurasi di monitor.
- Sumber jaringan AWS — Ini adalah sumber AWS asal probe monitor jaringan, yang akan menjadi subnet di salah satu VPC Anda.
- Tujuan — Ini adalah target di jaringan on-premise Anda untuk sumber daya jaringan AWS. Tujuan adalah kombinasi dari alamat IP on-premise Anda, protokol jaringan, port, dan ukuran paket jaringan. Alamat IPv4 dan IPv6 keduanya didukung.

Batasan dan persyaratan Monitor Jaringan

- Monitor Jaringan mendukung maksimal empat alamat IP tujuan, dan hingga 24 probe per monitor.
- Anda dapat memiliki hingga 100 monitor per Wilayah per akun.
- Multi-akun tidak didukung.
- Monitor Jaringan tidak menyediakan failover jaringan otomatis jika terjadi masalah jaringan AWS.
- Ada biaya untuk setiap probe yang Anda buat. Untuk detail harga, silakan lihat the section called “Harga”.

Cara kerja Monitor CloudWatch Jaringan Amazon

Monitor Jaringan membuat pemantauan lebih mudah dengan menyediakan solusi terkelola penuh dan tanpa agen. Saat Anda membuat sebuah monitor di sumber daya yang di-host oleh AWS, AWS

akan membuat dan mengelola semua infrastruktur di latar belakang untuk melakukan pengukuran waktu round-trip dan kehilangan paket. Akibatnya, Anda dapat menskalakan pemantauan Anda dengan cepat tanpa perlu menginstal atau menghapus instalasi agen apa pun dalam infrastruktur AWS Anda.

Monitor Jaringan berfokus pada pemantauan terhadap rute yang diambil oleh aliran dari sumber daya Anda yang di-host oleh AWS alih-alih memantau secara luas semua aliran dari Wilayah AWS Anda. Jika beban kerja Anda tersebar di beberapa Monitor Jaringan Zona Ketersediaan (AZ) dapat memantau rute dari masing-masing subnet privat Anda.

Monitor Jaringan menerbitkan metrik waktu round-trip dan kehilangan paket ke akun Amazon CloudWatch Anda berdasarkan interval agregasi yang ditetapkan saat Anda membuat sebuah monitor. Anda juga dapat mengatur latensi individu dan ambang batas kehilangan paket untuk setiap monitor yang digunakan. CloudWatch Sebagai contoh, Anda dapat membuat sebuah alarm untuk memberi tahu Anda jika rata-rata kehilangan paket Anda lebih tinggi dari ambang batas statis 0,1% untuk beban kerja sensitif kehilangan paket. Anda juga dapat menggunakan deteksi CloudWatch anomali untuk alarm pada kehilangan paket atau metrik latensi di luar rentang yang Anda inginkan.

Pengukuran ketersediaan dan performa

Monitor Jaringan mengirimkan probe aktif berkala dari sumber daya AWS Anda ke tujuan-tujuan on-premise Anda. Saat Anda membuat sebuah monitor, Anda akan menentukan hal-hal berikut:

- Interval agregasi. Waktu, dalam hitungan detik, yang CloudWatch menerima hasil yang diukur. Hal ini akan terjadi setiap 30 atau 60 detik. Periode agregasi yang Anda pilih untuk monitor tersebut berlaku untuk semua probe di monitor itu.
- Protokol probe. Setiap probe yang ditambahkan ke monitor harus menggunakan protokol Internet Control Message Protocol (ICMP) atau Transmission Control Protocol (TCP). Lihat the section called “Protokol komunikasi” untuk detail selengkapnya.
- Ukuran paket. Ukuran, dalam byte, dari setiap paket yang ditransmisikan antara sumber daya Anda yang di-host AWS dan tujuan Anda pada satu probe. Setiap probe di sebuah monitor dapat memiliki ukuran paketnya sendiri.

Untuk metrik,

- Metrik waktu round-trip, diukur dalam milidetik, mengukur dan mencatat ukuran performa dan mencatat waktu yang diperlukan untuk probe yang akan dikirim ke alamat IP tujuan dan untuk respons terkait yang akan diterima.

- Metrik kehilangan paket mengukur persentase total paket yang dikirim dan mencatat jumlah probe yang ditransmisikan yang tidak menerima respons terkait, yang menyiratkan bahwa paket-paket tersebut secara efektif hilang di sepanjang jalur jaringan.

Protokol komunikasi yang didukung

Probe berbasis ICMP membawa permintaan echo ICMP dari sumber daya Anda yang di-host oleh AWS ke alamat tujuan dan mengharapkan balasan gema ICMP kembali dari alamat tujuan. Monitor Jaringan menggunakan informasi tentang permintaan gema ICMP dan membalas pesan untuk menghitung waktu round-trip dan metrik kehilangan paket.

Probe berbasis TCP membawa paket TCP SYN dari sumber daya Anda yang di-host AWS ke alamat dan port tujuan dan mengharapkan paket TCP SYN+ACK atau RST kembali dari alamat dan port tujuan. Monitor Jaringan menggunakan informasi pada TCP SYN dan TCP SYN+ACK atau pesan RST untuk menghitung waktu round-trip dan metrik kehilangan paket. Selain itu, Monitor Jaringan secara berkala mengganti port TCP sumber untuk meningkatkan jangkauan jaringan, yang kemudian dapat meningkatkan kemungkinan untuk mendeteksi kehilangan paket.

Indikator Kondisi Kesehatan Jaringan AWS

Monitor Jaringan menerbitkan metrik Indikator Kondisi Kesehatan Jaringan (NHI), yang memberikan informasi tentang performa jaringan dan ketersediaan untuk tujuan yang terhubung melalui AWS Direct Connect. Metrik adalah sebuah ukuran statistik kondisi kesehatan jalur jaringan yang dikendalikan AWS dari sumber daya yang di-host oleh AWS, yang merupakan tempat monitor digunakan, ke lokasi Direct Connect.

Monitor Jaringan menggunakan deteksi anomali untuk menghitung penurunan ketersediaan atau penurunan performa di sepanjang jalur jaringan Anda.

> **Note**
>
> Setiap kali Anda membuat sebuah monitor baru, menambahkan probe, atau mengaktifkan kembali probe, NHI untuk monitor itu akan ditunda beberapa jam untuk memungkinkan pengumpulan data oleh AWS untuk melakukan deteksi anomali.

Untuk menyediakan metrik kondisi kesehatan NHI, Monitor Jaringan akan menerapkan korelasi statistik di seluruh set data sampel AWS, serta metrik latensi kehilangan paket dan round-trip untuk

lalu lintas yang mensimulasikan jalur jaringan Anda. Metrik tersebut dapat berupa salah satu dari dua variabel: 1 atau 0. Nilai 1 menunjukkan bahwa Monitor Jaringan mengamati degradasi jaringan dalam jalur jaringan yang dikendalikan oleh AWS. Nilai 0 menunjukkan bahwa Monitor Jaringan tidak mengamati degradasi jaringan di sepanjang jalur. Hal ini akan memungkinkan Anda untuk kemudian memecahkan masalah jaringan dengan lebih cepat. Anda dapat mengatur peringatan pada metrik NHI agar Anda diberitahu tentang masalah-masalah yang sedang terjadi di sepanjang jalur jaringan Anda.

Dukungan untuk alamat IPv4 dan IPv6

Monitor Jaringan menyediakan ketersediaan dan metrik performa melalui jaringan IPv4 atau IPv6 dan dapat memantau alamat IPv4 atau IPv6 dari vPC tumpukan ganda. Monitor Jaringan tidak mengizinkan tujuan IPv4 dan IPv6 untuk dikonfigurasi dalam monitor yang sama, tetapi Anda dapat membuat memisahkannya untuk tujuan-tujuan khusus IPv4 dan IPv6.

Ketersediaan wilayah

Monitor Jaringan saat ini tersedia di Wilayah AWS berikut:

Wilayah	
Asia Pasifik (Hong Kong)	ap-timur-1
Asia Pasifik (Mumbai)	ap-south-1
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1

Wilayah	
Asia Pacific (Sydney)	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
Kanada Barat (Calgary)	ca-west-1
Eropa (Frankfurt)	eu-central-1
Europe (Ireland)	eu-west-1
Europe (London)	eu-west-2
Europe (Paris)	eu-west-3
Europe (Stockholm)	eu-utara-1
Timur Tengah (Bahrain)	me-selatan-1
Amerika Selatan (São Paulo)	sa-east-1

Wilayah	
US East (Northern Virginia)	us-east-1
US East (Ohio)	us-east-2
US West (Northern California)	us-west-1
US West (Oregon)	us-west-2

Membuat Monitor Jaringan

Langkah-langkah berikut akan menjelaskan kepada Anda cara membuat monitor dan kemudian menambahkan probe yang diperlukan. Untuk probe, Anda akan memilih subnet sumber, dan hingga empat alamat IP tujuan untuk maksimum 24 probe untuk masing-masing monitor. Anda dapat membuat sebuah monitor dengan menggunakan konsol Amazon CloudWatch atau dengan menggunakan baris perintah atau API.

Topik

- Membuat sebuah Monitor Jaringan menggunakan konsol
- Membuat sebuah Monitor Jaringan menggunakan baris perintah atau API

Membuat sebuah Monitor Jaringan menggunakan konsol

Langkah-langkah berikut akan menjelaskan kepada Anda cara membuat sebuah monitor dengan menggunakan konsol Amazon CloudWatch. Anda akan memilih subnet sumber Anda, dan kemudian menambahkan hingga empat tujuan untuk membuat hingga 24 probe untuk masing-masing monitor. Anda dapat membuat sebuah monitor dengan menggunakan konsol Amazon CloudWatch atau dengan menggunakan baris perintah atau SDK.

> **Important**
>
> Langkah-langkah ini dirancang untuk Anda selesaikan sekaligus. Anda tidak lagi dapat menyimpan pekerjaan di tengah proses untuk kemudian melanjutkan pekerjaan apa pun untuk dilanjutkan nanti.

Tentukan detail monitor

Langkah pertama untuk membuat sebuah monitor adalah menentukan detail dasar. Hal ini termasuk memberi nama monitor dan menentukan periode agregasinya. Anda dapat menambahkan Tanda opsional ke monitor.

Cara menentukan detail monitor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/, dan kemudian di bawah Network Monitoring, pilih Network Monitor.
2. Pilih Buat monitor.
3. Untuk Nama monitor, masukkan nama yang ingin Anda gunakan untuk monitor ini.
4. Untuk periode Agregasi, pilih seberapa sering Anda ingin mengirim metrik. CloudWatch Periode agregasi yang tersedia adalah berikut ini:
   - 30 detik
   - 60 detik

   > **Note**
   >
   > Periode agregasi yang lebih singkat akan memberikan deteksi masalah jaringan yang lebih cepat; Namun demikian, periode agregasi yang Anda pilih dapat memengaruhi struktur penagihan Anda. Untuk informasi selengkapnya tentang harga, lihat halaman CloudWatch harga Amazon.
5. (Opsional) Di bagian Tag, tambahkan pasangan Kunci dan Nilai untuk membantu Anda dalam mengidentifikasi sumber daya ini lebih lanjut, memungkinkan Anda untuk mencari atau memfilter informasi tertentu.

   1. Pilih Tambahkan tanda baru.

2. Masukkan nama Kunci dan Nilai terkait.
3. Pilih Tambahkan tanda baru untuk menambahkan tanda baru tersebut.

   Anda dapat menambahkan beberapa tanda dengan memilih Tambahkan tanda baru, atau Anda dapat menghapus tanda apa pun dengan memilih Hapus.
4. Jika Anda ingin mengaitkan tanda Anda dengan monitor, biarkan Tambahkan tanda ke probe yang dibuat oleh monitor tetap dicentang. Langkah ini akan menambahkan tanda ke probe monitor, yang dapat membantu jika Anda menggunakan autentikasi atau pengukuran berbasis tag.

6. Pilih Berikutnya untuk the section called “Pilih sumber dan tujuan”.

Pilih sumber dan tujuan

Monitor jaringan menggunakan sebuah sumber AWS untuk VPC dan subnet terkait di Wilayah tempat jaringan Anda beroperasi. Tujuan monitor adalah kombinasi dari alamat IP on-premise Anda, protokol jaringan, port, dan ukuran paket jaringan.

Kombinasi sumber dan tujuan disebut sebagai probe. Anda dapat memiliki hingga empat probe untuk masing-masing subnet, dan hingga total 24 probe untuk masing-masing monitor.

> **Important**
>
> Langkah-langkah ini dirancang untuk Anda selesaikan sekaligus. Anda tidak lagi dapat menyimpan pekerjaan di tengah proses untuk kemudian melanjutkan pekerjaan apa pun untuk dilanjutkan nanti.

Cara memilih sumber dan tujuan

1. Pada sumber jaringan AWS, pilih satu atau beberapa subnet untuk disertakan dalam monitor. Anda dapat memilih satu VPC, yang kemudian akan memilih semua subnet dalam VPC itu, atau Anda juga dapat memilih subnet tertentu. VPC dan subnet yang Anda pilih akan menjadi sumber dari monitor jaringan tersebut.
2. Untuk Tujuan 1, masukkan alamat IP Tujuan jaringan on-premise. Alamat IPv4 dan IPv6 keduanya didukung.
3. Pilih Pengaturan lanjutan.
4. Untuk tujuan yang dikelola pelanggan ini, pilih Protokol jaringan. Itu bisa berupa:

- ICMP
- TCP

5. Jika Protokol adalah TCP, masukkan informasi berikut. Jika tidak, lewati dan langsung ke langkah berikutnya:

   1. Masukkan Port yang digunakan oleh jaringan Anda untuk terhubung. Port harus berupa angka dari 1 hingga 65535.
   2. Masukkan Ukuran paket. Ini adalah ukuran dari, dalam byte, masing-masing paket yang dikirim pada probe antara sumber dan tujuan. Ukuran paket harus berupa angka dari 56 hingga 8500.

6. Pilih Tambahkan tujuan untuk menambahkan tujuan on-premise lain ke monitor ini. Ulangi langkah-langkah ini untuk setiap tujuan yang ingin Anda tambahkan.
7. Pilih Berikutnya setelah selesai untuk mengonfirmasi probe.

Konfirmasi probe

Mengonfirmasi probe akan memungkinkan Anda untuk meninjau kombinasi probe jaringan untuk monitor. Halaman ini akan menampilkan semua kemungkinan kombinasi sumber dan tujuan yang telah Anda pilih. Sebagai contoh, jika Anda memiliki enam subnet sumber dan empat IP tujuan, maka Anda akan memiliki total 24 kemungkinan kombinasi probe.

> **Important**
>
> - Langkah-langkah ini dirancang untuk Anda selesaikan sekaligus. Anda tidak lagi dapat menyimpan pekerjaan di tengah proses untuk kemudian melanjutkan pekerjaan apa pun untuk dilanjutkan nanti.
> - Halaman Konfirmasi probe tidak menunjukkan apakah sebuah probe valid atau tidak. Oleh karena itu, kami merekomendasikan Anda untuk meninjau halaman ini secara menyeluruh dan menghapus setiap probe yang tidak valid. Jika Anda tidak menghapus probe yang tidak valid, maka Anda mungkin akan dikenai biaya untuk itu.

Cara mengonfirmasi probe monitor

1. Prasyarat: the section called “Pilih sumber dan tujuan”.

2. Pada halaman Konfirmasi probe, tinjau daftar kombinasi sumber dan tujuan.
3. Pilih satu atau beberapa probe yang ingin Anda hapus dari monitor, dan kemudian pilih Hapus.

> **Note**
>
> Anda tidak akan diminta untuk mengonfirmasi penghapusan. Setelah probe dihapus, Anda harus mengaturnya lagi, Anda dapat menambahkan probe kembali ke monitor dari bagian Monitor jaringan pada halaman Monitor Jaringan. Untuk informasi selengkapnya, lihat the section called “Menambahkan probe ke sebuah monitor”.

4. Pilih Berikutnya untuk meninjau detail monitor sebelum Anda membuatnya.

Tinjau dan buat

Langkah terakhir dalam membuat sebuah monitor dan probe adalah meninjau detail monitor dan probe tersebut. Anda dapat mengubah informasi apa pun pada langkah ini. Setelah Anda selesai meninjau dan telah membuat monitor, dan metrik mulai melacak, Anda akan mulai dikenakan biaya untuk setiap probe.

> **Important**
>
> - Langkah ini dirancang untuk Anda selesaikan sekaligus saat membuat sebuah monitor dan probe. Anda tidak lagi dapat menyimpan pekerjaan di tengah proses untuk kemudian melanjutkan pekerjaan apa pun untuk dilanjutkan nanti.
> - Jika Anda memilih untuk mengedit bagian apa pun, maka Anda harus menyelesaikan langkah pembuatan monitor dari titik di mana Anda melakukan suntingan. Namun demikian, Anda tidak perlu membuat ulang langkah-langkah selanjutnya. Halaman-halaman ini akan menyimpan informasi yang sudah Anda isi sebelumnya.

Cara meninjau dan membuat monitor

1. Pada halaman Tinjau dan buat probe, pilih Edit untuk bagian mana pun yang ingin Anda ubah.
2. Buat perubahan apa pun di bagian tersebut.
3. Pilih Berikutnya.
4. Lakukan salah satu langkah berikut ini:

- Buat perubahan apa pun yang ingin Anda lakukan pada halaman monitor tambahan, dan pilih Berikutnya hingga Anda kembali ke halaman Tinjau dan buat.
- Jika tidak ada halaman lain yang ingin Anda ubah, silakan pilih Berikutnya hingga Anda kembali ke halaman Tinjau dan buat.

5. Pilih Buat monitor.

   Halaman Monitor Jaringan menampilkan status pembuatan monitor saat ini di bagian Monitor jaringan. Selama pembuatan monitor, Status berada dalam keadaan Tertunda. Saat Status berubah menjadi Aktif, Anda dapat mengakses dasbor monitor untuk melihat CloudWatch metrik.

   Untuk informasi tentang cara menggunakan dasbor monitor, silakan lihat the section called “Dasbor Monitor Jaringan”.

> **Note**
>
> Dibutuhkan beberapa menit bagi monitor jaringan yang baru ditambahkan untuk mulai mengumpulkan metrik-metrik jaringan.

Membuat sebuah Monitor Jaringan menggunakan baris perintah atau API

Menggunakan baris perintah atau API untuk melihat dan membuat sebuah monitor jaringan.

Cara membuat monitor jaringan menggunakan baris perintah atau API

1. Buat sebuah monitor jaringan menggunakan create-monitor.
2. Buat sebuah probe monitor jaringan menggunakan create-probe.

Menggunakan monitor dan probe Monitor Jaringan

Anda dapat melakukan salah satu tugas berikut dengan monitor dan probe Anda, baik menggunakan konsol Amazon CloudWatch, atau dengan menggunakan baris perintah atau API.

Topik:

- Mengaktifkan atau Menonaktifkan sebuah probe
- Menambahkan probe ke sebuah monitor
- Mengedit probe
- Menghapus sebuah probe
- Memberikan tanda atau menghapus tanda pada sumber daya dengan menggunakan baris perintah atau API

Mengedit sebuah monitor

Anda dapat mengedit informasi apa pun untuk Monitor Jaringan, termasuk mengganti namanya, mengatur periode agregasi baru, atau menambahkan tanda ataupun menghapus tanda. Mengubah informasi sebuah monitor tidak akan mengubah probe yang dikaitkan kepadanya. Anda dapat mengedit sebuah monitor dengan menggunakan konsol Amazon CloudWatch atau dengan menggunakan baris perintah atau API.

Mengedit sebuah monitor menggunakan konsol

Gunakan CloudWatch konsol untuk mengedit monitor.

Cara mengedit sebuah monitor menggunakan konsol

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/, dan kemudian di bawah Network Monitoring, pilih Network Monitor.
2. Di bagian Monitor jaringan, pilih monitor yang ingin Anda edit.
3. Pada halaman dasbor monitor, pilih Edit.
4. Untuk Nama monitor, masukkan nama baru untuk monitor tersebut.
5. Untuk periode Agregasi, pilih seberapa sering Anda ingin mengirim metrik. CloudWatch Periode yang valid adalah:

    - 30 detik
    - 60 detik

> **Note**
>
> Periode agregasi yang lebih singkat akan memberikan deteksi masalah jaringan yang lebih cepat; Namun demikian, periode agregasi yang Anda pilih dapat memengaruhi

struktur penagihan Anda. Untuk informasi selengkapnya tentang harga, lihat halaman CloudWatch harga Amazon.

6. (Opsional) Di bagian Tag, tambahkan pasangan Kunci dan Nilai untuk membantu Anda dalam mengidentifikasi sumber daya ini lebih lanjut, memungkinkan Anda untuk mencari atau memfilter informasi tertentu. Anda juga dapat mengubah Nilai dari Kunci apa pun pada langkah ini.

   1. Pilih Tambahkan tanda baru.
   2. Masukkan nama Kunci dan Nilai terkait.
   3. Pilih Tambahkan tanda baru untuk menambahkan tanda baru tersebut.

      Anda dapat menambahkan beberapa tanda dengan memilih Tambahkan tanda baru, atau Anda dapat menghapus tanda apa pun dengan memilih Hapus.
   4. Jika Anda ingin mengaitkan tanda Anda dengan monitor, biarkan Tambahkan tanda ke probe yang dibuat oleh monitor tetap dicentang. Langkah ini akan menambahkan tanda ke probe monitor, yang dapat membantu jika Anda menggunakan autentikasi atau pengukuran berbasis tag.
7. Pilih Simpan perubahan.

Mengedit sebuah monitor menggunakan CLI atau API

Menggunakan baris perintah atau API untuk melihat dan mengedit sebuah monitor.

Cara mengedit monitor menggunakan baris perintah atau API

1. Gunakan list-monitor untuk mendapatkan daftar monitor Anda jika Anda tidak tahu nama monitor tersebut. Catat nama monitor yang ingin Anda edit.
2. Gunakan edit-monitor dengan menggunakan nama monitor dari langkah sebelumnya.

Menghapus sebuah monitor

Sebelum Anda dapat menghapus sebuah monitor, Anda harus menonaktifkan atau menghapus semua probe yang dikaitkan dengan monitor itu, terlepas dari Status yang dimiliki monitor itu. Setelah monitor dinonaktifkan atau dihapus, Anda tidak akan lagi dikenakan biaya untuk probe yang dikaitkan dengan monitor tersebut. Anda tidak dapat memulihkan monitor yang sudah dihapus. Anda dapat menghapus sebuah monitor dengan menggunakan konsol Amazon CloudWatch atau dengan menggunakan baris perintah atau API.

Meskipun probe mungkin dihapus atau dinonaktifkan, CloudWatch masih mempertahankan metrik selama 15 hari.

Menghapus sebuah monitor menggunakan konsol

Gunakan CloudWatch konsol untuk menghapus monitor.

Cara menghapus sebuah monitor menggunakan konsol

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/, dan kemudian di bawah Network Monitoring, pilih Network Monitor.
2. Di bagian Monitor jaringan, pilih monitor yang ingin Anda hapus.
3. Pilih Tindakan, lalu pilih Hapus.
4. Jika Anda memiliki probe yang masih aktif, Anda akan diminta untuk menonaktifkannya. Pilih Nonaktifkan probe.

> **Note**
>
> Anda tidak dapat membatalkan atau membatalkan tindakan ini setelah Anda memilih Nonaktifkan probe. Akan tetapi, probe yang dinonaktifkan tidak akan dihapus dari monitor. Anda dapat mengaktifkannya kembali nanti setelahnya. Lihat the section called “Mengaktifkan atau Menonaktifkan sebuah probe”.

5. Dalam bidang konfirmasi, masukkan **`confirm`**, dan kemudian pilih Hapus.

Menghapus sebuah monitor menggunakan baris perintah atau API

Menghapus sebuah monitor menggunakan baris perintah atau API.

Cara menghapus sebuah monitor jaringan menggunakan baris perintah atau API

1. Anda memerlukan nama monitor yang ingin Anda hapus. Jika Anda tidak tahu namanya, gunakan list-monitor untuk mendapatkan daftar monitor Anda. Perhatikan nama monitor yang ingin Anda hapus.
2. Verifikasi apakah monitor tersebut berisi probe apa pun. Gunakan get-monitor dengan nama monitor dari langkah yang sebelumnya. Hal ini akan mengembalikan daftar dari probe apa pun yang dikaitkan dengan monitor itu.

3. Jika monitor berisi probe, pertama-tama Anda harus mengatur probe tersebut agar menjadi tidak aktif atau menghapusnya.

   - Untuk menyetel probe menjadi tidak aktif, gunakan update-probe, dan setel statusnya menjadi `INACTIVE`.
   - Cara menghapus probe, gunakan delete-probe.

4. Setelah probe disetel menjadi `INACTIVE` atau dihapus, gunakan delete-monitor untuk menghapus monitor. Probe yang tidak aktif tidak dihapus.

Mengaktifkan atau Menonaktifkan sebuah probe

Anda dapat mengaktifkan atau menonaktifkan sebuah probe monitor sesuai kebutuhan. Anda mungkin ingin menonaktifkan sebuah probe jika saat ini Anda tidak menggunakannya, tetapi Anda mungkin ingin menggunakannya lagi di masa mendatang. Dengan menonaktifkan sebuah probe, Anda tidak perlu menghabiskan waktu untuk mengaturnya lagi. Anda tidak akan dikenai biaya untuk probe yang sudah dinonaktifkan.

Anda dapat mengubah status dari sebuah monitor dengan menggunakan konsol Amazon CloudWatch atau dengan menggunakan baris perintah atau API.

Mengatur probe menjadi aktif atau tidak aktif dengan menggunakan konsol

Gunakan CloudWatch konsol untuk mengatur probe menjadi aktif atau tidak aktif.

Cara mengatur sebuah probe menjadi aktif atau tidak aktif dengan menggunakan konsol

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/, dan kemudian di bawah Network Monitoring, pilih Network Monitor.
2. Pilih tab Detail monitor.
3. Di bagian Probe, pilih probe yang ingin diaktifkan atau dinonaktifkan.
4. Pilih Tindakan, kemudian pilih Aktifkan atau Nonaktifkan.

> **Note**
>
> Jika Anda mengaktifkan kembali sebuah probe yang sudah dinonaktifkan, Anda akan mulai mengeluarkan biaya penagihan untuk probe itu.

Mengatur sebuah probe menjadi aktif atau tidak aktif dengan menggunakan baris perintah atau API

Mengatur sebuah probe menjadi aktif atau tidak aktif atau melakukan penonaktifan dengan menggunakan baris perintah atau API. Anda hanya dapat menggunakan perintah ini untuk satu probe saja.

Cara mengatur sebuah probe menjadi aktif atau tidak aktif dengan menggunakan baris perintah atau API

1. Gunakan list-monitor untuk mendapatkan daftar monitor Anda jika Anda tidak tahu nama monitor tersebut. Perhatikan nama monitor yang ingin Anda ubah statusnya.
2. Gunakan get-monitor dengan nama monitor dari langkah yang sebelumnya. Hal ini akan mengembalikan daftar dari probe apa pun yang dikaitkan dengan monitor itu. Perhatikan ID probe dari probe yang ingin Anda ubah statusnya.
3. Gunakan update-probe dan atur probe yang statusnya ingin Anda ubah menjadi `ACTIVE` atau `INACTIVE`.

Menambahkan probe ke sebuah monitor

Anda dapat menambahkan probe ke sebuah monitor yang sudah ada. Perhatikan bahwa jika Anda menambahkan probe apa pun ke sebuah monitor, maka struktur penagihan Anda akan diperbarui untuk menunjukkan bahwa probe baru telah ditambahkan.

Menambahkan probe ke sebuah monitor menggunakan konsol

Cara menambahkan probe ke sebuah monitor menggunakan konsol

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/, dan kemudian di bawah Network Monitoring, pilih Network Monitor.
2. Di bagian Monitor jaringan, lakukan salah satu hal berikut ini:
   - Pilih tautan Nama monitor yang ingin Anda tambahkan probe padanya. Pilih tab Detail monitor, dan kemudian di bagian Probe, pilih Tambahkan probe.
   - Pilih kotak centang monitor, pilih Tindakan, kemudian pilih Tambahkan probe.
3. Pada halaman Tambahkan probe, lakukan hal berikut ini:
   1. Pada sumber jaringan AWS, pilih subnet yang akan ditambahkan ke monitor.

> **Note**
>
> Anda hanya dapat menambahkan satu probe pada satu waktu dan hingga empat probe untuk masing-masing monitor.

2. Masukkan alamat IP tujuan jaringan on-premise. Alamat IPv4 dan IPv6 keduanya didukung.
3. Pilih Pengaturan lanjutan.
4. Pilih Protokol jaringan untuk tujuan. Ini bisa berupa ICMP atau TCP.
5. Jika Protokol adalah TCP, masukkan informasi berikut. Jika tidak, lewati dan langsung ke langkah berikutnya:
   - Masukkan Port yang digunakan oleh jaringan Anda untuk terhubung. Port harus berupa angka dari 1 hingga 65535.
   - Masukkan Ukuran paket. Ini adalah ukuran dari masing-masing paket yang dikirim pada probe antara sumber dan tujuan yang dinyatakan dengan satuan byte. Ukuran paket harus berupa angka dari 56 hingga 8500.

4. (Opsional) Di bagian Tag, tambahkan pasangan Kunci dan Nilai untuk membantu Anda dalam mengidentifikasi sumber daya ini lebih lanjut, memungkinkan Anda untuk mencari atau memfilter informasi tertentu.

   1. Pilih Tambahkan tanda baru.
   2. Masukkan nama Kunci dan Nilai terkait.
   3. Pilih Tambahkan tanda baru untuk menambahkan tanda baru itu.

      Anda dapat menambahkan beberapa tanda dengan memilih Tambahkan tanda baru, atau Anda dapat menghapus tanda apa pun dengan memilih Hapus.

5. Pilih Tambahkan probe.

   Saat probe sedang diaktifkan, Status menunjukkan Ditunda. Diperlukan beberapa menit sampai probe menjadi Aktif.

Menambahkan probe ke monitor menggunakan baris perintah atau API

Menambahkan probe ke sebuah monitor menggunakan baris perintah atau API. Anda hanya dapat menggunakan perintah ini untuk menambahkan satu probe tunggal pada satu waktu.

Cara menambahkan sebuah probe ke monitor menggunakan baris perintah atau API

1. Gunakan list-monitor untuk mendapatkan daftar monitor Anda jika Anda tidak tahu nama monitor tersebut. Perhatikan nama monitor tempat Anda ingin menambahkan sebuah probe.
2. Gunakan create-probe untuk menambahkan sebuah probe ke monitor.

Mengedit probe

Anda dapat mengubah informasi apa pun untuk probe saat ini, terlepas dari apakah probe itu diaktifkan atau dinonaktifkan. Anda dapat mengedit sebuah probe dengan menggunakan konsol Amazon CloudWatch atau dengan menggunakan baris perintah atau API.

Mengedit sebuah probe dengan menggunakan konsol

Cara mengedit sebuah probe dengan menggunakan konsol

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/, dan kemudian di bawah Network Monitoring, pilih Network Monitor.

   Pilih tautan Nama untuk membuka dasbor monitor.
2. Pilih tab Detail monitor.
3. Di bagian Probe, pilih tautan untuk probe yang ingin Anda edit.
4. Pada halaman dasbor probe, pilih Edit, atau pilih Tindakan kemudian pilih Edit.
5. Pada halaman Edit probe, masukkan alamat IP probe tujuan baru. Alamat IPv4 dan IPv6 keduanya didukung.
6. Pilih Pengaturan lanjutan.
7. Pilih Protokol jaringan. Ini bisa berupa ICMP atau TCP.
8. Jika Protokol adalah TCP, masukkan informasi berikut. Jika tidak, lewati dan langsung ke langkah berikutnya:

   - Masukkan Port yang digunakan oleh jaringan Anda untuk terhubung. Port harus berupa angka dari 1 hingga 65535.
   - Masukkan Ukuran paket. Ini adalah ukuran dari masing-masing paket yang dikirim pada probe antara sumber dan tujuan yang dinyatakan dengan satuan byte. Ukuran paket harus berupa angka dari 56 hingga 8500.

9. (Opsional) Di bagian Tag, tambahkan pasangan Kunci dan Nilai untuk membantu Anda dalam mengidentifikasi sumber daya ini lebih lanjut, memungkinkan Anda untuk mencari atau memfilter informasi tertentu.

    1. Pilih Tambahkan tanda baru.
    2. Masukkan nama Kunci dan Nilai terkait.
    3. Pilih Tambahkan tanda baru untuk menambahkan tanda baru itu.

        Anda dapat menambahkan beberapa tanda dengan memilih Tambahkan tanda baru, atau Anda dapat menghapus tanda apa pun dengan memilih Hapus.

10. Pilih Simpan perubahan.

Mengedit sebuah probe menggunakan baris perintah atau API

Menggunakan baris perintah untuk mengedit sebuah probe monitor. Anda hanya dapat menggunakan perintah ini untuk satu probe saja.

Cara mengedit sebuah probe menggunakan baris perintah atau API

1. Gunakan list-monitor untuk mendapatkan daftar monitor Anda jika Anda tidak tahu nama monitor tersebut. Perhatikan nama monitor yang ingin Anda ubah statusnya.
2. Gunakan get-monitor dengan nama monitor dari langkah yang sebelumnya. Hal ini akan mengembalikan daftar dari probe apa pun yang dikaitkan dengan monitor itu. Perhatikan ID probe dari setiap probe yang ingin Anda edit.
3. Gunakan update-probe untuk mengubah informasi probe.

Menghapus sebuah probe

Anda dapat menghapus sebuah probe, alih-alih menonaktifkannya, jika Anda tahu bahwa Anda tidak akan membutuhkannya lagi di masa mendatang. Anda tidak akan dapat memulihkan probe yang dihapus dan sebagai gantinya harus membuatnya kembali. Penagihan akan dihentikan untuk probe itu ketika probe dihapus. Anda dapat menghapus sebuah probe dengan menggunakan konsol Amazon CloudWatch, atau baris perintah atau API.

Menghapus sebuah probe menggunakan konsol

Cara menghapus sebuah probe menggunakan konsol

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/, dan kemudian di bawah Network Monitoring, pilih Network Monitor.
2. Di bagian Monitor jaringan, pilih tautan Nama untuk membuka dasbor monitor.
3. Pilih tab Detail monitor.
4. Pilih kotak centang monitor, pilih Tindakan, kemudian pilih Hapus.
5. Pada kotak dialog Hapus probe, pilih Hapus untuk mengonfirmasi bahwa Anda ingin menghapus probe.
6. Pilih Hapus untuk mengonfirmasi bahwa Anda ingin menghapus probe tersebut.

   Status probe di bagian Probe menunjukkan Menghapus. Setelah dihapus, probe akan dihapus dari bagian Probe.

Menghapus sebuah probe dengan menggunakan baris perintah atau API

Menghapus sebuah probe dengan menggunakan baris perintah atau API. Anda hanya dapat menggunakan perintah ini untuk satu probe saja.

Cara mengatur sebuah probe menjadi aktif atau tidak aktif dengan menggunakan baris perintah atau API

1. Gunakan list-monitor untuk mendapatkan daftar monitor Anda jika Anda tidak tahu nama monitor tersebut. Perhatikan nama monitor yang memiliki probe yang ingin Anda hapus
2. Gunakan get-monitor dengan nama monitor dari langkah yang sebelumnya. Hal ini akan mengembalikan daftar dari probe apa pun yang dikaitkan dengan monitor itu. Perhatikan ID probe dari setiap probe yang ingin Anda hapus.
3. Gunakan delete-probe.

Memberikan tanda atau menghapus tanda pada sumber daya dengan menggunakan baris perintah atau API

Anda dapat menggunakan baris perintah atau CLI untuk menambahkan atau memperbarui tanda sumber daya.

Cara memperbarui tanda monitor jaringan dengan menggunakan baris perintah atau API

- Untuk membuat daftar tag sumber daya, gunakan list-tags-for-resources.
- Untuk menandai sumber daya, gunakan tag-resource.
- Untuk menghapus tanda sumber daya, gunakan untag-resource.

Dasbor Monitor Jaringan

Anda dapat menggunakan dasbor Amazon CloudWatch Network Monitor untuk melihat kesehatan AWS jaringan, dan memeriksa waktu pulang-pergi dan kehilangan paket. Anda dapat melihat metrik-metrik ini untuk monitor dan untuk masing-masing probe.

Dasbor Monitor Jaringan

- Dasbor monitor
- Dasbor probe

Alarm probe

Anda dapat membuat CloudWatch alarm Amazon berdasarkan metrik Monitor CloudWatch Jaringan Amazon, seperti yang Anda bisa untuk metrik Amazon CloudWatch lainnya. Alarm apa pun yang Anda buat akan muncul di kolom Status probe di bagian Detail Monitor pada dasbor Monitor Jaringan saat alarm dipicu. Status akan OK atau In Alarm. Jika tidak ada status yang ditampilkan untuk probe, maka tidak ada alarm yang dibuat untuk probe itu.

Misalnya, Anda dapat membuat alarm berdasarkan metrik `PacketLoss` di Monitor Jaringan, dan mengonfigurasinya untuk mengirimkan notifikasi ketika metrik turun lebih tinggi dari nilai yang Anda pilih. Anda mengonfigurasi alarm untuk metrik Monitor Jaringan mengikuti pedoman yang sama seperti metrik lainnya CloudWatch .

Metrik berikut tersedia di bawah `AWS/NetworkMonitor` saat membuat CloudWatch alarm untuk Monitor Jaringan.

- HealthIndicator
- PacketLoss
- Waktu round-trip (RTT)

Untuk langkah-langkah membuat alarm Monitor Jaringan, silakan lihat the section called “Membuat sebuah alarm berdasarkan pada ambang batas statis”.

Mengatur kerangka waktu metrik

Metrik dan peristiwa di kedua dasbor menggunakan waktu default dua jam, dihitung dari waktu saat ini. Anda dapat mengubah default tersebut dan menggunakan salah satu dari preset berikut:

- 1 jam — satu jam
- 2 jam — dua jam
- 1 hari — satu hari
- 1 minggu — satu minggu

Anda juga dapat mengatur kerangka waktu kustom. Pilih Kustom, pilih waktu Absolut atau Relatif, lalu atur kerangka waktu sesuai dengan waktu yang Anda pilih sendiri. Waktu relatif hanya mendukung 15 hari kembali dari tanggal hari ini, per CloudWatch default.

Selain itu, Anda juga dapat memilih waktu yang ditampilkan dalam grafik berdasarkan zona waktu UTC atau zona waktu Lokal.

Dasbor monitor

Anda dapat menggunakan dasbor Amazon CloudWatch Network Monitor untuk melihat kesehatan AWS jaringan, dan memeriksa waktu pulang-pergi dan kehilangan paket. Monitor Jaringan memiliki dasbor untuk monitor dan probe.

Cara mengakses dasbor monitor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/, dan kemudian di bawah Network Monitoring, pilih Network Monitor.
2. Di bagian Monitor jaringan, pilih tautan Nama untuk membuka dasbor monitor.

Ikhtisar

Halaman gambaran umum menampilkan informasi berikut untuk monitor Anda:

- Kondisi kesehatan Jaringan AWS — Kondisi kesehatan Jaringan AWS hanya menampilkan kondisi kesehatan jaringan AWS secara keseluruhan. Statusnya bisa Sehat atau Terdegradasi. Status

Sehat menunjukkan Monitor Jaringan tidak mengamati adanya masalah apa pun dengan jaringan AWS. Status Terdegradasi menunjukkan bahwa Monitor Jaringan mengamati adanya masalah dengan jaringan AWS. Bilah status di bagian ini menunjukkan status jaringan selama waktu default satu jam. Arahkan kursor ke titik mana pun pada bilah status untuk melihat detail tambahan.

- Ringkasan lalu lintas probe — Menampilkan status lalu lintas saat ini antara subnet AWS sumber di monitor dan alamat IP tujuan. Ringkasan lalu lintas probe menampilkan hal-hal berikut:
  - Probe dalam alarm — Nomor ini menunjukkan berapa banyak probe Anda yang berada dalam status terdegradasi. Alarm akan terpicu ketika ada sebuah metrik yang telah Anda atur ketika sebuah alarm terpicu. Untuk informasi tentang alarm metrik Monitor Jaringan, lihatthe section called “Alarm probe”.
  - Kehilangan paket — Jumlah paket yang hilang dari subnet sumber ke alamat IP tujuan. Hal ini direpresentasikan sebagai persentase dari total paket yang dikirim.
  - Waktu round-trip — Waktu yang dibutuhkan, dalam milidetik, untuk paket dari subnet sumber untuk mencapai alamat IP tujuan dan kemudian kembali lagi.

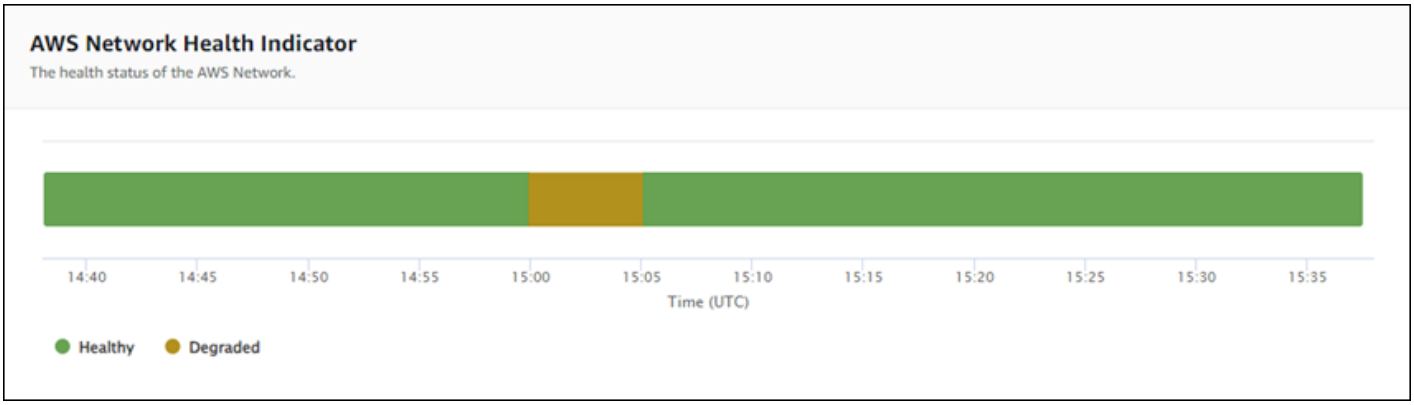
Data diwakili oleh grafik interaktif, yang memungkinkan Anda untuk melihat detailnya.

Secara default, data tersebut akan ditampilkan untuk jangka waktu dua jam, dihitung dari tanggal dan waktu saat ini. Namun demikian, Anda dapat mengubah rentang waktunya agar sesuai dengan kebutuhan Anda. Untuk informasi selengkapnya, lihat the section called “Mengatur kerangka waktu metrik”.

Melacak metrik

Dasbor Monitor Jaringan akan menampilkan representasi grafis dari monitor dan probe Anda. Berikut adalah grafik yang tersedia:

- Indikator Kondisi Kesehatan Jaringan AWS — Ini mewakili kondisi kesehatan jaringan AWS selama periode tertentu. Statusnya bisa berupa status Sehat atau Terdegradasi. Dalam contoh berikut, Anda akan melihat bahwa dari pukul 15:00 UTC hingga 15:05 UTC, jaringan AWS berada dalam status terdegradasi. Setelah 15:05 jaringan kembali ke dalam kondisi sehat. Anda dapat mengarahkan kursor ke bagian mana pun dari grafik untuk melihat detail-detail tambahannya.



> **Note**
>
> Indikator Kesehatan Jaringan tidak menunjukkan kesehatan probe tetapi hanya AWS jaringan.

- Kehilangan paket — Grafik ini akan menampilkan garis unik yang menunjukkan persentase kehilangan paket untuk setiap probe yang ada di monitor. Legenda yang ada di bagian bawah halaman menampilkan setiap probe dalam monitor dan diberi kode warna agar unik. Dengan mengarahkan kursor di atas probe dalam bagan ini akan menampilkan subnet sumber, IP tujuan, dan persentase kehilangan paket. Dalam contoh berikut, alarm kehilangan paket diatur untuk probe dari subnet ke alamat IP 127.0.0.1. Alarm tersebut dipicu ketika ambang batas kehilangan paket terlampaui untuk probe. Dengan mengarahkan kursor di atas grafik akan menunjukkan pada Anda sumber dan tujuan probe, dan menunjukkan bahwa ada kehilangan paket 30,97% untuk probe ini pada 21 November pukul 02:41:30.

- Waktu round-trip — Grafik ini akan menampilkan garis untuk setiap probe yang juga menunjukkan waktu round-trip untuk setiap probe. Legenda yang ada di bagian bawah halaman menampilkan setiap probe dalam monitor dan diberi kode warna agar unik. Dengan mengarahkan kursor di atas probe dalam bagan ini akan menampilkan subnet sumber, alamat IP tujuan, dan waktu round-trip. Contoh berikut menunjukkan bahwa pada hari Selasa, 21 November pukul 21:45:30, waktu round-trip untuk probe dari sebuah subnet ke alamat IP 127.0.0.1 adalah 0,075 detik.

Detail monitor

Halaman detail Monitor menampilkan detail-detail tentang monitor Anda, termasuk probe. Di halaman ini Anda akan dapat mengelola tanda atau menambahkan probe. Halaman ini dibagi ke dalam tiga bagian berikut:

- Detail monitor — Halaman ini memberikan detail tentang monitor Anda. Informasi yang ada di bagian ini tidak dapat diedit. Namun demikian, Anda dapat memilih tautan Nama peran untuk melihat detail peran terkait layanan Monitor Jaringan.
- Probe — Bagian ini menampilkan daftar semua probe yang dikaitkan dengan monitor. Pilih tautan VPC atau ID Subnet untuk membuka detail VPC atau subnet di Konsol VPC Amazon. Anda juga dapat melakukan modifikasi probe, termasuk mengaktifkan atau menonaktifkannya. Untuk informasi selengkapnya, lihat the section called “Menggunakan monitor dan probe”.

  Bagian Probe menampilkan informasi tentang setiap probe yang disiapkan untuk monitor itu, termasuk ID probe, ID VPC, ID Subnet, alamat IP, Protokol, dan apakah Status probe Aktif atau Tidak Aktif. Jika Anda telah mengatur alarm untuk probe, Status alarm saat ini akan ditampilkan. OK menunjukkan bahwa tidak ada peristiwa metrik yang memicu alarm apa pun; Dalam alarm menunjukkan bahwa metrik yang Anda atur CloudWatch memicu alarm. Jika tidak ada status yang ditampilkan untuk probe, maka tidak ada CloudWatch alarm yang diatur. Untuk informasi tentang jenis alarm probe Monitor Jaringan yang dapat Anda buat, lihatthe section called “Alarm probe”.
- Tanda — Melihat tanda saat ini untuk monitor. Anda dapat menambahkan atau menghapus tanda dengan memilih Kelola tag. Ini akan membuka halaman Edit probe. Untuk informasi selengkapnya tentang cara mengedit tag, silakan lihat the section called “Mengedit sebuah monitor”.

Dasbor probe

Anda dapat menggunakan dasbor Amazon CloudWatch Network Monitor untuk melihat kesehatan AWS jaringan, dan informasi tentang waktu pulang-pergi tertentu dan kehilangan paket untuk probe tertentu. Ada dua dasbor probe, yaitu Gambaran umum dan Detail probe.

Anda dapat membuat CloudWatch alarm untuk mengatur kehilangan paket dan ambang batas metrik waktu pulang-pergi. Ketika ambang batas tercapai untuk metrik, CloudWatch alarm memberi tahu Anda. Untuk informasi tentang cara membuat alarm probe, silakan lihat the section called “Alarm probe”.

Cara mengakses dasbor probe

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/, dan kemudian di bawah Network Monitoring, pilih Network Monitor.
2. Di bagian Monitor jaringan, pilih tautan Nama untuk membuka dasbor monitor.
3. Pilih tautan ID untuk melihat dasbor untuk probe tersebut.

Ikhtisar

Halaman Gambaran umum akan menampilkan informasi berikut untuk probe Anda:

- Detail Indikator Kondisi Kesehatan Jaringan AWS — Ini akan menyediakan kondisi kesehatan keseluruhan hanya jaringan AWS. Statusnya bisa Sehat atau Terdegradasi. Status Terdegradasi menunjukkan bahwa ada masalah dengan jaringan AWS, dan tidak menunjukkan apakah ada masalah dengan probe Anda.
- Kehilangan paket — Jumlah paket yang hilang dari subnet sumber ke alamat IP tujuan untuk probe ini.
- Waktu round-trip — Waktu yang dibutuhkan, dalam milidetik, untuk sebuah paket dari subnet sumber untuk mencapai alamat IP tujuan dan kemudian kembali lagi.

Detail probe

Halaman detail Probe ini menampilkan detail tentang sebuah probe. Di halaman ini Anda dapat mengedit probe. Untuk informasi selengkapnya, lihat the section called “Menggunakan monitor dan probe”.

- Detail probe — Halaman ini akan memberikan informasi umum tentang probe. Informasi yang ada di bagian ini tidak dapat diedit.
- Sumber dan tujuan probe — Bagian ini menampilkan detail tentang probe. Pilih tautan VPC atau ID Subnet untuk membuka detail VPC atau subnet di Konsol VPC Amazon. Anda juga dapat melakukan modifikasi probe, termasuk mengaktifkan atau menonaktifkannya.
- Tanda — Melihat tanda saat ini untuk monitor. Anda dapat menambahkan atau menghapus tanda dengan memilih Kelola tag. Ini akan membuka halaman Edit probe. Untuk informasi selengkapnya tentang cara mengedit tag, silakan lihat the section called “Mengedit probe ”.

Kuota Monitor Jaringan

Berikut ini adalah kuota Monitor Jaringan:

Kuota	Default	Dapat disesuaikan
Jumlah maksimum monitor per akun untuk setiap Wilayah AWS	100	Ya
Jumlah probe maksimum untuk setiap monitor	24	Ya
Jumlah maksimum probe per subnet untuk setiap monitor	4	Ya

Keamanan data dan perlindungan data di Monitor Jaringan

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

Keamanan adalah tanggung jawab bersama antara AWS dan Anda. Model tanggung jawab bersama menjelaskan hal ini sebagai keamanan dari cloud dan keamanan dalam cloud:

- Keamanan cloud – AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan layanan-layanan AWS di AWS Cloud Cloud. AWS juga memberikan Anda layanan yang dapat digunakan dengan aman. Auditor pihak ketiga menguji dan memverifikasi secara berkala efektivitas keamanan kami sebagai bagian dari Program Kepatuhan AWS. Untuk mempelajari tentang program kepatuhan yang berlaku untuk Amazon CloudWatch Network Monitor, lihat AWSLayanan dalam Lingkup berdasarkan AWS Layanan Program Kepatuhan .
- Keamanan di cloud – Tanggung jawab Anda ditentukan menurut layanan AWS yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, yang mencakup sensitivitas data Anda, persyaratan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan Monitor CloudWatch Jaringan. Topik berikut menunjukkan cara mengonfigurasi

Monitor CloudWatch Jaringan untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga mempelajari cara menggunakan AWS layanan lain yang membantu Anda memantau dan mengamankan sumber daya Monitor CloudWatch Jaringan Anda.

Topik

Perlindungan data di Monitor CloudWatch Jaringan Amazon

Model tanggung jawab AWS bersama model berlaku untuk perlindungan data di Amazon CloudWatch Network Monitor. Sebagaimana diuraikan dalam model ini, AWS bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk memelihara kendali atas isi yang dihost pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Lihat informasi yang lebih lengkap tentang privasi data dalam Pertanyaan Umum Privasi Data. Lihat informasi tentang perlindungan data di Eropa di pos blog Model Tanggung Jawab Bersama dan GDPR AWS di Blog Keamanan AWS.

Untuk tujuan perlindungan data, sebaiknya lindungi kredensial Akun AWS dan siapkan untuk masing-masing pengguna AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan sumber daya AWS. Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pengelogan aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi enkripsi AWS, bersama semua kontrol keamanan bawaan dalam Layanan AWS.
- Gunakan layanan keamanan terkelola lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-2 ketika mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi yang lebih lengkap tentang titik akhir FIPS yang tersedia di Standar Pemrosesan Informasi Federal (FIPS) 140-2.

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan CloudWatch Network Monitor atau lainnya Layanan AWS menggunakan konsol, APIAWS CLI, atau AWS SDK. Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

Keamanan Infrastruktur di Monitor CloudWatch Jaringan Amazon

Sebagai layanan terkelola, Amazon CloudWatch Network Monitor dilindungi oleh prosedur keamanan jaringan AWS global yang dijelaskan dalam whitepaper Amazon Web Services: Overview of Security Processes.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses Monitor CloudWatch Jaringan melalui jaringan. Klien harus mendukung Keamanan Lapisan Pengangkutan (TLS) 1.0 atau versi yang lebih baru. Kami merekomendasikan TLS 1.2 atau versi yang lebih baru. Klien juga harus mendukung suite cipher dengan perfect forward secrecy (PFS) seperti Ephemeral Diffie-Hellman (DHE) atau Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

Selain itu, permintaan harus ditandatangani dengan menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan pengguna utama IAM. Atau Anda dapat menggunakan AWS Security Token Service (AWS STS) untuk menghasilkan kredensial keamanan sementara untuk menandatangani permintaan.

Manajemen Identitas dan Akses untuk Monitor CloudWatch Jaringan Amazon

Identity and Access Management (IAM) AWS adalah sebuah layanan AWS yang membantu seorang administrator dalam mengendalikan akses ke sumber daya AWS secara aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diberi wewenang (memiliki izin) untuk menggunakan CloudWatch sumber daya Monitor Jaringan. IAM adalah layanan AWS yang dapat Anda gunakan tanpa dikenakan biaya tambahan. Anda dapat menggunakan fitur-fitur IAM untuk memungkinkan pengguna, layanan, dan aplikasi lain untuk menggunakan sumber daya AWS Anda sepenuhnya atau secara terbatas tanpa perlu membagikan kredensial keamanan Anda.

Secara default, para pengguna IAM tidak memiliki izin untuk membuat, melihat, atau memodifikasi sumber daya AWS. Untuk mengizinkan seorang pengguna IAM mengakses sumber daya, seperti jaringan global, dan melakukan tugas, Anda harus:

- Membuat sebuah kebijakan IAM yang memberikan izin kepada pengguna IAM untuk menggunakan sumber daya tertentu dan tindakan API tertentu yang mereka butuhkan
- Melampirkan kebijakan untuk pengguna IAM atau ke grup yang dimiliki pengguna tersebut

Saat Anda melampirkan sebuah kebijakan ke seorang pengguna atau grup pengguna, kebijakan itu mengizinkan atau menolak izin pengguna untuk melakukan tugas yang ditentukan pada sumber daya yang ditentukan.

Kunci syarat

Elemen `Condition` (atau blok Syarat) akan memungkinkan Anda menentukan syarat yang menjadi dasar suatu pernyataan berlaku. Elemen Syarat bersifat opsional. Anda dapat membangun ekspresi bersyarat yang menggunakan operator syarat, misalnya sama dengan atau kurang dari, untuk mencocokkan syarat dalam kebijakan dengan nilai-nilai yang diminta. Untuk informasi selengkapnya, silakan lihat Elemen Kebijakan IAM JSON: Syarat operator dalam Panduan Pengguna AWS Identity and Access Management.

Jika Anda menentukan beberapa elemen `Condition` dalam satu pernyataan, atau beberapa kunci dalam satu elemen `Condition`, AWS akan mengevaluasinya dengan menggunakan operasi `AND` logis. Jika Anda menentukan beberapa niilai untuk satu kunci persyaratan, AWS akan mengevaluasi syarat tersebut menggunakan operasi `OR` yang logis. Semua kondisi harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan kondisi. Sebagai contoh, Anda dapat memberikan izin kepada pengguna IAM untuk mengakses sumber daya hanya jika izin tersebut mempunyai tanda yang sesuai dengan nama pengguna IAM mereka.

Anda dapat melampirkan tag ke sumber daya Monitor CloudWatch Jaringan atau meneruskan tag dalam permintaan ke Cloud WAN. Untuk mengendalikan akses berdasarkan tag, Anda harus memberikan informasi tentang tanda di elemen syarat dari sebuah kebijakan dengan menggunakan kunci syarat `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`. Silakan lihat Elemen kebijakan IAM JSON: Syarat dalam Panduan Pengguna AWS Identity and Access Management untuk mendapatkan informasi selengkapnya.

Untuk melihat semua kunci persyaratan global AWS, silakan lihat Kunci konteks syarat global AWS di Panduan Pengguna AWS Identity and Access Management.

Memberikan tanda pada sumber daya jaringan inti

Sebuah tanda merupakan label metadata yang Anda tetapkan ke sebuah sumber daya AWS oleh Anda atau AWS. Setiap tanda terdiri dari sebuah kunci dan sebuah nilai. Untuk tanda yang Anda tetapkan, Anda menentukan kunci dan nilai. Sebagai contoh, Anda dapat menentukan kunci sebagai `purpose` dan nilai sebagai `test` untuk satu sumber daya. Tanda membantu Anda melakukan hal berikut:

- Mengidentifikasi dan mengorganisasi sumber daya AWS Anda. Banyak layanan AWS yang mendukung pemberian tag, sehingga Anda dapat menetapkan tanda yang sama ke sumber daya dari layanan yang berbeda untuk menunjukkan bahwa sumber daya tersebut terkait.
- Mengendalikan akses ke sumber daya AWS Anda. Untuk informasi selengkapnya, lihat Mengontrol akses ke sumber daya AWSmenggunakan tanda dalam Panduan Pengguna AWS Identity and Access Management.

Bagaimana Amazon CloudWatch Network Monitor bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke Monitor CloudWatch Jaringan, pelajari fitur IAM apa yang tersedia untuk digunakan dengan Monitor CloudWatch Jaringan.

Fitur IAM yang dapat Anda gunakan dengan Amazon CloudWatch Network Monitor

Fitur IAM	CloudWatch Dukungan Monitor Jaringan
Kebijakan berbasis identitas	Ya
Kebijakan berbasis sumber daya	Tidak
Tindakan kebijakan	Ya
Sumber daya kebijakan	Ya
Kunci persyaratan kebijakan	Ya
ACL	Tidak
ABAC (tanda dalam kebijakan)	Parsial

Fitur IAM	CloudWatch Dukungan Monitor Jaringan
Kredensial sementara	Ya
Izin pengguna utama	Ya
Peran layanan	Tidak
Peran terkait layanan	Ya

Untuk mendapatkan tampilan tingkat tinggi tentang cara kerja Monitor CloudWatch Jaringan dan AWS layanan lainnya dengan sebagian besar fitur IAM, lihat AWSlayanan yang bekerja dengan IAM di Panduan Pengguna IAM.

Kebijakan berbasis identitas untuk Amazon Network Monitor CloudWatch

Mendukung kebijakan berbasis identitas	Ya

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 pengguna dan peran, di sumber daya mana, dan dengan ketentuan apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan tindakan dan sumber daya yang diizinkan atau ditolak, serta ketentuan terkait jenis tindakan yang diizinkan atau ditolak. Anda tidak dapat menentukan pengguna utama dalam kebijakan berbasis identitas karena kebijakan ini berlaku untuk pengguna atau peran yang dilampiri kebijakan. Untuk mempelajari semua elemen yang dapat digunakan dalam kebijakan JSON, lihat Referensi elemen kebijakan JSON IAM dalam Panduan Pengguna IAM.

Contoh kebijakan berbasis identitas untuk Network Monitor CloudWatch

Untuk melihat contoh kebijakan berbasis identitas Monitor CloudWatch Jaringan, lihat. Contoh kebijakan berbasis identitas untuk Amazon CloudWatch

Kebijakan berbasis sumber daya dalam Network Monitor CloudWatch

Mendukung kebijakan berbasis sumber daya	Tidak

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 yang dilampiri kebijakan tersebut, kebijakan ini menentukan jenis tindakan yang dapat dilakukan oleh pengguna utama tertentu di sumber daya tersebut dan apa ketentuannya. Anda harus menentukan pengguna utama dalam kebijakan berbasis sumber daya. Pengguna utama dapat mencakup akun, pengguna, peran, pengguna gabungan, atau Layanan AWS.

Untuk mengaktifkan akses lintas akun, Anda dapat menentukan seluruh akun atau entitas IAM di akun lain sebagai pengguna utama dalam kebijakan berbasis sumber daya. Menambahkan pengguna utama lintas akun ke kebijakan berbasis sumber daya bagian dari membangun hubungan kepercayaan. Ketika pengguna utama dan sumber daya berada di Akun AWS yang berbeda, administrator IAM di akun tepercaya juga harus memberikan izin kepada entitas pengguna utama (pengguna atau peran) untuk mengakses sumber daya. Izin diberikan dengan melampirkan kebijakan berbasis identitas ke entitas tersebut. Namun, jika kebijakan berbasis sumber daya memberikan akses kepada pengguna utama dalam akun yang sama, kebijakan berbasis identitas lainnya tidak diperlukan. Untuk informasi selengkapnya, lihat Perbedaan peran IAM dengan kebijakan berbasis sumber daya di Panduan Pengguna IAM.

Tindakan kebijakan untuk Monitor CloudWatch Jaringan

Mendukung tindakan kebijakan	Ya

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan siapa yang memiliki akses ke apa. Yaitu, pengguna utama mana yang dapat melakukan tindakan pada sumber daya apa, 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 seperti operasi API AWS 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.

Menyertakan tindakan dalam suatu kebijakan untuk memberikan izin melakukan operasi terkait.

Untuk melihat daftar tindakan Monitor CloudWatch Jaringan, lihat Tindakan yang ditentukan oleh Amazon CloudWatch Network Monitor di Referensi Otorisasi Layanan.

Tindakan kebijakan di Monitor CloudWatch Jaringan menggunakan awalan berikut sebelum tindakan:

```
networkmonitor
```

Untuk menetapkan secara spesifik beberapa tindakan dalam satu pernyataan, pisahkan tindakan-tindakan tersebut dengan koma.

```
"Action": [
      "networkmonitor:action1",
      "networkmonitor:action2"
         ]
```

Untuk melihat contoh kebijakan berbasis identitas Monitor CloudWatch Jaringan, lihat. Contoh kebijakan berbasis identitas untuk Amazon CloudWatch

Sumber daya kebijakan untuk Monitor CloudWatch Jaringan

Mendukung sumber daya kebijakan	Ya

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

Elemen kebijakan JSON `Resource` menentukan objek atau beberapa 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 di tingkat sumber daya, misalnya operasi pencantuman, gunakan wildcard (*) untuk mengindikasikan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

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

Untuk melihat daftar jenis sumber daya Monitor CloudWatch Jaringan dan ARNnya, lihat Sumber daya yang ditentukan oleh Amazon CloudWatch Network Monitor di Referensi Otorisasi Layanan. Untuk mempelajari tindakan mana yang dapat Anda tentukan ARN dari setiap sumber daya, lihat Tindakan yang ditentukan oleh Amazon CloudWatch Network Monitor.

Kunci kondisi kebijakan untuk Monitor CloudWatch Jaringan

Mendukung kunci kondisi kebijakan spesifik layanan	Ya

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

Elemen `Condition` (atau blok `Condition`) memungkinkan Anda menentukan kondisi di mana suatu pernyataan akan diterapkan. Elemen `Condition` bersifat opsional. Anda dapat membuat ekspresi kondisional yang menggunakan operator kondisi, misalnya sama dengan atau kurang dari, untuk mencocokkan kondisi dalam kebijakan dengan nilai-nilai yang diminta.

Jika Anda menentukan beberapa elemen `Condition` dalam satu pernyataan, atau beberapa kunci dalam satu elemen `Condition`, AWS akan mengevaluasinya dengan menggunakan operasi `AND` logis. Jika Anda menentukan beberapa niilai untuk satu kunci persyaratan, AWS akan mengevaluasi syarat tersebut menggunakan operasi `OR` yang logis. Semua kondisi harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan kondisi. Sebagai contoh, Anda dapat memberikan izin kepada pengguna IAM untuk mengakses sumber daya hanya jika izin tersebut mempunyai tanda yang sesuai dengan nama pengguna IAM mereka. Untuk informasi selengkapnya, silakan lihat Elemen kebijakan IAM: variabel dan tanda di Panduan Pengguna IAM.

AWS mendukung kunci kondisi global dan kunci kondisi spesifik layanan. Untuk melihat semua kunci kondisi global AWS, lihat kunci konteks kondisi global AWS dalam Panduan Pengguna IAM.

Untuk melihat daftar kunci kondisi Monitor CloudWatch Jaringan, lihat Kunci kondisi untuk Monitor CloudWatch Jaringan Amazon di Referensi Otorisasi Layanan. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan kunci kondisi, lihat Tindakan yang ditentukan oleh Amazon CloudWatch Network Monitor.

ACL di Monitor CloudWatch Jaringan

Mendukung ACL	Tidak

Daftar kontrol akses (ACL) mengontrol pengguna utama (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL sama dengan kebijakan berbasis sumber daya, meskipun tidak menggunakan format dokumen kebijakan JSON.

ABAC dengan Monitor CloudWatch Jaringan

Mendukung ABAC (tanda dalam kebijakan)	Parsial

Kontrol akses berbasis atribut (ABAC) adalah strategi otorisasi yang menentukan izin berdasarkan atribut. Di AWS, atribut ini disebut tag. Anda dapat melampirkan tanda ke entitas IAM (pengguna atau peran) dan ke banyak sumber daya AWS. Pemberian tanda ke entitas dan sumber daya adalah langkah pertama dari ABAC. Kemudian rancanglah kebijakan ABAC untuk mengizinkan operasi-operasi ketika tanda milik pengguna utama cocok dengan tanda yang ada di sumber daya yang ingin diakses.

ABAC sangat berguna di lingkungan yang berkembang dengan cepat dan berguna dalam situasi di mana pengelolaan kebijakan menjadi rumit.

Untuk mengendalikan akses berdasarkan tag, berikan informasi tentang tanda di elemen syarat dari sebuh kebijakan dengan menggunakan kunci-kunci persyaratan `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`.

Jika sebuah layanan mendukung ketiga kunci kondisi untuk setiap jenis sumber daya, nilainya adalah Ya untuk layanan tersebut. Jika suatu layanan mendukung ketiga kunci kondisi hanya untuk beberapa jenis sumber daya, nilainya adalah Parsial.

Untuk informasi selengkapnya tentang ABAC, lihat Apa itu ABAC? di Panduan Pengguna IAM. Untuk melihat tutorial terkait langkah-langkah penyiapan ABAC, lihat Menggunakan kontrol akses berbasis atribut (ABAC) di Panduan Pengguna IAM.

Menggunakan kredensyal sementara dengan Network Monitor CloudWatch

Mendukung kredensial sementara	Ya

Beberapa Layanan AWS tidak berfungsi saat Anda masuk menggunakan kredensial sementara. Sebagai informasi tambahan, termasuk tentang Layanan AWS mana saja yang berfungsi dengan kredensial sementara, lihat Layanan AWS yang berfungsi dengan IAM di Panduan Pengguna IAM.

Anda menggunakan kredensial sementara jika Anda masuk ke AWS Management Console menggunakan metode apa pun kecuali nama pengguna dan kata sandi. Misalnya, ketika Anda mengakses AWS dengan menggunakan tautan masuk tunggal (SSO) milik perusahaan Anda, proses itu secara otomatis akan membuat kredensial temporer. Anda juga akan membuat kredensial sementara secara otomatis saat masuk ke konsol sebagai pengguna dan kemudian beralih peran. Untuk informasi selengkapnya tentang cara beralih peran, lihat Beralih peran (konsol) di Panduan Pengguna IAM.

Anda dapat membuat kredensial sementara secara manual menggunakan AWS CLI atau AWS API. Anda kemudian dapat menggunakan kredensial sementara untuk mengakses AWS. AWS menyarankan Anda membuat kredensial sementara secara dinamis, alih-alih menggunakan kunci akses jangka panjang. Untuk informasi selengkapnya, lihat Kredensial keamanan sementara di IAM.

Izin utama lintas layanan untuk Monitor Jaringan CloudWatch

Mendukung sesi akses maju (FAS)	Ya

Jika menggunakan pengguna IAM atau peran IAM untuk melakukan tindakan di AWS, Anda akan dianggap sebagai pengguna utama. Jika menggunakan beberapa layanan, Anda mungkin melakukan tindakan yang kemudian dilanjutkan oleh tindakan lain di layanan yang berbeda. FAS menggunakan izin dari pengguna utama untuk memanggil Layanan AWS, yang dikombinasikan dengan Layanan AWS yang diminta untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya diajukan saat layanan menerima permintaan yang memerlukan interaksi dengan Layanan AWS lain atau sumber daya lain untuk diselesaikan. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Meneruskan sesi akses.

Peran layanan untuk Monitor CloudWatch Jaringan

Mendukung peran layanan	Tidak

Peran layanan adalah sebuah peran IAM yang diambil oleh sebuah layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

> **Warning**
>
> Mengubah izin untuk peran layanan dapat merusak fungsionalitas Monitor CloudWatch Jaringan. Edit peran layanan hanya jika CloudWatch Network Monitor memberikan panduan untuk melakukannya.

Menggunakan peran terkait layanan untuk CloudWatch Monitor Jaringan

Mendukung peran yang terkait layanan	Ya

Peran yang terkait layanan adalah jenis peran layanan yang terkait dengan Layanan AWS. Layanan ini dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan akan muncul di Akun AWS Anda dan dimiliki oleh layanan tersebut. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

Untuk detail tentang pembuatan atau pengelolaan peran terkait layanan, lihat Layanan AWS yang berfungsi dengan IAM. Temukan sebuah layanan dalam tabel yang memiliki `Yes` di kolom Peran terkait layanan. Pilih tautan Ya untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Contoh kebijakan berbasis identitas untuk Network Monitor CloudWatch

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi sumber daya Monitor CloudWatch Jaringan. Pengguan dan peran tersebut juga tidak dapat melakukan tugas dengan menggunakan AWS Management Console, AWS Command Line Interface (AWS CLI), atau API AWS. Untuk memberikan izin kepada pengguna untuk melakukan tindakan pada sumber daya

yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat menjalankan peran.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM menggunakan contoh dokumen kebijakan JSON ini, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Untuk detail tentang tindakan dan jenis sumber daya yang ditentukan oleh Monitor CloudWatch Jaringan, termasuk format ARN untuk setiap jenis sumber daya, lihat Kunci tindakan, sumber daya, dan kondisi untuk Monitor CloudWatch Jaringan Amazon di Referensi Otorisasi Layanan.

Topik

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya Monitor CloudWatch Jaringan di akun Anda. Tindakan ini dikenai biaya untuk Akun AWS Anda. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulai menggunakan kebijakan yang dikelola AWS dan beralih ke izin dengan hak akses paling rendah – Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan yang dikelola AWS yang memberikan izin untuk banyak kasus penggunaan umum. Kebijakan ini ada di Akun AWS Anda. Sebaiknya Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola pelanggan AWS yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat kebijakan yang dikelola AWS atau kebijakan yang dikelola AWS untuk fungsi pekerjaan di 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 melakukan ini dengan menentukan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, juga dikenal sebagai izin hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan IAM untuk menerapkan izin, lihat Kebijakan dan izin di IAM di Panduan Pengguna IAM.

- Gunakan kondisi dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Misalnya, Anda dapat menulis syarat kebijakan untuk menentukan bahwa semua pengajuan harus dikirim menggunakan SSL. Anda juga dapat menggunakan kondisi untuk memberi akses ke tindakan layanan jika digunakan melalui Layanan AWS yang spesifk, seperti AWS CloudFormation. Untuk informasi selengkapnya, lihat Elemen kebijakan JSON IAM: Syarat di Panduan Pengguna IAM.
- Menggunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda guna memastikan izin yang aman dan berfungsi – 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 Analizer Akses IAM di Panduan Pengguna IAM.
- Wajibkan autentikasi multi-faktor (MFA) – Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Akun AWS Anda, aktifkan MFA untuk keamanan tambahan. Untuk mewajibkan MFA saat operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda. Untuk informasi selengkapnya, lihat Mengonfigurasi akses API yang dilindungi MFA di Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat Praktik terbaik keamanan di IAM di Panduan Pengguna IAM.

Menggunakan konsol CloudWatch Network Monitor

Untuk mengakses konsol Monitor CloudWatch Jaringan Amazon, Anda harus memiliki set izin minimum. Izin ini harus memungkinkan Anda untuk membuat daftar dan melihat detail tentang sumber daya Monitor CloudWatch Jaringan di AndaAkun 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.

Anda tidak perlu memberikan izin konsol minimum untuk pengguna yang melakukan panggilan hanya ke AWS CLI atau API AWS. Sebaliknya, izinkan akses hanya ke tindakan yang cocok dengan operasi API yang coba dilakukan.

Untuk memastikan bahwa pengguna dan peran masih dapat menggunakan konsol Monitor CloudWatch Jaringan, lampirkan juga Monitor CloudWatch Jaringan `ConsoleAccess` atau kebijakan `ReadOnly` AWS terkelola ke entitas. Untuk informasi selengkapnya, lihat Menambahkan izin ke pengguna di Panduan Pengguna IAM.

Izinkan pengguna melihat izin mereka sendiri

Contoh ini menunjukkan cara membuat kebijakan yang mengizinkan para pengguna IAM melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan pada konsol atau menggunakan AWS CLI atau AWS API secara terprogram.

```
{
    "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": "*"
        }
    ]
}
```

Pemecahan Masalah CloudWatch Jaringan Monitor identitas dan akses

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan CloudWatch Network Monitor dan IAM.

Topik

Saya tidak berwenang untuk melakukan tindakan di CloudWatch Network Monitor

Jika Anda menerima pesan kesalahan bahwa Anda tidak memiliki otorisasi untuk melakukan tindakan, kebijakan Anda harus diperbarui agar Anda dapat melakukan tindakan tersebut.

Contoh kesalahan berikut terjadi ketika pengguna IAM `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya *`my-example-widget`* rekaan, tetapi tidak memiliki izin `networkmonitor:`*`GetWidget`* rekaan.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 networkmonitor:GetWidget on resource: my-example-widget
```

Dalam hal ini, kebijakan untuk pengguna `mateojackson` harus diperbarui untuk mengizinkan akses ke sumber daya *`my-example-widget`* dengan menggunakan tindakan `networkmonitor:`*`GetWidget`*.

Jika Anda membutuhkan bantuan, hubungi administrator AWS Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan bahwa Anda tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran ke Monitor CloudWatch Jaringan.

Sebagian Layanan AWS mengizinkan Anda untuk memberikan peran yang sudah ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait-layanan. Untuk melakukan tindakan tersebut, Anda harus memiliki izin untuk memberikan peran pada layanan tersebut.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol untuk melakukan tindakan di CloudWatch Network Monitor. 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 membutuhkan bantuan, hubungi administrator AWS Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses sumber daya Monitor CloudWatch Jaringan saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau pengguna 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 (ACL), Anda dapat menggunakan kebijakan tersebut untuk memberi pengguna akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa hal berikut:

- Untuk mengetahui apakah CloudWatch Network Monitor mendukung fitur-fitur ini, lihatBagaimana Amazon CloudWatch bekerja dengan IAM.
- Untuk mempelajari cara memberikan akses ke sumber daya di seluruh Akun AWS yang Anda miliki, lihat Menyediakan akses ke pengguna IAM di Akun AWS lainnya yang Anda miliki dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses ke sumber daya Anda ke pihak ketiga Akun AWS, lihat Menyediakan akses ke akun Akun AWS yang dimiliki oleh pihak ketiga dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat Menyediakan akses ke pengguna terautentikasi eksternal (gabungan identitas) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara penggunaan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Perbedaan antara peran IAM dan kebijakan berbasis sumber daya di Panduan Pengguna IAM.

AWSkebijakan terkelola untuk CloudWatch Network Monitor

Menambahkan izin ke para pengguna, grup, dan peran lebih mudah dilakukan dengan menggunakan kebijakan terkelola AWS dibandingkan dengan menulis kebijakan sendiri. Dibutuhkan waktu dan keahlian untuk membuat kebijakan terkelola pelanggan IAM yang hanya menyediakan izin sesuai kebutuhan tim Anda. Untuk mulai dengan cepat, Anda dapat menggunakan kebijakan-kebijakan terkelola AWS kami. Kebijakan-kebijakan ini mencakup kasus penggunaan umum dan tersedia di akun AWS Anda. Untuk informasi lebih lanjut tentang kebijakan-kebijakan terkelola AWS, lihat kebijakan terkelola AWS di Panduan Pengguna IAM.

Layanan AWS mempertahankan dan memperbarui kebijakan-kebijakan terkelola AWS. Anda tidak dapat mengubah izin yang ada dalam kebijakan-kebijakan yang dikelola AWS. Layanan terkadang menambahkan izin tambahan ke kebijakan yang dikelola AWS untuk mendukung fitur-fitur baru. Jenis pembaruan ini akan memengaruhi semua identitas (pengguna, grup, dan peran) di mana kebijakan tersebut dilampirkan. Layanan kemungkinan besar akan memperbarui kebijakan yang dikelola AWS saat ada fitur baru yang diluncurkan atau saat ada operasi baru yang tersedia. Layanan tidak menghapus izin yang ada di kebijakan yang dikelola AWS, sehingga pembaruan-pembaruan yang terjadi pada kebijakan tidak akan membuat izin yang ada rusak.

Selain itu, AWS mendukung kebijakan-kebijakan terkelola untuk fungsi tugas yang mencakup beberapa layanan. Sebagai contoh, kebijakan `ReadOnlyAccess` terkelola AWS menyediakan akses hanya-baca ke semua layanan dan sumber daya AWS. Saat layanan meluncurkan fitur baru, AWS menambahkan izin hanya-baca untuk operasi dan sumber daya baru. Untuk melihat daftar dan deskripsi dari kebijakan fungsi tugas, lihat kebijakan yang dikelola AWS untuk fungsi tugas di Panduan Pengguna IAM.

Kebijakan terkelola AWS: CloudWatchNetworkMonitorServiceRolePolicy

`CloudWatchNetworkMonitorServiceRolePolicy`Ini dilampirkan ke peran terkait layanan yang memungkinkan layanan untuk melakukan tindakan atas nama Anda dan mengakses sumber daya yang terkait dengan Monitor CloudWatch Jaringan. Anda dapat melampirkan kebijakan ini ke identitas-identitas IAM Anda. Untuk informasi selengkapnya, lihat the section called “Peran terkait layanan”.

CloudWatch Pembaruan Pemantauan Jaringan ke kebijakan AWS terkelola

Lihat detail tentang pembaruan kebijakan AWS terkelola untuk Pemantauan CloudWatch Jaringan sejak layanan ini mulai melacak perubahan ini pada November 2023.

Perubahan	Deskripsi	Tanggal
CloudWatchNetworkMonitorServiceRolePolicy: Kebijakan baru.	Kebijakan baru ditambahkan ke Monitor CloudWatch Jaringan.	27 November 2023
the section called “AWSServiceRoleForNetworkMonitor”. Peran baru.	Peran baru ditambahkan ke Monitor CloudWatch Jaringan.	27 November 2023

Izin IAM untuk Monitor Jaringan CloudWatch

Untuk menggunakan Monitor CloudWatch Jaringan Amazon, pengguna harus memiliki izin yang benar.

Untuk informasi selengkapnya tentang keamanan di Amazon CloudWatch, lihatManajemen identitas dan akses untuk Amazon CloudWatch.

Izin yang diperlukan untuk melihat monitor

Untuk melihat monitor untuk Monitor CloudWatch Jaringan Amazon diAWS Management Console, Anda harus masuk sebagai pengguna atau peran yang memiliki izin berikut:

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "networkmonitor:Get*",
                "networkmonitor:List*"
                ],
            "Resource": "*"
        }
    ]
}
```

Izin diperlukan untuk melihat monitor

Untuk membuat monitor di Amazon CloudWatch Network Monitor, pengguna harus memiliki izin untuk membuat peran terkait layanan yang terkait dengan Monitor Jaringan. Untuk mempelajari lebih lanjut tentang peran terkait layanan, silakan lihat Menggunakan peran terkait layanan untuk CloudWatch Monitor Jaringan.

Untuk membuat monitor untuk Monitor CloudWatch Jaringan Amazon diAWS Management Console, Anda harus masuk sebagai pengguna atau peran yang memiliki izin yang disertakan dalam kebijakan berikut.

> **Note**
>
> Jika Anda membuat sebuah kebijakan izin berbasis identitas yang lebih ketat, pengguna dengan kebijakan tersebut tidak akan dapat membuat monitor.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "networkmonitor:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
networkmonitor.amazonaws.com/AWSServiceRoleForNetworkMonitor",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "networkmonitor.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "iam:AttachRolePolicy",
                "iam:GetRole",
                "iam:PutRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/
networkmonitor.amazonaws.com/AWSServiceRoleForNetworkMonitor"
        },
        {
            "Action": [
                "ec2:CreateSecurityGroup",
                "ec2:CreateNetworkInterface",
                "ec2:CreateTags"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Menggunakan peran terkait layanan untuk CloudWatch Monitor Jaringan

Amazon CloudWatch Network Monitor menggunakan peran terkait layanan berikut untuk izin yang diperlukan untuk memanggil AWS layanan lain atas nama Anda:

- AWSServiceRoleForNetworkMonitor

AWSServiceRoleForNetworkMonitor

CloudWatch Network Monitoring menggunakan peran terkait layanan bernama `AWSServiceRoleForNetworkMonitor` untuk memperbarui dan mengelola monitor CloudWatch jaringan.

Peran terkait layanan `AWSServiceRoleForNetworkMonitor` memercayai layanan berikut untuk mengambil peran tersebut:

- `networkmonitor.amazonaws.com`

`CloudWatchNetworkMonitorServiceRolePolicy` dilampirkan ke peran terkait layanan dan memberikan akses untuk layanan untuk mengakses sumber daya VPC dan EC2 di akun Anda, serta mengelola monitor jaringan yang dibuat.

Pengelompokan izin

Kebijakan dikelompokkan ke dalam kumpulan izin berikut:

- **`cloudwatch`**- Ini memungkinkan kepala layanan untuk mempublikasikan metrik pemantauan jaringan ke CloudWatch sumber daya.
- **`ec2`** - Hal ini memungkinkan pengguna utama layanan untuk menggambarkan VPC dan subnet di akun Anda untuk membuat atau memperbarui monitor dan probe. Hal ini juga memungkinkan pengguna utama untuk membuat, memodifikasi, dan menghapus grup keamanan, antarmuka jaringan, dan izin terkait untuk mengonfigurasi monitor atau probe untuk mengirim lalu lintas pemantauan ke titik akhir Anda.

Untuk informasi lebih selengkapnya tentang kebijakan kepercayaan, silakan lihat the section called “Kebijakan terkelola AWS”.

Berikut ini menunjukkan `CloudWatchNetworkMonitorServiceRolePolicy`:

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "PublishCw",
   "Effect": "Allow",
   "Action": "cloudwatch:PutMetricData",
   "Resource": "*",
   "Condition": {
    "StringEquals": {
     "cloudwatch:namespace": "AWS/NetworkMonitor"
    }
   }
  },
  {
   "Sid": "DescribeAny",
   "Effect": "Allow",
   "Action": [
    "ec2:DescribeNetworkInterfaces",
    "ec2:DescribeNetworkInterfaceAttribute",
    "ec2:DescribeVpcs",
    "ec2:DescribeNetworkInterfacePermissions",
    "ec2:DescribeSubnets",
    "ec2:DescribeSecurityGroups"
```

```
    ],
    "Resource": "*"
   },
   {
    "Sid": "DeleteModifyEc2Resources",
    "Effect": "Allow",
    "Action": [
     "ec2:AuthorizeSecurityGroupEgress",
     "ec2:CreateNetworkInterfacePermission",
     "ec2:DeleteNetworkInterfacePermission",
     "ec2:RevokeSecurityGroupEgress",
     "ec2:ModifyNetworkInterfaceAttribute",
     "ec2:DeleteNetworkInterface",
     "ec2:DeleteSecurityGroup"
    ],
    "Resource": [
     "arn:aws:ec2:*:*:network-interface/*",
     "arn:aws:ec2:*:*:security-group/*"
    ],
    "Condition": {
     "StringEquals": {
      "aws:ResourceTag/ManagedByCloudWatchNetworkMonitor": "true"
     }
    }
   }
  ]
}
```

Membuat peran terkait layanan

`AWSServiceRoleForNetworkMonitor`

Anda tidak perlu membuat peran atau peran `AWSServiceRoleForNetworkMonitor` secara manual.

- CloudWatch Network Monitor menciptakan `AWSServiceRoleForNetworkMonitor` peran saat Anda membuat monitor jaringan pertama Anda. Peran ini akan berlaku untuk monitor berikutnya yang Anda buat.

Untuk membuat sebuah peran terkait layanan atas nama Anda, Anda harus memiliki izin yang diperlukan. Untuk informasi selengkapnya, silakan lihat Izin Peran Terkait Layanan di Panduan Pengguna IAM.

Mengedit peran terkait layanan

Anda dapat mengedit deskripsi `AWSServiceRoleForNetworkMonitor` dengan menggunakan IAM. Untuk informasi lebih lanjut, lihat Mengubah Peran Terkait Layanan di Panduan Pengguna IAM.

Hapus peran tertaut layanan

Jika Anda tidak perlu lagi menggunakan Monitor CloudWatch Jaringan, kami sarankan Anda menghapus `AWSServiceRoleForNetworkMonitor` peran tersebut.

Anda dapat menghapus peran terkait layanan ini hanya setelah Anda menghapus monitor jaringan Anda. Untuk informasi tentang menghapus monitor jaringan, silakan lihat Menghapus monitor jaringan.

Anda dapat menggunakan konsol IAM, CLI IAM, atau API IAM untuk menghapus peran terkait layanan . Untuk informasi selengkapnya, silakan lihat Menghapus Peran Terkait Layanan di Panduan Pengguna IAM.

Setelah Anda menghapus `AWSServiceRoleForNetworkMonitor` CloudWatch Network Monitor akan membuat peran lagi ketika Anda membuat monitor baru.

Wilayah yang Didukung untuk peran CloudWatch terkait layanan Monitor Jaringan

CloudWatch Network Monitor mendukung peran terkait layanan di semua Wilayah AWS tempat layanan tersedia. Untuk informasi selengkapnya, silakan lihat titik akhir AWS di Referensi Umum AWS.

Menghapus peran terkait layanan

Jika Anda tidak perlu lagi menggunakan Monitor CloudWatch Jaringan, kami sarankan Anda menghapus `AWSServiceRoleForNetworkMonitor` peran tersebut.

Anda dapat menghapus peran terkait layanan ini hanya setelah Anda menghapus monitor jaringan Anda. Untuk informasi tentang menghapus monitor jaringan, silakan lihat Menghapus monitor jaringan.

Anda dapat menggunakan konsol IAM, CLI IAM, atau API IAM untuk menghapus peran terkait layanan . Untuk informasi selengkapnya, silakan lihat Menghapus Peran Terkait Layanan di Panduan Pengguna IAM.

Setelah Anda menghapus `AWSServiceRoleForNetworkMonitor` CloudWatch Network Monitor akan membuat peran lagi ketika Anda membuat monitor baru.

Harga

Dengan Amazon CloudWatch Network Monitor, tidak ada biaya di muka atau komitmen jangka panjang. Penentuan harga untuk Monitor Jaringan memiliki dua komponen berikut:

- biaya sumber daya per jam yang dipantau, dan
- CloudWatch biaya metrik.

Saat Anda membuat sebuah monitor jaringan, Anda mengaitkan sumber daya dengannya untuk dipantau. Untuk Monitor Jaringan, ini akan menjadi subnet di Amazon Virtual Private Cloud (VPC) Anda. Setiap sumber daya yang dipantau akan memungkinkan Anda membuat hingga empat probe dari setiap subnet di VPC Anda ke empat tujuan. Untuk membantu mengontrol tagihan, Anda dapat menyesuaikan cakupan subnet dan cakupan IP on-premise dengan mengurangi jumlah sumber daya yang dipantau.

Untuk informasi selengkapnya tentang harga, lihat halaman CloudWatch harga Amazon.

Pemantauan Infrastruktur

Topik di bagian ini menjelaskan CloudWatch fitur yang dapat membantu Anda mendapatkan visibilitas operasional ke AWS sumber daya Anda.

Topik

Wawasan Kontainer

Gunakan CloudWatch Wawasan Kontainer untuk mengumpulkan, menggabungkan, dan meringkas metrik dan log dari aplikasi dan layanan mikro dalam kontainer Anda. Wawasan Kontainer tersedia untuk platform Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), dan Kubernetes di Amazon EC2. Container Insights mendukung pengumpulan metrik dari cluster yang digunakan untuk Amazon ECS AWS Fargate dan Amazon EKS.

CloudWatch Secara otomatis mengumpulkan metrik untuk banyak sumber daya, seperti CPU, memori, disk, dan jaringan. Wawasan Kontainer juga akan menyediakan informasi diagnostik, seperti kegagalan mengulang kembali kontainer, untuk membantu Anda melakukan isolasi atas masalah dan mengatasi masalah itu dengan cepat. Anda juga dapat menyetel CloudWatch alarm pada metrik yang dikumpulkan Container Insights.

Wawasan Kontainer mengumpulkan data sebagai peristiwa log performa dengan menggunakan format metrik tersemat. Peristiwa log performa ini adalah entri yang menggunakan sebuah skema JSON terstruktur yang memungkinkan data dengan kardinalitas tinggi dapat diserap dan disimpan dalam skala besar. Dari data ini, CloudWatch buat metrik agregat di tingkat cluster, node, pod, task, dan service sebagai CloudWatch metrik. Metrik yang dikumpulkan Container Insights tersedia di dasbor CloudWatch otomatis, dan juga dapat dilihat di bagian Metrik konsol. CloudWatch Metrik-metrik tidak terlihat sampai tugas kontainer telah berjalan selama beberapa waktu.

Saat Anda menerapkan Wawasan Kontainer, ia akan secara otomatis membuat suatu grup log untuk peristiwa log performa. Anda tidak harus membuat grup log ini sendiri.

Untuk membantu Anda mengelola biaya Wawasan Kontainer, CloudWatch tidak secara otomatis membuat semua metrik yang mungkin dari data log. Namun, Anda dapat melihat metrik tambahan dan tingkat perincian tambahan dengan menggunakan Wawasan CloudWatch Log untuk menganalisis peristiwa log kinerja mentah.

Dengan Wawasan Kontainer versi asli, metrik-metrik yang dikumpulkan dan log yang diserap sebagai metrik kustom. Dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS, metrik dan log Wawasan Kontainer dikenakan biaya per observasi, bukan dibebankan per metrik yang disimpan atau log yang diserap. Untuk informasi selengkapnya tentang CloudWatch harga, lihat CloudWatch Harga Amazon.

Di Amazon EKS dan Kubernetes, Container Insights menggunakan versi kontainer dari CloudWatch agen untuk menemukan semua kontainer yang sedang berjalan dalam sebuah cluster. Kemudian, Wawasan Kontainer mengumpulkan data performa di setiap lapisan tumpukan performa.

Container Insights mendukung enkripsi dengan log dan metrik yang dikumpulkannya. AWS KMS key Untuk mengaktifkan enkripsi ini, Anda harus mengaktifkan AWS KMS enkripsi secara manual untuk grup log yang menerima data Wawasan Kontainer. Hal ini menyebabkan Wawasan Kontainer mengenkripsi data ini menggunakan kunci KMS yang disediakan. Hanya kunci simetris yang didukung. Jangan gunakan kunci KMS asimetris untuk melakukan enkripsi pada grup log Anda.

Untuk informasi selengkapnya, lihat Mengenkripsi Data Log di CloudWatch Log Menggunakan AWS KMS.

Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS

Pada 6 November 2023, Wawasan Kontainer versi baru dirilis. Versi ini mendukung peningkatan observabilitas untuk klaster Amazon EKS yang berjalan di Amazon EC2 dan dapat mengumpulkan metrik terperinci dari klaster ini. Setelah instalasi selesai, ia secara otomatis akan mengumpulkan telemetri infrastruktur terperinci dan log kontainer untuk klaster-klaster Amazon EKS Anda. Anda kemudian dapat menggunakan dasbor yang dikuratori yang pada saat itu juga dapat digunakan untuk menelusuri telemetri aplikasi dan infrastruktur secara lebih dalam.

Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS mengumpulkan metrik kondisi, performa, dan status detail hingga tingkat kontainer, dan juga metrik bidang kontrol. Untuk informasi selengkapnya tentang metrik dan dimensi tambahan yang dikumpulkan, silakan lihat Metrik-metrik Wawasan Kontainer Amazon EKS dan Kubernetes.

Jika Anda menginstal Container Insights dengan menggunakan CloudWatch agen di klaster Amazon EKS di Amazon EC2 setelah 6 November 2023, Anda memiliki Wawasan Kontainer dengan observabilitas yang ditingkatkan untuk Amazon EKS. Jika tidak, Anda dapat memperbarui klaster Amazon EKS ke versi baru ini dengan mengikuti petunjuk yang ada di Meningkatkan ke Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS.

Wawasan Kontainer mendukung pengamatan CloudWatch lintas akun. Anda menggunakan satu akun pemantauan untuk memantau dan memecahkan masalah aplikasi Anda yang menjangkau beberapa AWS akun dalam satu Wilayah. Untuk informasi selengkapnya, lihat CloudWatch observabilitas lintas akun.

Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS tidak didukung di Fargate.

> **Note**
>
> Anda dapat mengetahui apakah Anda memiliki klaster yang dapat ditingkatkan ke Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS dengan menavigasi ke konsol Wawasan Kontainer. Untuk melakukannya, pilih Insights, Container Insights di panel navigasi konsol. CloudWatch Di konsol Wawasan Kontainer, sebuah spanduk akan menginformasikan kepada Anda jika Anda memiliki klaster Amazon EKS yang dapat ditingkatkan, dan ditautkan ke halaman pembaruan.

Platform-platform yang didukung

Wawasan Kontainer tersedia untuk platform Amazon Elastic Container Service, Amazon Elastic Kubernetes Service, dan Kubernetes di instans Amazon EC2.

- Untuk Amazon ECS, Wawasan Kontainer mengumpulkan metrik di tingkat klaster, tugas, dan layanan pada instans Server Linux dan Windows. Solusi ini hanya dapat mengumpulkan metrik di tingkat instans pada instans Linux.

  Untuk Amazon ECS, metrik-metrik jaringan tersedia hanya untuk kontainer di mode jaringan `bridge` dan mode jaringan `awsvpc`. Tidak tersedia untuk kontainer dalam mode jaringan `host`.
- Untuk Amazon Elastic Kubernetes Service, dan platform Kubernetes di instans Amazon EC2, Wawasan Kontainer hanya didukung di instans Linux.

CloudWatch gambar kontainer agen

Amazon menyediakan gambar kontainer CloudWatch agen di Amazon Elastic Container Registry. Untuk informasi selengkapnya, silakan lihat cloudwatch-agent di Amazon ECR.

Wilayah yang didukung

Wawasan Kontainer untuk Amazon ECS didukung di Wilayah berikut:

- AS Timur (Virginia Utara)
- AS Timur (Ohio)
- AS Barat (California Utara)
- AS Barat (Oregon)
- Afrika (Cape Town)
- Asia Pasifik (Hong Kong)
- Asia Pasifik (Hyderabad)
- Asia Pasifik (Jakarta)
- Asia Pasifik (Mumbai)
- Asia Pasifik (Osaka)
- Asia Pasifik (Seoul)
- Asia Pasifik (Singapura)
- Asia Pasifik (Tokyo)
- Asia Pasifik (Sydney)
- Kanada (Pusat)
- Eropa (Frankfurt)
- Eropa (Irlandia)
- Eropa (London)
- Eropa (Milan)
- Eropa (Paris)
- Eropa (Spanyol)
- Eropa (Stockholm)
- Eropa (Zürich)

- Timur Tengah (Bahrain)
- Timur Tengah (UEA)
- Amerika Selatan (Sao Paulo)
- AWS GovCloud (AS-Timur)
- AWS GovCloud (AS-Barat)
- China (Beijing)
- Tiongkok (Ningxia)

Wilayah yang Didukung untuk Amazon EKS dan Kubernetes

Wawasan Kontainer untuk Amazon EKS dan Kubernetes didukung di Wilayah berikut:

- AS Timur (Virginia Utara)
- AS Timur (Ohio)
- AS Barat (California Utara)
- AS Barat (Oregon)
- Asia Pasifik (Hong Kong)
- Asia Pasifik (Mumbai)
- Asia Pasifik (Seoul)
- Asia Pasifik (Singapura)
- Asia Pasifik (Sydney)
- Asia Pasifik (Tokyo)
- Kanada (Pusat)
- Tiongkok (Beijing)
- Tiongkok (Ningxia)
- Eropa (Frankfurt)
- Eropa (Irlandia)
- Eropa (London)
- Eropa (Paris)
- Eropa (Stockholm)
- Timur Tengah (Bahrain)
- Amerika Selatan (Sao Paulo)

- AWS GovCloud (AS-Timur)
- AWS GovCloud (AS-Barat)

Menyiapkan Wawasan Kontainer

Proses menyiapkan Wawasan Kontainer berbeda dengan menyiapkan Amazon ECS dan Amazon EKS dan Kubernetes.

Topik

Menyiapkan Wawasan Kontainer di Amazon ECS

Anda dapat menggunakan salah satu atau kedua opsi berikut untuk mengaktifkan Wawasan Kontainer di klaster Amazon ECS:

- Gunakan AWS Management Console atau AWS CLI untuk mulai mengumpulkan metrik tingkat klaster, tingkat tugas, dan tingkat layanan.
- Terapkan CloudWatch agen sebagai layanan daemon untuk mulai mengumpulkan metrik tingkat instans pada kluster yang dihosting di instans Amazon EC2.

Topik

Menyiapkan Wawasan Kontainer di Amazon ECS untuk metrik-metrik tingkat klaster dan layanan

Anda dapat mengaktifkan Wawasan Kontainer pada klaster Amazon ECS baru dan yang sudah ada. Wawasan Kontainer mengumpulkan metrik-metrik di tingkat klaster, tugas, dan layanan. Anda dapat mengaktifkan Container Insights menggunakan konsol Amazon ECS atau. AWS CLI

Jika Anda menggunakan Amazon ECS pada sebuah instans Amazon EC2, dan Anda ingin mengumpulkan metrik-metrik jaringan dan penyimpanan dari Wawasan Kontainer, luncurkan instans tersebut dengan menggunakan AMI yang mencakup agen Amazon ECS versi 1.29. Untuk informasi tentang pembaruan versi agen Anda, silakan lihat Memperbarui Agen Kontainer Amazon ECS

Anda dapat menggunakan izin AWS CLI untuk menetapkan tingkat akun untuk mengaktifkan Wawasan Kontainer untuk setiap kluster Amazon ECS baru yang dibuat di akun Anda. Untuk melakukan hal itu, masukkan perintah berikut.

```
aws ecs put-account-setting --name "containerInsights" --value "enabled"
```

> **Note**
>
> Jika AWS KMS kunci terkelola pelanggan yang Anda gunakan untuk metrik Amazon ECS Container Insights belum dikonfigurasi untuk berfungsi CloudWatch, Anda harus memperbarui kebijakan kunci untuk mengizinkan log terenkripsi di Log. CloudWatch Anda juga harus mengaitkan AWS KMS kunci Anda sendiri dengan grup log di bawah `/aws/ecs/containerinsights/`*`ClusterName`*`/performance`. Untuk informasi selengkapnya, lihat Mengenkripsi data log di CloudWatch Log menggunakan AWS Key Management Service.

Menyiapkan Wawasan Kontainer di klaster Amazon ECS yang sudah ada

Untuk mengaktifkan Wawasan Kontainer di sebuah klaster Amazon ECS yang sudah ada, masukkan perintah berikut. Anda harus menjalankan versi 1.16.200 atau yang lebih baru AWS CLI agar perintah berikut berfungsi.

```
aws ecs update-cluster-settings --cluster myCICluster --settings
 name=containerInsights,value=enabled
```

Menyiapkan Wawasan Kontainer di klaster Amazon ECS yang baru

Ada dua cara untuk mengaktifkan Wawasan Kontainer pada klaster Amazon ECS yang baru. Anda dapat mengonfigurasi Amazon ECS sehingga semua klaster baru diaktifkan secara bawaan untuk Wawasan Kontainer. Jika tidak, maka Anda dapat mengaktifkan sebuah klaster baru saat membuatnya.

Menggunakan AWS Management Console

Anda dapat mengaktifkan Wawasan Kontainer di semua klaster baru secara default, atau di sebuah klaster individual saat Anda membuatnya.

Untuk mengaktifkan Wawasan Kontainer di semua klaster baru secara default

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Di halaman navigasi, pilih Pengaturan Akun.
3. Pilih Perbarui.
4. Untuk menggunakan CloudWatch Container Insights secara default untuk cluster, di bawah CloudWatchContainer Insights, pilih atau hapus CloudWatch Container Insights.
5. Pilih Simpan perubahan.

Jika Anda belum menggunakan prosedur sebelumnya untuk mengaktifkan Wawasan Kontainer pada semua klaster baru secara bawaan, gunakan langkah-langkah berikut untuk membuat sebuah klaster dengan Wawasan Kontainer yang sudah diaktifkan.

Cara membuat sebuah klaster dengan Wawasan Kontainer yang sudah diaktifkan

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Klaster.
3. Pada halaman Klaster, pilih Buat klaster.
4. Pada Konfigurasi klaster, untuk Nama klaster, masukkan nama unik.

   Nama tersebut dapat berisi hingga 255 huruf (huruf besar dan huruf kecil), angka, dan tanda hubung.
5. Untuk mengaktifkan Wawasan Kontainer, perluas Pemantauan, lalu aktifkan Gunakan Wawasan Kontainer.

Sekarang, Anda dapat membuat penetapan tugas, mengeksekusi tugas, dan meluncurkan layanan dalam klaster. Untuk informasi selengkapnya, silakan lihat yang berikut ini:

- Membuat sebuah penetapan tugas
- Mengeksekusi tugas
- Membuat sebuah layanan

Menyiapkan Wawasan Kontainer di kluster Amazon ECS baru menggunakan AWS CLI

Untuk mengaktifkan Wawasan Kontainer di semua klaster baru secara bawaan, masukkan perintah berikut.

```
aws ecs put-account-setting --name "containerInsights" --value "enabled"
```

Jika Anda belum menggunakan prosedur sebelumnya untuk mengaktifkan Wawasan Kontainer pada semua klaster baru secara bawaan, maka Anda harus menggunakan langkah-langkah berikut untuk membuat sebuah klaster baru dengan Wawasan Kontainer yang sudah diaktifkan. Anda harus menjalankan AWS CLI versi 1.16.200 atau yang lebih baru untuk perintah berikut agar bisa digunakan.

```
aws ecs create-cluster --cluster-name myCICluster --settings
 "name=containerInsights,value=enabled"
```

Menonaktifkan Wawasan Kontainer di klaster-klaster Amazon ECS

Untuk menonaktifkan Wawasan Kontainer pada sebuah klaster Amazon ECS yang ada, Anda harus memasukkan perintah berikut.

```
aws ecs update-cluster-settings --cluster myCICluster --settings
 name=containerInsights,value=disabled
```

Menyiapkan Wawasan Kontainer di Amazon ECS menggunakan AWS Distro untuk OpenTelemetry

Gunakan bagian ini jika Anda ingin menggunakan AWS Distro OpenTelemetry untuk menyiapkan CloudWatch Wawasan Kontainer di kluster Amazon ECS. Untuk informasi lebih lanjut tentang AWS Distro untuk Telemetri Terbuka, lihat AWS Distro untuk. OpenTelemetry

Langkah-langkah ini mengasumsikan Anda sudah memiliki sebuah klaster yang menjalankan Amazon ECS. Untuk informasi selengkapnya tentang penggunaan AWS Distro untuk Open Telemetry dengan Amazon ECS dan menyiapkan klaster Amazon ECS untuk tujuan ini, lihat Menyiapkan AWS Distro untuk OpenTelemetry Kolektor di Amazon Elastic Container Service.

Langkah 1: Membuat sebuah peran tugas

Langkah pertama adalah membuat peran tugas di cluster yang akan digunakan AWS OpenTelemetry Kolektor.

Untuk membuat peran tugas untuk AWS Distro untuk OpenTelemetry

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi, silakan pilih Kebijakan dan kemudian pilih Buat kebijakan.
3. Pilih tab JSON dan salin kebijakan berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:DescribeLogGroups",
                "ssm:GetParameters"
            ],
            "Resource": "*"
        }
    ]
}
```

4. Pilih Tinjau kebijakan.
5. Untuk nama, masukkan **`AWSDistroOpenTelemetryPolicy`**, kemudian pilih Buat kebijakan.
6. Dalam panel navigasi yang ada di sisi kiri, pilih Peran dan kemudian pilih Buat peran.
7. Dalam daftar layanan, pilih Layanan Kontainer Elastis.
8. Pada bagian bawah halaman, pilih Tugas Layanan Kontainer Elastis dan kemudian pilih Berikutnya: Izin.
9. Dalam daftar kebijakan, cari AWSDistroOpenTelemetryPolicy.
10. Pilih kotak centang di sebelah AWSDistroOpenTelemetryPolicy.
11. Pilih Berikutnya: Tanda dan kemudian pilih Berikutnya: Tinjau.
12. Untuk Nama peran masukkan **`AWSOpenTelemetryTaskRole`** kemudian pilih Buat peran.

Langkah 2: Membuat sebuah peran eksekusi tugas

Langkah selanjutnya adalah membuat peran eksekusi tugas untuk AWS OpenTelemetry Kolektor.

Untuk membuat peran eksekusi tugas untuk AWS Distro untuk OpenTelemetry

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Dalam panel navigasi yang ada di sisi kiri, pilih Peran dan kemudian pilih Buat peran.
3. Dalam daftar layanan, pilih Layanan Kontainer Elastis.
4. Pada bagian bawah halaman, pilih Tugas Layanan Kontainer Elastis dan kemudian pilih Berikutnya: Izin.
5. Dalam daftar kebijakan, cari AmazonECS lalu pilih kotak centang di TaskExecutionRolePolicy samping AmazonECS. TaskExecutionRolePolicy
6. Dalam daftar kebijakan, cari, CloudWatchLogsFullAccesslalu pilih kotak centang di sebelahnya CloudWatchLogsFullAccess.
7. Dalam daftar kebijakan, cari AmazonSSM ReadOnlyAccess lalu pilih kotak centang di samping AmazonSSM. ReadOnlyAccess
8. Pilih Berikutnya: Tanda dan kemudian pilih Berikutnya: Tinjau.
9. Untuk Nama peran masukkan **`AWSOpenTelemetryTaskExecutionRole`** kemudian pilih Buat peran.

Langkah 3: Membuat sebuah penetapan tugas

Langkah selanjutnya adalah membuat sebuah penetapan tugas.

Untuk membuat definisi tugas untuk AWS Distro untuk OpenTelemetry

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi, silakan pilih Penetapan tugas.
3. Pilih Buat penetapan tugas baru, Buat penetapan tugas baru.
4. Untuk Keluarga penetapan tugas, tentukan nama unik untuk penetapan tugas tersebut.
5. Mengonfigurasikan kontainer-kontainer Anda, lalu pilih Berikutnya.
6. Pada Metrik dan pencatatan log, pilih Gunakan koleksi metrik.
7. Pilih Berikutnya.
8. Pilih Buat.

Untuk informasi selengkapnya tentang penggunaan AWS OpenTelemetry kolektor dengan Amazon ECS, lihat Menyiapkan AWS Distro untuk OpenTelemetry Kolektor di Amazon Elastic Container Service.

Langkah 4: Mengeksekusi tugas

Langkah terakhir adalah mengeksekusi tugas yang telah Anda buat.

Untuk menjalankan tugas untuk AWS Distro untuk OpenTelemetry

1. Buka konsol di https://console.aws.amazon.com/ecs/v2.
2. Pada panel navigasi yang ada di sebelah kiri, pilih Penetapan Tugas dan kemudian pilih tugas yang baru Anda buat.
3. Pilih Tindakan, Terapkan, Eksekusi tugas.
4. Pilih Terapkan, Eksekusi tugas.
5. Pada bagian Opsi komputasi, dari Klaster yang ada, pilih klaster.
6. Pilih Buat.
7. Selanjutnya, Anda dapat memeriksa metrik baru di CloudWatch konsol.
8. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
9. Di panel navigasi, pilih Metrik.

   Anda akan melihat ECS/namespace ContainerInsights. Pilih namespace tersebut dan Anda akan melihat ada delapan metrik.

Menerapkan CloudWatch agen untuk mengumpulkan metrik tingkat instans EC2 di Amazon ECS

Untuk menerapkan CloudWatch agen untuk mengumpulkan metrik tingkat instans dari kluster Amazon ECS yang dihosting di instans EC2, gunakan penyiapan mulai cepat dengan konfigurasi default, atau instal agen secara manual agar dapat menyesuaikannya.

Kedua metode tersebut mengharuskan Anda sudah memiliki setidaknya satu kluster Amazon ECS yang disebarkan dengan tipe peluncuran EC2 dan kontiner CloudWatch agen memiliki akses ke Layanan Metadata Instans Amazon EC2 (IMDS). Untuk informasi selengkapnya tentang IMDS, silakan lihat Metadata dan data pengguna instans.

Metode ini juga mengasumsikan bahwa Anda telah AWS CLI menginstal. Selain itu, untuk menjalankan perintah dalam prosedur berikut, Anda harus masuk ke akun atau peran yang memiliki kebijakan IAM FullAccess dan FullAccessAmazonECS_.

Topik

Pengaturan cepat menggunakan AWS CloudFormation

Untuk menggunakan pengaturan cepat, masukkan perintah berikut untuk digunakan AWS CloudFormation untuk menginstal agen. Dan kemudian ganti *cluster-name* dan *cluster-region* dengan nama dan Wilayah klaster Amazon ECS Anda.

Perintah ini menciptakan peran IAM CWagentecs dan CWagentecs TaskRole. ExecutionRole Jika peran-peran ini sudah ada di akun Anda, gunakan `ParameterKey=CreateIAMRoles,ParameterValue=False` alih-alih `ParameterKey=CreateIAMRoles,ParameterValue=True` saat Anda memasukkan perintah. Jika tidak, maka perintah akan gagal.

```
ClusterName=cluster-name
Region=cluster-region
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/daemon-service/cwagent-
ecs-instance-metric/cloudformation-quickstart/cwagent-ecs-instance-metric-cfn.json
aws cloudformation create-stack --stack-name CWAgentECS-${ClusterName}-${Region} \
    --template-body file://cwagent-ecs-instance-metric-cfn.json \
    --parameters ParameterKey=ClusterName,ParameterValue=${ClusterName} \
                 ParameterKey=CreateIAMRoles,ParameterValue=True \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${Region}
```

(Alternatif) Menggunakan peran IAM Anda sendiri

Jika Anda ingin menggunakan peran tugas ECS kustom Anda sendiri dan peran eksekusi tugas ECS alih-alih peran CWAgenTECS TaskRole dan CWAgenTECS, pertama-tama pastikan bahwa peran yang akan digunakan sebagai ExecutionRole peran tugas ECS telah dilampirkan. CloudWatchAgentServerPolicy Juga, pastikan bahwa peran yang akan digunakan sebagai peran eksekusi tugas ECS memiliki kebijakan CloudWatchAgentServerPolicydan AmazonECS TaskExecutionRolePolicy terlampir. Masukkan perintah berikut ini. Dalam perintah, ganti *task-role-arn*dengan ARN peran tugas ECS kustom Anda, dan ganti dengan ARN peran *execution-role-arn*eksekusi tugas ECS kustom Anda.

```
ClusterName=cluster-name
Region=cluster-region
TaskRoleArn=task-role-arn
ExecutionRoleArn=execution-role-arn
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/daemon-service/cwagent-
ecs-instance-metric/cloudformation-quickstart/cwagent-ecs-instance-metric-cfn.json
aws cloudformation create-stack --stack-name CWAgentECS-${ClusterName}-${Region} \
    --template-body file://cwagent-ecs-instance-metric-cfn.json \
    --parameters ParameterKey=ClusterName,ParameterValue=${ClusterName} \
                 ParameterKey=TaskRoleArn,ParameterValue=${TaskRoleArn} \
                 ParameterKey=ExecutionRoleArn,ParameterValue=${ExecutionRoleArn} \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${Region}
```

Pemecahan masalah pengaturan cepat

Untuk memeriksa status AWS CloudFormation tumpukan, masukkan perintah berikut.

```
ClusterName=cluster-name
Region=cluster-region
aws cloudformation describe-stacks --stack-name CWAgentECS-$ClusterName-$Region --
region $Region
```

Jika Anda melihat status dari `StackStatus` bukan `CREATE_COMPLETE` atau `CREATE_IN_PROGRESS`, maka Anda harus memeriksa peristiwa tumpukan untuk menemukan kesalahan. Masukkan perintah berikut.

```
ClusterName=cluster-name
Region=cluster-region
aws cloudformation describe-stack-events --stack-name CWAgentECS-$ClusterName-$Region
 --region $Region
```

Untuk memeriksa status dari layanan daemon `cwagent`, Anda harus memasukkan perintah berikut. Dalam outputnya, Anda akan melihat bahwa `runningCount` sama dengan `desiredCount` dalam bagian `deployment`. Jika tidak sama, periksa bagian `failures` yang ada di output tersebut.

```
ClusterName=cluster-name
Region=cluster-region
aws ecs describe-services --services cwagent-daemon-service --cluster $ClusterName --
region $Region
```

Anda juga dapat menggunakan konsol CloudWatch Log untuk memeriksa log agen. Cari grup log / ecs/ ecs-cwagent-daemon-service.

Menghapus AWS CloudFormation tumpukan untuk agen CloudWatch

Jika Anda perlu menghapus AWS CloudFormation tumpukan, masukkan perintah berikut.

```
ClusterName=cluster-name
Region=cluster-region
aws cloudformation delete-stack --stack-name CWAgentECS-${ClusterName}-${Region} --
region ${Region}
```

Konfigurasi manual dan kustom

Ikuti langkah-langkah di bagian ini untuk menerapkan CloudWatch agen secara manual guna mengumpulkan metrik tingkat instans dari kluster Amazon ECS yang dihosting di instans EC2.

Peran IAM dan kebijakan yang diperlukan

Ada dua peran IAM yang diperlukan. Anda harus membuat dua peran ini jika belum ada. Untuk informasi selengkapnya tentang peran-peran ini, silakan lihat Peran IAM untuk Tugas dan Peran Eksekusi Tugas Amazon ECS.

- Peran tugas ECS, yang digunakan oleh CloudWatch agen untuk mempublikasikan metrik. Jika peran ini sudah ada, maka Anda harus memastikan bahwa peran tersebut melampirkan kebijakan `CloudWatchAgentServerPolicy`.
- Peran eksekusi tugas ECS, yang digunakan oleh agen Amazon ECS untuk meluncurkan agen. CloudWatch Jika peran ini sudah ada, maka Anda harus memastikan bahwa peran tersebut melampirkan kebijakan `AmazonECSTaskExecutionRolePolicy` dan `CloudWatchAgentServerPolicy`.

Jika Anda belum memiliki peran-peran ini, maka Anda dapat menggunakan perintah-perintah berikut untuk membuat dan melampirkan kebijakan yang diperlukan. Perintah pertama ini untuk membuat peran tugas ECS.

```
aws iam create-role --role-name CWAgentECSTaskRole \
    --assume-role-policy-document "{\"Version\": \"2012-10-17\",\"Statement\": [{\"Sid
\": \"\",\"Effect\": \"Allow\",\"Principal\": {\"Service\": \"ecs-tasks.amazonaws.com
\"},\"Action\": \"sts:AssumeRole\"}]}"
```

Setelah Anda memasukkan perintah sebelumnya, perhatikan nilai `Arn` dari output perintah sebagai "TaskRoleArn”. Anda perlu menggunakannya nanti saat membuat definisi tugas. Kemudian masukkan perintah berikut untuk melampirkan kebijakan-kebijakan yang diperlukan.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
CloudWatchAgentServerPolicy \
    --role-name CWAgentECSTaskRole
```

Perintah berikutnya untuk membuat peran eksekusi tugas ECS.

```
aws iam create-role --role-name CWAgentECSExecutionRole \
    --assume-role-policy-document "{\"Version\": \"2012-10-17\",\"Statement\": [{\"Sid
\": \"\",\"Effect\": \"Allow\",\"Principal\": {\"Service\": \"ecs-tasks.amazonaws.com
\"},\"Action\": \"sts:AssumeRole\"}]}"
```

Setelah Anda memasukkan perintah sebelumnya, perhatikan nilai `Arn` dari output perintah sebagai "ExecutionRoleArn”. Anda harus menggunakannya nanti saat Anda membuat penetapan tugas. Kemudian masukkan perintah-perintah berikut untuk melampirkan kebijakan-kebijakan yang diperlukan.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
CloudWatchAgentServerPolicy \
    --role-name CWAgentECSExecutionRole

aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy \
    --role-name CWAgentECSExecutionRole
```

Membuat penetapan tugas dan meluncurkan layanan daemon

Buat definisi tugas dan gunakan untuk meluncurkan CloudWatch agen sebagai layanan daemon. Untuk membuat penetapan tugas, Anda harus memasukkan perintah berikut. Di baris pertama, ganti placeholder dengan nilai aktual untuk penerapan Anda. *`logs-region adalah Wilayah`* tempat CloudWatch Log berada, dan *`cluster-region adalah Wilayah`* tempat klaster Anda berada. *`task-role-arn`*adalah Arn dari peran tugas ECS yang Anda gunakan, dan *`execution-role-arn`*merupakan Arn dari peran eksekusi tugas ECS.

```
TaskRoleArn=task-role-arn
ExecutionRoleArn=execution-role-arn
AWSLogsRegion=logs-region
```

```
Region=cluster-region
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/daemon-service/cwagent-
ecs-instance-metric/cwagent-ecs-instance-metric.json \
    | sed "s|{{task-role-arn}}|${TaskRoleArn}|;s|{{execution-role-arn}}|
${ExecutionRoleArn}|;s|{{awslogs-region}}|${AWSLogsRegion}|" \
    | xargs -0 aws ecs register-task-definition --region ${Region} --cli-input-json
```

Dan kemudian, eksekusi perintah berikut ini untuk meluncurkan layanan daemon. Dan kemudian ganti *cluster-name* dan *cluster-region* dengan nama dan Wilayah klaster Amazon ECS Anda.

> **Important**
>
> Hapus semua strategi penyedia kapasitas sebelum Anda menjalankan perintah ini. Jika tidak, perintah tidak akan berfungsi.

```
ClusterName=cluster-name
Region=cluster-region
aws ecs create-service \
    --cluster ${ClusterName} \
    --service-name cwagent-daemon-service \
    --task-definition ecs-cwagent-daemon-service \
    --scheduling-strategy DAEMON \
    --region ${Region}
```

Jika Anda melihat pesan kesalahan ini, `An error occurred (InvalidParameterException) when calling the CreateService operation: Creation of service was not idempotent`, maka Anda telah membuat sebuah layanan daemon bernama `cwagent-daemon-service`. Anda harus menghapus layanan yang pertama dengan menggunakan perintah berikut sebagai contohnya.

```
ClusterName=cluster-name
Region=cluster-region
aws ecs delete-service \
    --cluster ${ClusterName} \
    --service cwagent-daemon-service \
    --region ${Region} \
    --force
```

(Opsional) Konfigurasi lanjutan

Secara opsional, Anda dapat menggunakan SSM untuk menentukan opsi konfigurasi lain untuk CloudWatch agen di kluster Amazon ECS yang dihosting di instans EC2. Opsi-opsi ini adalah sebagai berikut:

- `metrics_collection_interval`— Seberapa sering dalam hitungan detik CloudWatch agen mengumpulkan metrik. Bawaannya adalah 60. Rentangnya adalah 1–172.000.
- `endpoint_override` – (Opsional) Menentukan sebuah titik akhir berbeda yang akan menerima kiriman log. Anda mungkin ingin melakukan hal ini jika Anda melakukan penerbitan dari sebuah klaster di VPC dan Anda ingin data log masuk ke sebuah titik akhir VPC.

  Nilai dari `endpoint_override` harus berupa string yang berupa URL.
- `force_flush_interval` – Menentukan lamanya waktu maksimum log tetap berada dalam buffer memori sebelum dikirim ke server, dalam satuan detik. Apa pun pengaturan yang Anda tetapkan untuk bidang ini, jika ukuran log dalam penyangga mencapai 1 MB, log akan segera dikirim ke server saat itu juga. Nilai bawaannya adalah 5 detik.
- `region` – Secara bawaan, agen menerbitkan metrik–metrik ke Wilayah yang sama di mana instans kontainer Amazon ECS berada. Untuk melakukan penggantian atas ini, Anda dapat menentukan Wilayah yang berbeda di sini. Sebagai contoh, `"region" : "us-east-1"`.

Berikut ini adalah sebuah contoh konfigurasi yang disesuaikan:

```
{
    "agent": {
        "region": "us-east-1"
    },
    "logs": {
        "metrics_collected": {
            "ecs": {
                "metrics_collection_interval": 30
            }
        },
        "force_flush_interval": 5
    }
}
```

Untuk menyesuaikan konfigurasi CloudWatch agen Anda di wadah Amazon ECS Anda

1. Pastikan bahwa ReadOnlyAccess kebijakan AmazonSSM dilampirkan ke peran Eksekusi Tugas Amazon ECS Anda. Untuk melakukan hal itu, Anda harus memasukkan perintah berikut. Contoh ini mengasumsikan bahwa peran Eksekusi Tugas Amazon ECS Anda adalah CWAGENTECS. ExecutionRole Jika Anda menggunakan peran yang berbeda, maka Anda harus mengganti nama peran itu dengan perintah berikut.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonSSMReadOnlyAccess \
        --role-name CWAgentECSExecutionRole
```

2. Buat file konfigurasi kustom yang serupa dengan contoh sebelumnya. Beri nama file ini dengan nama `/tmp/ecs-cwagent-daemon-config.json`.

3. Eksekusi perintah berikut untuk memasukkan konfigurasi ini ke dalam Parameter Store. Ganti *`cluster-region`* dengan Wilayah dari klaster Amazon ECS Anda. Untuk menjalankan perintah ini, Anda harus login ke pengguna atau peran yang memiliki kebijakan FullAccessAmazonSSM.

```
Region=cluster-region
aws ssm put-parameter \
    --name "ecs-cwagent-daemon-service" \
    --type "String" \
    --value "`cat /tmp/ecs-cwagent-daemon-config.json`" \
    --region $Region
```

4. Unduh file penetapan tugas ke file lokal, misalnya `/tmp/cwagent-ecs-instance-metric.json`

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/daemon-service/
cwagent-ecs-instance-metric/cwagent-ecs-instance-metric.json -o /tmp/cwagent-ecs-
instance-metric.json
```

5. Ubah file penetapan tugas. Hapus bagian berikut:

```
"environment": [
                {
                    "name": "USE_DEFAULT_CONFIG",
                    "value": "True"
                }
```

```
            ],
```

Ganti bagian tersebut dengan hal berikut ini:

```
"secrets": [
                {
                    "name": "CW_CONFIG_CONTENT",
                    "valueFrom": "ecs-cwagent-daemon-service"
                }
            ],
```

6. Mulai ulang agen sebagai sebuah layanan daemon dengan mengikuti langkah-langkah berikut:

   a. Jalankan perintah berikut.

   ```
   TaskRoleArn=task-role-arn
   ExecutionRoleArn=execution-role-arn
   AWSLogsRegion=logs-region
   Region=cluster-region
   cat /tmp/cwagent-ecs-instance-metric.json \
       | sed "s|{{task-role-arn}}|${TaskRoleArn}|;s|{{execution-role-arn}}|
   ${ExecutionRoleArn}|;s|{{awslogs-region}}|${AWSLogsRegion}|" \
       | xargs -0 aws ecs register-task-definition --region ${Region} --cli-input-
   json
   ```

   b. Jalankan perintah berikut ini untuk meluncurkan layanan daemon. Dan kemudian ganti *cluster-name* dan *cluster-region* dengan nama dan Wilayah klaster Amazon ECS Anda.

   ```
   ClusterName=cluster-name
   Region=cluster-region
   aws ecs create-service \
       --cluster ${ClusterName} \
       --service-name cwagent-daemon-service \
       --task-definition ecs-cwagent-daemon-service \
       --scheduling-strategy DAEMON \
       --region ${Region}
   ```

   Jika Anda melihat pesan kesalahan ini, `An error occurred (InvalidParameterException) when calling the CreateService operation: Creation of service was not idempotent`, maka Anda telah

membuat sebuah layanan daemon bernama `cwagent-daemon-service`. Anda harus menghapus layanan yang pertama dengan menggunakan perintah berikut sebagai contohnya.

```
ClusterName=cluster-name
Region=Region
aws ecs delete-service \
    --cluster ${ClusterName} \
    --service cwagent-daemon-service \
    --region ${Region} \
    --force
```

Menerapkan AWS Distro OpenTelemetry untuk mengumpulkan metrik tingkat instans EC2 di kluster Amazon ECS

Gunakan langkah-langkah di bagian ini untuk menggunakan AWS Distro OpenTelemetry untuk mengumpulkan metrik tingkat instans EC2 di klaster Amazon ECS. Untuk informasi selengkapnya tentang AWS Distro OpenTelemetry, lihat AWS Distro untuk. OpenTelemetry

Langkah-langkah ini mengasumsikan Anda sudah memiliki sebuah klaster yang menjalankan Amazon ECS. Klaster ini harus di-deploy dengan tipe peluncuran EC2. Untuk informasi selengkapnya tentang penggunaan AWS Distro untuk Open Telemetry dengan Amazon ECS dan menyiapkan klaster Amazon ECS untuk tujuan ini, lihat Menyiapkan AWS Distro untuk OpenTelemetry Kolektor di Amazon Elastic Container Service untuk metrik tingkat instans ECS EC2.

Topik

- Pengaturan cepat menggunakan AWS CloudFormation
- Konfigurasi manual dan kustom

Pengaturan cepat menggunakan AWS CloudFormation

Unduh file AWS CloudFormation template untuk menginstal AWS Distro untuk OpenTelemetry kolektor untuk Amazon ECS di EC2. Jalankan perintah curl berikut.

```
curl -O https://raw.githubusercontent.com/aws-observability/aws-otel-collector/main/
deployment-template/ecs/aws-otel-ec2-instance-metrics-daemon-deployment-cfn.yaml
```

Setelah Anda mengunduh file template, buka dan ganti *PATH_TO_ CloudFormation _TEMPLATE* dengan jalur tempat Anda menyimpan file template. Kemudian ekspor parameter berikut dan jalankan AWS CloudFormation perintah, seperti yang ditunjukkan pada perintah berikut.

- Cluster_Name– Nama klaster Amazon ECS
- AWS_Region– Wilayah tempat data nanti akan dikirim
- PATH_TO_ CloudFormation _TEMPLATE - Jalur tempat Anda menyimpan file template. AWS CloudFormation
- perintah — Untuk mengaktifkan AWS Distro for OpenTelemetry collector untuk mengumpulkan metrik tingkat instans untuk Amazon ECS di Amazon EC2, Anda harus menentukan parameter ini. `--config=/etc/ecs/otel-instance-metrics-config.yaml`

```
ClusterName=Cluster_Name
Region=AWS_Region
command=--config=/etc/ecs/otel-instance-metrics-config.yaml
aws cloudformation create-stack --stack-name AOCECS-${ClusterName}-${Region} \
--template-body file://PATH_TO_CloudFormation_TEMPLATE \
--parameters ParameterKey=ClusterName,ParameterValue=${ClusterName} \
ParameterKey=CreateIAMRoles,ParameterValue=True \
ParameterKey=command,ParameterValue=${command} \
--capabilities CAPABILITY_NAMED_IAM \
--region ${Region}
```

Setelah menjalankan perintah ini, Anda harus menggunakan konsol Amazon ECS untuk melihat apakah tugas sedang berjalan, atau tidak.

Pemecahan masalah pengaturan cepat

Untuk memeriksa status AWS CloudFormation tumpukan, masukkan perintah berikut.

```
ClusterName=cluster-name
Region=cluster-region
aws cloudformation describe-stack --stack-name AOCECS-$ClusterName-$Region --region
 $Region
```

Jika nilai dari `StackStatus` bukan `CREATE_COMPLETE` atau `CREATE_IN_PROGRESS`, maka Anda harus memeriksa peristiwa-peristiwa tumpukan untuk menemukan kesalahannya. Masukkan perintah berikut.

```
ClusterName=cluster-name
Region=cluster-region
aws cloudformation describe-stack-events --stack-name AOCECS-$ClusterName-$Region --
region $Region
```

Untuk memeriksa status dari layanan daemon `AOCECS`, Anda harus memasukkan perintah berikut. Dalam outputnya, Anda akan melihat bahwa `runningCount` sama dengan `desiredCount` di bagian deployment. Jika tidak sama, periksa bagian kegagalan yang ada di output tersebut.

```
ClusterName=cluster-name
Region=cluster-region
aws ecs describe-services --services AOCECS-daemon-service --cluster $ClusterName --
region $Region
```

Anda juga dapat menggunakan konsol CloudWatch Log untuk memeriksa log agen. Cari grup log / aws/ecs/containerinsights/ {} /performance. ClusterName

Konfigurasi manual dan kustom

Ikuti langkah-langkah di bagian ini untuk menerapkan AWS Distro secara manual OpenTelemetry untuk mengumpulkan metrik tingkat instans dari kluster Amazon ECS yang dihosting di instans Amazon EC2.

Langkah 1: Peran dan kebijakan yang diperlukan

Ada dua peran IAM yang diperlukan. Anda harus membuat dua peran ini jika belum ada. Untuk informasi selengkapnya tentang peran-peran ini, silakan lihat Membuat kebijakan IAM dan Membuat peran IAM.

Langkah 2: Membuat penetapan tugas

Buat definisi tugas dan gunakan untuk meluncurkan AWS Distro OpenTelemetry sebagai layanan daemon.

Untuk menggunakan template definisi tugas untuk membuat definisi tugas, ikuti petunjuk di Buat Definisi Tugas EC2 ECS untuk instans EC2 dengan AWS OTel Collector.

Untuk menggunakan konsol Amazon ECS untuk membuat definisi tugas, ikuti petunjuk di Instal AWS OTel Collector dengan membuat Definisi Tugas melalui AWS konsol untuk metrik instans Amazon ECS EC2.

Langkah 3: Meluncurkan layanan daemon

Untuk meluncurkan layanan AWS Distro for OpenTelemetry as a daemon, ikuti petunjuk di Jalankan tugas Anda di Amazon Elastic Container Service (Amazon ECS) Container Service (Amazon ECS) menggunakan layanan daemon.

(Opsional) Konfigurasi lanjutan

Secara opsional, Anda dapat menggunakan SSM untuk menentukan opsi konfigurasi lain untuk AWS Distro untuk di kluster Amazon ECS OpenTelemetry Anda yang dihosting di instans Amazon EC2. Untuk informasi selengkapnya, tentang membuat file konfigurasi, lihat OpenTelemetry Konfigurasi Kustom. Untuk informasi selengkapnya tentang opsi-opsi yang dapat Anda gunakan dalam file konfigurasi tersebut, silakan lihat Penerima Wawasan Kontainer AWS.

Siapkan FireLens untuk mengirim log ke CloudWatch Log

FireLens untuk Amazon ECS memungkinkan Anda menggunakan parameter definisi tugas untuk merutekan log ke Amazon CloudWatch Logs untuk penyimpanan log dan analitik. FireLens bekerja dengan Fluent Bit dan Fluentd. Kami menyediakan gambar AWS Fluent Bit, atau Anda dapat menggunakan gambar Fluent Bit atau Fluentd Anda sendiri. Membuat definisi tugas Amazon ECS dengan FireLens konfigurasi didukung menggunakan AWS SDK, AWS CLI, dan. AWS Management Console Untuk informasi selengkapnya tentang CloudWatch Log, lihat Apa itu CloudWatch Log? .

Ada pertimbangan utama saat menggunakan FireLens untuk Amazon ECS. Untuk informasi selengkapnya, silakan lihat Pertimbangan.

Untuk menemukan gambar Fluent Bit, lihat Menggunakan gambar AWS for Fluent Bit. AWS

Untuk membuat definisi tugas yang menggunakan FireLens konfigurasi, lihat Membuat definisi tugas yang menggunakan FireLens konfigurasi.

Contoh

Contoh definisi tugas berikut menunjukkan cara menentukan konfigurasi log yang meneruskan log ke grup log Log. CloudWatch Untuk informasi selengkapnya, lihat Apa itu Amazon CloudWatch Logs? di Panduan Pengguna CloudWatch Log Amazon.

Dalam opsi-opsi konfigurasi log, Anda harus menentukan nama grup log dan Wilayah tempatnya berada. Untuk membuat Fluent Bit menciptakan grup log atas nama Anda, tentukan `"auto_create_group":"true"`. Anda juga dapat menentukan ID tugas sebagai awalan log

stream, yang membantu dalam penyaringan. Untuk informasi selengkapnya, lihat Plugin Bit Lancar untuk CloudWatch Log.

```
{
 "family": "firelens-example-cloudwatch",
 "taskRoleArn": "arn:aws:iam::123456789012:role/ecs_task_iam_role",
 "containerDefinitions": [
  {
   "essential": true,
   "image": "906394416424.dkr.ecr.us-west-2.amazonaws.com/aws-for-fluent-bit:latest",
   "name": "log_router",
   "firelensConfiguration": {
    "type": "fluentbit"
   },
   "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
     "awslogs-group": "firelens-container",
     "awslogs-region": "us-west-2",
     "awslogs-create-group": "true",
     "awslogs-stream-prefix": "firelens"
    }
   },
   "memoryReservation": 50
   },
   {
    "essential": true,
    "image": "nginx",
    "name": "app",
    "logConfiguration": {
     "logDriver":"awsfirelens",
     "options": {
     "Name": "cloudwatch",
     "region": "us-west-2",
     "log_key": "log",
                                "log_group_name": "/aws/ecs/containerinsights/
$(ecs_cluster)/application",
     "auto_create_group": "true",
     "log_stream_name": "$(ecs_task_id)"
    }
   },
   "memoryReservation": 100
  }
```

```
 ]
}
```

Menyiapkan Wawasan Kontainer di Amazon EKS dan Kubernetes

Wawasan Kontainer didukung di Amazon EKS versi 1.23 dan versi yang lebih baru. Metode instalasi mulai cepat hanya didukung pada versi 1.24 dan versi yang lebih baru.

Keseluruhan proses untuk menyiapkan Wawasan Kontainer di Amazon EKS atau Kubernetes adalah sebagai berikut:

1. Verifikasi bahwa Anda sudah memiliki prasyarat yang diperlukan.
2. Siapkan add-on Amazon CloudWatch Observability EKS, CloudWatch agen, atau AWS Distro untuk OpenTelemetry di cluster Anda untuk mengirim metrik. CloudWatch

   > **Note**
   >
   > Untuk menggunakan Wawasan Kontainer dengan kemampuan observasi yang ditingkatkan untuk Amazon EKS, Anda harus menggunakan add-on Amazon CloudWatch Observability EKS atau agennya. CloudWatch Untuk informasi selengkapnya tentang versi Wawasan Kontainer ini, silakan lihat Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS.
   > Untuk menggunakan Wawasan Kontainer dengan Fargate, Anda harus AWS menggunakan Distro untuk. OpenTelemetry Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS tidak didukung di Fargate.

   Siapkan Fluent Bit atau Fluentd untuk mengirim log ke Log. CloudWatch (Ini diaktifkan secara default jika Anda menginstal add-on Amazon CloudWatch Observability EKS.)

   Anda dapat melakukan langkah-langkah ini sekaligus sebagai bagian dari pengaturan mulai cepat jika Anda menggunakan CloudWatch agen, atau melakukannya secara terpisah.
3. (Opsional) Menyiapkan pencatatan log bidang kontrol Amazon EKS.
4. (Opsional) Siapkan CloudWatch agen sebagai titik akhir StatSD di cluster untuk mengirim metrik StatSD ke. CloudWatch
5. (Opsional) Mengaktifkan Log Akses App Mesh Envoy.

Dengan Wawasan Kontainer versi asli, metrik-metrik yang dikumpulkan dan log yang diserap sebagai metrik kustom. Dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS, metrik dan log Wawasan Kontainer dikenakan biaya per observasi, bukan dibebankan per metrik yang disimpan atau log yang diserap. Untuk informasi selengkapnya tentang CloudWatch harga, lihat CloudWatch Harga Amazon.

Topik

Memverifikasi prasyarat

Sebelum Anda melakukan instalasi Wawasan Kontainer di Amazon EKS atau Kubernetes, Anda harus memverifikasi prasyarat-prasyarat berikut: Prasyarat ini berlaku apakah Anda menggunakan CloudWatch agen atau AWS Distro untuk menyiapkan Wawasan Kontainer di OpenTelemetry kluster Amazon EKS.

- Anda memiliki sebuah klaster fungsional Amazon EKS atau Kubernetes dengan simpul-simpul yang dilampirkan di salah satu Wilayah yang mendukung Wawasan Kontainer untuk Amazon EKS dan Kubernetes. Untuk melihat daftar Wilayah yang didukung, silakan lihat Wawasan Kontainer.
- Anda memiliki `kubectl` yang terinstal dan berjalan. Untuk informasi selengkapnya, silakan lihat Menginstal `kubectl` dalam Panduan Pengguna Amazon EKS.
- Jika Anda menggunakan Kubernetes berjalan AWS alih-alih menggunakan Amazon EKS, prasyarat berikut juga diperlukan:
  - Anda harus memastikan bahwa klaster Kubernetes Anda telah mengaktifkan kontrol akses berbasis peran (RBAC). Untuk informasi selengkapnya, silakan lihat Menggunakan Otorisasi RBAC dalam Referensi Kubernetes.
  - Kubelet Anda telah mengaktifkan mode otorisasi Webhook. Untuk informasi selengkapnya, silakan lihat Otentikasi/otorisasi Kubelet dalam Referensi Kubernetes.

Anda juga harus memberikan izin IAM untuk mengaktifkan node pekerja Amazon EKS Anda untuk mengirim metrik dan log ke. CloudWatch Ada dua cara untuk melakukan hal ini:

- Lampirkan sebuah kebijakan ke peran IAM simpul pekerja Anda. Hal ini bisa dilakukan baik untuk klaster Amazon EKS dan klaster Kubernetes lainnya.
- Gunakan sebuah peran IAM untuk akun-akun layanan klaster, dan lampirkan kebijakan pada peran ini. Ini hanya berfungsi untuk klaster Amazon EKS.

Opsi pertama memberikan izin CloudWatch untuk seluruh node, sementara menggunakan peran IAM untuk akun layanan hanya memberikan CloudWatch akses ke pod daemonset yang sesuai.

Melampirkan sebuah kebijakan ke peran IAM simpul pekerja Anda

Ikuti langkah-langkah ini jika Anda ingin melampirkan kebijakan pada peran IAM simpul pekerja Anda. Hal ini bisa dilakukan untuk klaster Amazon EKS dan klaster Kubernetes di luar Amazon EKS.

Cara melampirkan kebijakan yang diperlukan ke peran IAM untuk simpul pekerja Anda

1. Buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.
2. Pilih salah satu instans simpul pekerja dan pilih peran IAM dalam deskripsi.
3. Di halaman peran IAM, silakan pilih Lampirkan kebijakan.
4. Dalam daftar kebijakan, pilih kotak centang di sebelah CloudWatchAgentServerPolicy. Jika perlu, Anda bisa menggunakan kotak pencarian untuk menemukan kebijakan ini.
5. Pilih Lampirkan kebijakan.

Jika Anda menjalankan sebuah klaster Kubernetes di luar Amazon EKS, maka Anda mungkin belum memiliki sebuah peran IAM yang dilampirkan pada simpul pekerja Anda. Jika tidak, maka Anda harus terlebih dahulu melampirkan sebuah peran IAM ke instans tersebut dan kemudian menambahkan kebijakan sebagaimana yang dijelaskan dalam langkah sebelumnya. Untuk informasi selengkapnya tentang cara melampirkan sebuah peran, silakan lihat Melampirkan sebuah Peran IAM ke Instans dalam Panduan Pengguna Amazon EC2 untuk Instans Windows.

Jika Anda menjalankan sebuah klaster Kubernetes di luar Amazon EKS dan Anda ingin mengumpulkan ID volume EBS dalam metrik tersebut, maka Anda harus menambahkan kebijakan lain ke peran IAM yang terlampir ke instans. Tambahkan hal berikut sebagai kebijakan selaras. Untuk informasi selengkapnya, silakan lihat Menambahkan dan Menghapus Izin Identitas IAM dalam Panduan Pengguna IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ec2:DescribeVolumes"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

Menggunakan sebuah peran akun layanan IAM

Metode ini hanya digunakan pada klaster Amazon EKS.

Untuk memberikan izin untuk CloudWatch menggunakan peran akun layanan IAM

1. Jika belum, Anda harus mengaktifkan peran IAM untuk akun layanan di klaster Anda. Untuk informasi selengkapnya, silakan lihat Mengaktifkan peran IAM untuk akun layanan di klaster Anda .
2. Jika belum, Anda harus mengonfigurasi akun layanan tersebut untuk menggunakan sebuah peran IAM. Untuk informasi selengkapnya, silakan lihat Mengonfigurasi sebuah akun layanan Kubernetes untuk mengambil peran IAM.

   Saat Anda membuat peran, lampirkan kebijakan CloudWatchAgentServerPolicyIAM ke peran selain kebijakan yang Anda buat untuk peran tersebut. Selain itu, Akun Layanan Kubernetes terkait yang ditautkan ke peran ini harus dibuat di `amazon-cloudwatch` namespace, di mana daemonset CloudWatch dan Fluent Bit akan diterapkan pada langkah-langkah mendatang
3. Jika belum, Anda harus mengaitkan peran IAM dengan sebuah akun layanan di klaster Anda. Untuk informasi selengkapnya, silakan lihat Mengonfigurasi sebuah akun layanan Kubernetes untuk mengambil peran IAM.

Menggunakan CloudWatch agen dengan kemampuan observasi yang ditingkatkan Wawasan Kontainer diaktifkan

Gunakan instruksi di salah satu bagian berikut untuk menyiapkan Container Insights di klaster Amazon EKS atau klaster Kubernetes dengan menggunakan agen. CloudWatch Instruksi mulai cepat hanya bisa digunakan di Amazon EKS versi 1.24 dan versi yang lebih baru.

> **Note**
>
> Anda dapat melakukan instalasi Wawasan Kontainer dengan mengikuti petunjuk-petunjuk yang diuraikan di salah satu bagian berikut. Anda tidak perlu mengikuti ketiga instruksi tersebut semuanya.

Topik

- Instal add-on Amazon CloudWatch Observability EKS
- Pengaturan Mulai Cepat untuk Wawasan Kontainer di Amazon EKS dan Kubernetes
- Siapkan CloudWatch agen untuk mengumpulkan metrik klaster

Instal add-on Amazon CloudWatch Observability EKS

Anda dapat menggunakan add-on Amazon EKS untuk melakukan instalasi Wawasan Kontainer yang memiliki peningkatan observabilitas untuk Amazon EKS. Add-on menginstal CloudWatch agen untuk mengirim metrik infrastruktur dari cluster, menginstal Fluent Bit untuk mengirim log kontainer, dan juga memungkinkan CloudWatch Sinyal Aplikasi untuk mengirim telemetri kinerja aplikasi.

Add-on Amazon EKS tidak bisa digunakan untuk klaster yang menjalankan Kubernetes, bukan Amazon EKS.

Untuk informasi selengkapnya tentang add-on Amazon CloudWatch Observability EKS, lihat. Instal CloudWatch agen dengan menggunakan add-on Amazon CloudWatch Observability EKS

Untuk menginstal add-on Amazon CloudWatch Observability EKS

1. Pertama, siapkan izin yang diperlukan dengan melampirkan kebijakan CloudWatchAgentServerPolicydan AWSXrayWriteOnlyAccessIAM ke node pekerja Anda. Untuk melakukan hal itu, masukkan perintah berikut. Ganti *my-worker-node-role*dengan peran IAM yang digunakan oleh node pekerja Kubernetes Anda.

```
aws iam attach-role-policy \
--role-name my-worker-node-role \
--policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy \
--policy-arn arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess
```

2. Masukkan perintah berikut untuk melakukan instalasi add-on tersebut:

```
aws eks create-addon --cluster-name my-cluster-name --addon-name amazon-cloudwatch-observability
```

Pengaturan Mulai Cepat untuk Wawasan Kontainer di Amazon EKS dan Kubernetes

> **Important**
>
> Jika Anda menginstal Container Insights di kluster Amazon EKS, sebaiknya gunakan add-on Amazon CloudWatch Observability EKS untuk penginstalan, alih-alih menggunakan petunjuk di bagian ini. Selain itu, untuk mengambil jaringan komputasi yang dipercepat, Anda harus menggunakan add-on Amazon CloudWatch Observability EKS. Untuk informasi dan petunjuk selengkapnya, silakan lihat Instal add-on Amazon CloudWatch Observability EKS.

Untuk menyelesaikan penyiapan Wawasan Kontainer, Anda dapat mengikuti instruksi-instruksi mulai cepat yang diuraikan di bagian ini. Jika Anda sedang melakukan instalasi di sebuah klaster Amazon EKS dan Anda menggunakan instruksi yang diuraikan di bagian ini pada atau setelah 6 November 2023, maka Anda melakukan instalasi Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS di klaster.

> **Important**
>
> Sebelum menyelesaikan langkah-langkah yang dijelaskan di bagian ini, Anda harus selesai memverifikasi prasyarat yang diperlukan, termasuk izin IAM. Untuk informasi selengkapnya, lihat Memverifikasi prasyarat.

Atau, Anda dapat mengikuti instruksi-instruksi yang akan diuraikan dalam dua bagian berikut, Siapkan CloudWatch agen untuk mengumpulkan metrik klaster dan Kirim log ke CloudWatch Log. Bagian tersebut memberikan detail konfigurasi lebih lanjut tentang cara kerja CloudWatch agen

dengan Amazon EKS dan Kubernetes, tetapi mengharuskan Anda untuk melakukan lebih banyak langkah instalasi.

Dengan Wawasan Kontainer versi asli, metrik-metrik yang dikumpulkan dan log yang diserap sebagai metrik kustom. Dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS, metrik dan log Wawasan Kontainer dikenakan biaya per observasi, bukan dibebankan per metrik yang disimpan atau log yang diserap. Untuk informasi selengkapnya tentang CloudWatch harga, lihat CloudWatchHarga Amazon.

> **Note**
>
> Amazon sekarang telah meluncurkan Fluent Bit sebagai solusi log bawaan untuk Wawasan Kontainer dengan keuntungan performa yang signifikan. Kami menyarankan agar Anda menggunakan Fluent Bit, bukan Fluentd.

Mulai Cepat dengan CloudWatch agen dan Fluent Bit

Ada dua konfigurasi untuk Fluent Bit, yakni: versi yang sudah dioptimalkan dan versi yang dapat memberikan Anda pengalaman yang lebih mirip dengan FluentD. Konfigurasi Mulai Cepat dengan menggunakan versi yang sudah dioptimalkan. Untuk detail selengkapnya tentang konfigurasi yang kompatibel dengan Fluentd, silakan lihat Siapkan Fluent Bit sebagai a DaemonSet untuk mengirim log ke CloudWatch Log.

Untuk menerapkan Wawasan Kontainer dengan menggunakan mulai cepat, Anda harus memasukkan perintah berikut.

```
ClusterName=<my-cluster-name>
RegionName=<my-cluster-region>
FluentBitHttpPort='2020'
FluentBitReadFromHead='Off'
[[ ${FluentBitReadFromHead} = 'On' ]] && FluentBitReadFromTail='Off'||
 FluentBitReadFromTail='On'
[[ -z ${FluentBitHttpPort} ]] && FluentBitHttpServer='Off' || FluentBitHttpServer='On'
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/
container-insights-monitoring/quickstart/cwagent-fluent-bit-quickstart-enhanced.yaml
 | sed 's/{{cluster_name}}/'${ClusterName}'/;s/{{region_name}}/'${RegionName}'/;s/
{{http_server_toggle}}/"'${FluentBitHttpServer}'"/;s/
{{http_server_port}}/"'${FluentBitHttpPort}'"/;s/
```

```
{{read_from_head}}/"'${FluentBitReadFromHead}'"/;s/
{{read_from_tail}}/"'${FluentBitReadFromTail}'"/' | kubectl apply -f -
```

Dalam perintah ini, *my-cluster-name* adalah nama Amazon EKS atau kluster Kubernetes Anda, dan *my-cluster-region* merupakan nama Wilayah tempat log diterbitkan. Kami menyarankan Anda menggunakan Wilayah yang sama di mana klaster Anda digunakan untuk mengurangi biaya transfer data AWS keluar.

Sebagai contoh, untuk menerapkan Wawasan Kontainer pada klaster yang bernama `MyCluster` dan menerbitkan log serta metrik-metrik ke AS Barat (Oregon), masukkan perintah berikut.

```
ClusterName='MyCluster'
LogRegion='us-west-2'
FluentBitHttpPort='2020'
FluentBitReadFromHead='Off'
[[ ${FluentBitReadFromHead} = 'On' ]] && FluentBitReadFromTail='Off'||
 FluentBitReadFromTail='On'
[[ -z ${FluentBitHttpPort} ]] && FluentBitHttpServer='Off' || FluentBitHttpServer='On'
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/k8s-deployment-manifest-templates/deployment-
mode/daemonset/container-insights-monitoring/quickstart/cwagent-fluent-
bit-quickstart-enhanced.yaml | sed 's/{{cluster_name}}/'${ClusterName}'/;s/
{{region_name}}/'${LogRegion}'/;s/{{http_server_toggle}}/"'${FluentBitHttpServer}'"/;s/
{{http_server_port}}/"'${FluentBitHttpPort}'"/;s/
{{read_from_head}}/"'${FluentBitReadFromHead}'"/;s/
{{read_from_tail}}/"'${FluentBitReadFromTail}'"/' | kubectl apply -f -
```

Migrasi dari Fluentd

Jika Anda sudah memiliki Fluentd yang sudah dikonfigurasi dan ingin berpindah ke Fluent Bit, maka Anda harus menghapus pod FluentD setelah Anda melakukan instalasi Fluent Bit. Anda dapat menggunakan perintah berikut untuk menghapus FluentD.

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-
insights-monitoring/fluentd/fluentd.yaml | kubectl delete -f -
kubectl delete configmap cluster-info -n amazon-cloudwatch
```

Menghapus Wawasan Kontainer

Jika Anda ingin menghapus Wawasan Kontainer setelah menggunakan pengaturan mulai cepat, maka Anda harus memasukkan perintah berikut.

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/k8s-deployment-manifest-templates/deployment-
mode/daemonset/container-insights-monitoring/quickstart/cwagent-fluent-
bit-quickstart.yaml | sed 's/{{cluster_name}}/'${ClusterName}'/;s/
{{region_name}}/'${LogRegion}'/;s/{{http_server_toggle}}/"'${FluentBitHttpServer}'"/;s/
{{http_server_port}}/"'${FluentBitHttpPort}'"/;s/
{{read_from_head}}/"'${FluentBitReadFromHead}'"/;s/
{{read_from_tail}}/"'${FluentBitReadFromTail}'"/' | kubectl delete -f -
```

Mulai Cepat dengan CloudWatch agen dan Fluentd

Jika Anda sudah menggunakan Fluentd di klaster Kubernetes Anda dan ingin memperpanjangnya menjadi solusi log untuk Wawasan Kontainer, kami akan menyediakan sebuah konfigurasi Fluentd bagi Anda untuk melakukannya.

> **Warning**
>
> Dukungan Container Insights untuk Fluentd sekarang dalam mode pemeliharaan, yang berarti bahwa tidak AWS akan memberikan pembaruan lebih lanjut untuk Fluentd dan kami berencana untuk menghentikannya dalam waktu dekat. Selain itu, konfigurasi Fluentd saat ini untuk Wawasan Kontainer menggunakan Citra Fluentd `fluent/fluentd-kubernetes-daemonset:v1.10.3-debian-cloudwatch-1.0` versi lama yang tidak memiliki patch perbaikan dan keamanan terbaru. Untuk gambar Fluentd terbaru yang didukung oleh komunitas open source, lihat. fluentd-kubernetes-daemonset
> Kami sangat menyarankan Anda bermigrasi untuk digunakan FluentBit dengan Wawasan Kontainer bila memungkinkan. Menggunakan FluentBit sebagai log forwarder untuk Container Insights memberikan peningkatan kinerja yang signifikan.
> Lihat informasi yang lebih lengkap di Siapkan Fluent Bit sebagai a DaemonSet untuk mengirim log ke CloudWatch Log dan Perbedaan-perbedaan jika Anda sudah menggunakan Fluentd.

Untuk menyebarkan CloudWatch agen dan Fluentd menggunakan mulai cepat, gunakan perintah berikut. Penyiapan berikut berisi citra kontainer Fluentd yang didukung komunitas yang didukung di Amazon EKS versi 1.24 dan versi yang lebih baru. Anda dapat mengganti citra tersebut dengan citra Fluentd Anda sendiri selama citra itu memenuhi persyaratan citra FluentD. Untuk informasi selengkapnya, lihat (Opsional) Siapkan Fluentd sebagai DaemonSet untuk mengirim log ke Log CloudWatch .

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-
insights-monitoring/quickstart/cwagent-fluentd-quickstart-enhanced.yaml | sed "s/
{{cluster_name}}/cluster-name/;s/{{region_name}}/cluster-region/" | kubectl apply -f -
```

Dalam perintah ini, *cluster-name* adalah nama klaster dari Amazon EKS atau Kubernetes Anda, dan *cluster-region* adalah nama Wilayah tempat log diterbitkan. Kami menyarankan Anda menggunakan Wilayah yang sama di mana klaster Anda digunakan untuk mengurangi biaya transfer data AWS keluar.

Sebagai contoh, untuk menerapkan Wawasan Kontainer pada klaster yang bernama `MyCluster` dan menerbitkan log serta metrik-metrik ke AS Barat (Oregon), masukkan perintah berikut.

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/
container-insights-monitoring/quickstart/cwagent-fluentd-quickstart.yaml | sed "s/
{{cluster_name}}/MyCluster/;s/{{region_name}}/us-west-2/" | kubectl apply -f -
```

Menghapus Wawasan Kontainer

Jika Anda ingin menghapus Wawasan Kontainer setelah menggunakan pengaturan mulai cepat, maka Anda harus memasukkan perintah berikut.

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/
container-insights-monitoring/quickstart/cwagent-fluentd-quickstart.yaml | sed "s/
{{cluster_name}}/cluster-name/;s/{{region_name}}/cluster-region/" | kubectl delete -f -
```

Siapkan CloudWatch agen untuk mengumpulkan metrik klaster

> **Important**
>
> Jika Anda menginstal Container Insights di kluster Amazon EKS, sebaiknya gunakan add-on Amazon CloudWatch Observability EKS untuk penginstalan, alih-alih menggunakan instruksi di bagian ini. Untuk informasi dan instruksi selengkapnya, silakan lihat Instal add-on Amazon CloudWatch Observability EKS.

Untuk menyiapkan Wawasan Kontainer guna mengumpulkan metrik-metrik, Anda dapat mengikuti langkah-langkah yang diuraikan di Pengaturan Mulai Cepat untuk Wawasan Kontainer di Amazon

EKS dan Kubernetes atau Anda dapat mengikuti langkah-langkah di bagian ini. Pada langkah-langkah berikut, Anda mengatur CloudWatch agen untuk dapat mengumpulkan metrik dari cluster Anda.

Jika Anda sedang melakukan instalasi di sebuah klaster Amazon EKS dan Anda menggunakan instruksi yang diuraikan di bagian ini pada atau setelah 6 November 2023, maka Anda melakukan instalasi Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS di klaster.

Langkah 1: Buat namespace untuk CloudWatch

Gunakan langkah berikut untuk membuat namespace Kubernetes yang dipanggil. `amazon-cloudwatch` CloudWatch Anda dapat melewati langkah ini jika Anda telah membuat namespace ini.

Untuk membuat namespace untuk CloudWatch

- Masukkan perintah berikut.

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/
daemonset/container-insights-monitoring/cloudwatch-namespace.yaml
```

Langkah 2: Membuat sebuah akun layanan di klaster

Gunakan langkah berikut untuk membuat akun layanan untuk CloudWatch agen, jika Anda belum memilikinya.

Untuk membuat akun layanan untuk CloudWatch agen

- Masukkan perintah berikut.

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/
daemonset/container-insights-monitoring/cwagent/cwagent-serviceaccount.yaml
```

Jika Anda tidak mengikuti langkah-langkah sebelumnya, tetapi Anda sudah memiliki akun layanan untuk CloudWatch agen yang ingin Anda gunakan, Anda harus memastikan bahwa ia memiliki aturan berikut. Selain itu, di langkah-langkah selanjutnya dalam instalasi Wawasan Kontainer, Anda juga harus menggunakan nama akun layanan tersebut, bukan `cloudwatch-agent`.

```
rules:
  - apiGroups: [""]
    resources: ["pods", "nodes", "endpoints"]
    verbs: ["watch", "list"]
  - apiGroups: [""]
    resources: ["nodes/proxy"]
    verbs: ["get"]
  - apiGroups: [""]
    resources: ["nodes/stats", "configmaps", “events”]
    verbs: ["create"]
  - apiGroups: [""]
    resources: ["configmaps"]
    resourceNames: ["cwagent-clusterleader"]
    verbs: ["get", "update"]
  - nonResourceURLs: ["/metrics"]
    verbs: ["get"]
```

Langkah 3: Buat ConfigMap untuk CloudWatch agen

Gunakan langkah-langkah berikut untuk membuat ConfigMap untuk CloudWatch agen.

Untuk membuat ConfigMap untuk CloudWatch agen

1. Unduh ConfigMap YAMG ke host `kubectl` klien Anda dengan menjalankan perintah berikut:

   ```
   curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
   insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/
   container-insights-monitoring/cwagent/cwagent-configmap.yaml
   ```

2. Edit file YAML yang sudah Anda unduh, sebagai berikut:

   - cluster_name – Di bagian `kubernetes`, ganti `{{cluster_name}}` dengan nama klaster Anda. Hapus karakter `{{}}`. Atau, jika Anda menggunakan sebuah klaster Amazon EKS, Anda dapat menghapus bidang dan nilai `"cluster_name"`. Jika Anda melakukannya, CloudWatch agen mendeteksi nama cluster dari tag Amazon EC2.

3. (Opsional) Buat perubahan lebih lanjut ConfigMap berdasarkan persyaratan pemantauan Anda, sebagai berikut:

   - metrics_collection_interval – Di bagian `kubernetes`, Anda dapat menentukan seberapa sering agen mengumpulkan metrik. Bawaannya adalah 60 detik. Interval pengumpulan

kadvisor bawaan di kubelet adalah 15 detik, jadi jangan mengatur nilai ini menjadi kurang dari 15 detik.

- endpoint_override - Di `logs` bagian ini, Anda dapat menentukan titik akhir CloudWatch Log jika Anda ingin mengganti titik akhir default. Anda mungkin ingin melakukan ini jika Anda menerbitkan dari sebuah klaster di sebuah VPC dan Anda ingin data masuk ke sebuah titik akhir VPC.
- force_flush_interval — Di `logs` bagian ini, Anda dapat menentukan interval untuk batching peristiwa log sebelum dipublikasikan ke Log. CloudWatch Bawaannya adalah 5 detik.
- region – Secara bawaan, agen menerbitkan metrik–metrik ke Wilayah tempat simpul pekerja berada. Untuk menggantinyaini, Anda dapat menambahkan sebuah bidang `region` di bagian `agent`: misalnya, `"region":"us-west-2"`.
- bagian statsd - Jika Anda ingin agen CloudWatch Log juga berjalan sebagai pendengar StatSD di setiap node pekerja klaster Anda, Anda dapat menambahkan `statsd` bagian ke bagian `metrics` tersebut, seperti pada contoh berikut. Untuk informasi tentang pilihan StatsD lainnya untuk bagian ini, silakan lihat Ambil metrik kustom dengan StatSD .

```
"metrics": {
  "metrics_collected": {
    "statsd": {
      "service_address":":8125"
    }
  }
}
```

Contoh bagian JSON lengkapnya adalah sebagai berikut.

```
{
    "agent": {
        "region": "us-east-1"
    },
    "logs": {
        "metrics_collected": {
            "kubernetes": {
                "cluster_name": "MyCluster",
                "metrics_collection_interval": 60
            }
        },
        "force_flush_interval": 5,
        "endpoint_override": "logs.us-east-1.amazonaws.com"
```

```
    },
    "metrics": {
        "metrics_collected": {
            "statsd": {
                "service_address": ":8125"
            }
        }
    }
}
```

4. Buat ConfigMap di cluster dengan menjalankan perintah berikut.

```
kubectl apply -f cwagent-configmap.yaml
```

Langkah 4: Menyebarkan CloudWatch agen sebagai DaemonSet

Untuk menyelesaikan instalasi CloudWatch agen dan mulai mengumpulkan metrik kontainer, gunakan langkah-langkah berikut.

Untuk menyebarkan CloudWatch agen sebagai DaemonSet

1. • Jika Anda tidak ingin menggunakan StatsD pada klaster tersebut, Anda harus memasukkan perintah berikut.

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-
cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/
deployment-mode/daemonset/container-insights-monitoring/cwagent/cwagent-
daemonset.yaml
```

   • Jika Anda ingin menggunakan StatsD, Anda harus mengikuti langkah-langkah ini:

      a. Unduh DaemonSet YAMG ke host `kubectl` klien Anda dengan menjalankan perintah berikut.

```
curl -O  https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/k8s-deployment-manifest-templates/deployment-
mode/daemonset/container-insights-monitoring/cwagent/cwagent-daemonset.yaml
```

      b. Batalkan komentar di bagian `port` dalam file `cwagent-daemonset.yaml` seperti berikut ini:

```
ports:
  - containerPort: 8125
    hostPort: 8125
    protocol: UDP
```

c. Terapkan CloudWatch agen di cluster Anda dengan menjalankan perintah berikut.

```
kubectl apply -f cwagent-daemonset.yaml
```

2. Lakukan validasi bahwa agen di-deploy dengan menjalankan perintah berikut.

```
kubectl get pods -n amazon-cloudwatch
```

Saat selesai, CloudWatch agen membuat grup log bernama `/aws/containerinsights/`*`Cluster_Name`*`/performance` dan mengirimkan peristiwa log kinerja ke grup log ini. Jika Anda juga menyiapkan agen sebagai sebuah pendengar StatsD, maka agen itu juga akan mendengarkan metrik-metrik StatsD pada port 8125 dengan alamat IP simpul tempat pod aplikasi dijadwalkan.

Pemecahan Masalah

Jika agen tersebut tidak melakukan deployment dengan benar, coba lakukan hal berikut:

- Jalankan perintah berikut untuk mendapatkan daftar pod.

```
kubectl get pods -n amazon-cloudwatch
```

- Jalankan perintah berikut dan periksa peristiwa di bagian bawah output.

```
kubectl describe pod pod-name -n amazon-cloudwatch
```

- Jalankan perintah berikut untuk memeriksa log.

```
kubectl logs pod-name  -n amazon-cloudwatch
```

Menggunakan AWS Distro untuk OpenTelemetry

Anda dapat mengatur Wawasan Kontainer untuk mengumpulkan metrik dari kluster Amazon EKS dengan menggunakan AWS Distro untuk kolektor. OpenTelemetry Untuk informasi selengkapnya tentang AWS Distro OpenTelemetry, lihat AWS Distro untuk. OpenTelemetry

> **Important**
>
> Jika Anda menginstal menggunakan AWS Distro for OpenTelemetry, Anda menginstal Container Insights tetapi tidak mendapatkan Container Insights dengan peningkatan observabilitas untuk Amazon EKS. Anda tidak akan mengumpulkan metrik-metrik terperinci yang didukung di Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS.

Cara Anda menyiapkan Wawasan Kontainer bergantung pada apakah klaster di-host di instans Amazon EC2 atau di AWS Fargate (Fargate).

Klaster-klaster Amazon EKS yang di-host di Amazon EC2

Jika Anda belum melakukan seperti itu, Anda harus memastikan bahwa Anda telah memenuhi prasyaratnya termasuk peran IAM yang diperlukan. Untuk informasi selengkapnya, lihat Memverifikasi prasyarat.

Amazon menyediakan sebuah bagan Helm yang dapat Anda gunakan untuk menyiapkan pemantauan Amazon Elastic Kubernetes Service di Amazon EC2. Pemantauan ini menggunakan AWS Distro for OpenTelemetry (ADOT) Collector untuk metrik dan Fluent Bit untuk log. Oleh karena itu, bagan Helm berguna bagi pelanggan yang menggunakan Amazon EKS di Amazon EC2 dan ingin mengumpulkan metrik dan log untuk dikirim CloudWatch ke Wawasan Kontainer. Untuk informasi selengkapnya tentang bagan Helm ini, lihat bagan Helm ADOT untuk EKS pada metrik EC2 dan log ke Amazon Container Insights. CloudWatch

Atau, Anda juga dapat menggunakan instruksi-instruksi yang diuraikan di bagian ini.

Pertama, gunakan AWS Distro untuk OpenTelemetry kolektor sebagai DaemonSet dengan memasukkan perintah berikut.

```
curl https://raw.githubusercontent.com/aws-observability/aws-otel-collector/main/
deployment-template/eks/otel-container-insights-infra.yaml |
```

```
kubectl apply -f -
```

Untuk mengonfirmasi bahwa kolektor itu sedang berjalan, Anda perlu memasukkan perintah berikut.

```
kubectl get pods -l name=aws-otel-eks-ci -n aws-otel-eks
```

Jika output dari perintah ini menyertakan beberapa pod yang berada dalam status `Running`, artinya kolektor tersebut sedang berjalan dan mengumpulkan metrik dari klaster. Kolektor tersebut membuat sebuah grup log dengan nama `aws/containerinsights/`*`cluster-name`*`/performance` dan mengirimkan peristiwa log performa ke sana.

Untuk informasi tentang cara melihat metrik Wawasan Kontainer CloudWatch, lihat. Menampilkan metrik-metrik Wawasan Kontainer

AWS juga telah menyediakan dokumentasi GitHub untuk skenario ini. Jika Anda ingin melakukan kustom pada metrik dan log yang diterbitkan oleh Wawasan Kontainer, silakan lihat https://aws-otel.github.io/docs/getting-started/container-insights/eks-infra.

Klaster Amazon EKS yang di-host di Fargate

Untuk petunjuk tentang cara mengonfigurasi dan menerapkan Kolektor ADOT guna mengumpulkan metrik sistem dari beban kerja yang diterapkan ke klaster Amazon EKS di Fargate dan mengirimkannya ke Wawasan Kontainer, lihat Wawasan Kontainer EKS Fargate di Distro CloudWatch untuk dokumentasi. AWS OpenTelemetry

Kirim log ke CloudWatch Log

Untuk mengirim log dari kontainer Anda ke Amazon CloudWatch Logs, Anda dapat menggunakan Fluent Bit atau Fluentd. Untuk informasi selengkapnya, silakan lihat Fluent Bit dan Fluentd.

Jika Anda belum menggunakan Fluentd, kami menyarankan Anda untuk menggunakan Fluent Bit karena alasan berikut:

- Fluent Bit memiliki jejak sumber daya yang lebih kecil dan lebih hemat sumber daya dengan penggunaan memori dan CPU dari pada Fluentd. Untuk melihat perbandingannya dengan lebih detail, silakan lihat Perbandingan performa antara Fluent Bit dan Fluentd.
- Gambar Fluent Bit dikembangkan dan dipelihara oleh AWS. Ini memberikan AWS kemampuan untuk mengadopsi fitur gambar Fluent Bit baru dan merespons masalah dengan lebih cepat.

Topik

- Perbandingan performa antara Fluent Bit dan Fluentd
- Siapkan Fluent Bit sebagai a DaemonSet untuk mengirim log ke CloudWatch Log
- (Opsional) Siapkan Fluentd sebagai DaemonSet untuk mengirim log ke Log CloudWatch
- (Opsional) Menyiapkan pencatatan log bidang kontrol Amazon EKS
- (Opsional) Mengaktifkan log akses App Mesh Envoy
- (Opsional) Mengaktifkan fitur Use_Kubelet untuk klaster besar

Perbandingan performa antara Fluent Bit dan Fluentd

Tabel-tabel berikut menunjukkan keunggulan performa yang dimiliki oleh Fluent Bit dibandingkan Fluentd dalam penggunaan memori dan CPU. Angka-angka berikut hanya untuk referensi dan mungkin berubah tergantung pada lingkungan.

Log per detik	Penggunaan CPU Fluentd	Penggunaan CPU Fluent Bit dengan konfigurasi yang kompatibel dengan Fluentd	Penggunaan CPU Fluent Bit dengan konfigurasi yang sudah dioptimalkan
100	0,35 vCPU	0,02 vCPU	0,02 vCPU
1.000	0,32 vCPU	0,14 vCPU	0,11 vCPU
5.000	0,85 vCPU	0,48 vCPU	0,30 vCPU
10.000	0,94 vCPU	0,60 vCPU	0,39 vCPU

Log per detik	Penggunaan memori fluentd	Penggunaan memori Fluent Bit dengan konfigurasi yang kompatibel dengan Fluentd	Penggunaan memori Fluent Bit dengan konfigurasi yang sudah dioptimalkan
100	153 MB	46 MB	37 MB
1.000	270 MB	45 MB	40 MB

Log per detik	Penggunaan memori fluentd	Penggunaan memori Fluent Bit dengan konfigurasi yang kompatibel dengan Fluentd	Penggunaan memori Fluent Bit dengan konfigurasi yang sudah dioptimalkan
5.000	320 MB	55 MB	45 MB
10.000	375 MB	92 MB	75 MB

Siapkan Fluent Bit sebagai a DaemonSet untuk mengirim log ke CloudWatch Log

Bagian berikut membantu Anda menerapkan Fluent Bit untuk mengirim log dari kontainer ke CloudWatch Log.

Topik

- Perbedaan-perbedaan jika Anda sudah menggunakan Fluentd
- Menyiapkan Fluent Bit
- Dukungan log multi-baris
- (Opsional) Mengurangi volume log dari Fluent Bit
- Pemecahan Masalah
- Dasbor

Perbedaan-perbedaan jika Anda sudah menggunakan Fluentd

Jika Anda sudah menggunakan Fluentd untuk mengirim log dari kontainer ke CloudWatch Log, baca bagian ini untuk melihat perbedaan antara Fluentd dan Fluent Bit. Jika Anda belum menggunakan Fluentd dengan Wawasan Kontainer, maka Anda dapat melewatinya dan lihat Menyiapkan Fluent Bit.

Kami menyediakan untuk Anda dua konfigurasi bawaan untuk Fluent Bit:

- Konfigurasi Fluent Bit dioptimalkan— Sebuah konfigurasi yang diselaraskan dengan praktik terbaik Fluent Bit.
- Konfigurasi kompatibel dengan Fluentd— Sebuah konfigurasi yang sebisa mungkin diselaraskan dengan perilaku Fluentd.

Daftar berikut ini akan menjelaskan kepada Anda perbedaan-perbedaan antara Fluentd dan masing-masing konfigurasi Fluent Bit secara detail.

- Perbedaan dalam nama log stream— Jika Anda menggunakan konfigurasi Fluent Bit yang dioptimalkan, maka nama log stream akan berbeda.

  Pada `/aws/containerinsights/Cluster_Name/application`

  - Konfigurasi Fluent Bit yang dioptimalkan akan mengirimkan log ke *`kubernetes-nodeName`*`-application.var.log.containers.`*`kubernetes-podName_kubernetes-namespace_kubernetes-container-name-kubernetes-containerID`*
  - Fluentd mengirimkan log ke *`kubernetes-podName_kubernetes-namespace_kubernetes-containerName_kubernetes-containerID`*

  Pada `/aws/containerinsights/Cluster_Name/host`

  - Konfigurasi Fluent Bit yang dioptimalkan akan mengirimkan log ke *`kubernetes-nodeName`*`.`*`host-log-file`*
  - Fluentd mengirimkan log ke *`host-log-file-Kubernetes-NodePrivateIp`*

  Pada `/aws/containerinsights/Cluster_Name/dataplane`

  - Konfigurasi Fluent Bit yang dioptimalkan akan mengirimkan log ke *`kubernetes-nodeName`*`.`*`dataplaneServiceLog`*
  - Fluentd mengirimkan log ke *`dataplaneServiceLog-Kubernetes-nodeName`*
- File log `kube-proxy` dan `aws-node` yang ditulis oleh Wawasan Kontainer berada di lokasi yang berbeda. Dalam konfigurasi Fluentd, file-file itu berada di `/aws/containerinsights/Cluster_Name/application`. Dalam konfigurasi Fluent Bit yang dioptimalkan, file-file itu berada di `/aws/containerinsights/Cluster_Name/dataplane`.
- Sebagian besar metadata seperti `pod_name` dan `namespace_name` adalah sama, baik di Fluent Bit maupun di Fluentd, tetapi yang berikut ini berbeda.
  - Konfigurasi Fluent Bit yang dioptimalkan menggunakan `docker_id`, sedangkan Fluentd menggunakan `Docker.container_id`.
  - Kedua konfigurasi Fluent Bit tidak menggunakan metadata berikut ini. Mereka hanya ada dalam Fluentd: `container_image_id`, `master_url`, `namespace_id`, dan `namespace_labels`.

Menyiapkan Fluent Bit

Untuk menyiapkan Fluent Bit untuk mengumpulkan log dari kontainer-kontainer Anda, Anda dapat mengikuti langkah-langkah yang diuraikan dalam Pengaturan Mulai Cepat untuk Wawasan Kontainer di Amazon EKS dan Kubernetes atau Anda dapat mengikuti langkah-langkah yang dijelaskan di bagian ini.

Dengan metode mana pun, peran IAM yang dilampirkan ke simpul klaster harus memiliki izin yang memadai. Untuk informasi selengkapnya tentang izin-izin yang diperlukan untuk menjalankan sebuah klaster Amazon EKS, silakan lihat Kebijakan IAM, Peran, dan Izin Amazon EKS dalam Panduan Pengguna Amazon EKS .

Pada langkah-langkah berikut, Anda mengatur Fluent Bit sebagai DaemonSet untuk mengirim log ke Log. CloudWatch Setelah Anda menyelesaikan langkah ini, Fluent Bit akan membuat grup log berikut jika belum ada.

> **Important**
>
> Jika Anda sudah memiliki FluentD yang dikonfigurasi di Container Insights dan DaemonSet fluentD tidak berjalan seperti yang diharapkan (ini dapat terjadi jika Anda `containerd` menggunakan runtime), Anda harus menghapus instalannya sebelum menginstal Fluent Bit untuk mencegah Fluent Bit memproses pesan log kesalahan FluentD. Jika tidak, maka Anda harus menghapus instalasi FluentD segera setelah Anda berhasil melakukan instalasi Fluent Bit. Menghapus instalasi Fluentd setelah menyelesaikan instalasi Fluent Bit akan memastikan keberlanjutan pencatatan log selama proses migrasi ini. Hanya satu dari Fluent Bit atau FluentD yang diperlukan untuk mengirim log ke Log. CloudWatch

Nama grup log	Sumber log
`/aws/containerinsights/` *`Cluster_Name`* `/application`	Semua file log yang ada di `/var/log/containers`
`/aws/containerinsights/` *`Cluster_Name`* `/host`	Log dari `/var/log/dmesg` , `/var/log/secure` , dan `/var/log/messages`

Nama grup log	Sumber log
/aws/containerinsights/ *Cluster_Name* /dataplane	Log yang ada di /var/log/journal untuk kubelet.service , kubeproxy .service , dan docker.service .

Untuk menginstal Fluent Bit untuk mengirim log dari kontainer ke CloudWatch Log

1. Jika Anda belum memiliki sebuah namespace yang mempunyai nama amazon-cloudwatch, maka Anda harus membuatnya dengan memasukkan perintah berikut:

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/
daemonset/container-insights-monitoring/cloudwatch-namespace.yaml
```

2. Jalankan perintah berikut untuk membuat ConfigMap nama cluster-info dengan nama cluster dan Region untuk mengirim log ke. Ganti *cluster-name* dan *cluster-region* dengan nama dan Wilayah klaster Anda.

```
ClusterName=cluster-name
RegionName=cluster-region
FluentBitHttpPort='2020'
FluentBitReadFromHead='Off'
[[ ${FluentBitReadFromHead} = 'On' ]] && FluentBitReadFromTail='Off'||
 FluentBitReadFromTail='On'
[[ -z ${FluentBitHttpPort} ]] && FluentBitHttpServer='Off' ||
 FluentBitHttpServer='On'
kubectl create configmap fluent-bit-cluster-info \
--from-literal=cluster.name=${ClusterName} \
--from-literal=http.server=${FluentBitHttpServer} \
--from-literal=http.port=${FluentBitHttpPort} \
--from-literal=read.head=${FluentBitReadFromHead} \
--from-literal=read.tail=${FluentBitReadFromTail} \
--from-literal=logs.region=${RegionName} -n amazon-cloudwatch
```

   Dalam perintah ini, FluentBitHttpServer untuk memantau metrik plugin akan aktif secara bawaan. Untuk menonaktifkannya, Anda harus mengubah baris ketiga dalam perintah tersebut menjadi FluentBitHttpPort='' (string kosong) dalam perintah tersebut.

Secara bawaan, Fluent Bit juga akan membaca file log dari ekor, dan akan menangkap hanya log baru saja, setelah di-deploy. Jika Anda ingin sebaliknya, maka Anda harus mengatur `FluentBitReadFromHead='On'` dan itu akan mengumpulkan semua log di sistem file.

3. Mengunduh dan menerapkan daemonset Fluent Bit ke klaster dengan menjalankan perintah-perintah berikut.

   - Jika Anda ingin konfigurasi Fluent Bit yang dioptimalkan, maka Anda harus menjalankan perintah ini.

     ```
     kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/fluent-bit/fluent-bit.yaml
     ```

   - Jika Anda ingin konfigurasi Fluent Bit yang lebih mirip dengan Fluentd, maka Anda harus menjalankan perintah ini.

     ```
     kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/fluent-bit/fluent-bit-compatible.yaml
     ```

   > **Important**
   >
   > Konfigurasi daemonset Fluent Bit secara default menyetel level log ke INFO, yang dapat menghasilkan biaya konsumsi CloudWatch Log yang lebih tinggi. Jika Anda ingin mengurangi volume dan biaya penyerapan log, Anda dapat mengubah level log menjadi ERROR.
   > Untuk informasi selengkapnya tentang cara mengurangi volume log ini, silakan lihat (Opsional) Mengurangi volume log dari Fluent Bit

4. Lakukan validasi atas deployment dengan memasukkan perintah berikut. Masing-masing simpul harus memiliki satu pod yang diberi nama fluent-bit-*.

   ```
   kubectl get pods -n amazon-cloudwatch
   ```

Langkah-langkah di atas akan membuat sumber daya berikut di klaster:

- Sebuah akun layanan yang bernama `Fluent-Bit` dalam namespace `amazon-cloudwatch`. Akun layanan ini digunakan untuk menjalankan daemonSet Fluent Bit. Untuk informasi selengkapnya, silakan lihat Mengelola Akun Layanan dalam Referensi Kubernetes.
- Sebuah peran klaster yang diberi nama `Fluent-Bit-role` dalam namespace `amazon-cloudwatch`. Peran klaster ini akan memberikan izin `get`, `list`, dan `watch` pada log pod ke akun layanan `Fluent-Bit`. Untuk informasi selengkapnya, silakan lihat Gambaran umum API di Referensi Kubernetes.
- A ConfigMap bernama `Fluent-Bit-config` di `amazon-cloudwatch` namespace. Ini ConfigMap berisi konfigurasi yang akan digunakan oleh Fluent Bit. Untuk informasi selengkapnya, lihat Mengonfigurasi Pod untuk Digunakan ConfigMap dalam dokumentasi Tugas Kubernetes.

Jika Anda ingin melakukan verifikasi terhadap pengaturan Fluent Bit Anda, ikuti langkah-langkah berikut.

Melakukan verifikasi terhadap pengaturan Fluent Bit

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Grup log.
3. Anda harus memastikan bahwa Anda berada di Wilayah tempat menerapkan Fluent Bit Anda.
4. Memeriksa daftar grup log di Wilayah. Anda akan melihat yang berikut ini:
   - `/aws/containerinsights/Cluster_Name/application`
   - `/aws/containerinsights/Cluster_Name/host`
   - `/aws/containerinsights/Cluster_Name/dataplane`
5. Navigasikan ke salah satu grup log ini dan periksa Waktu Peristiwa Terakhir untuk log stream. Jika ini relatif baru saat Anda menerapkan Fluent Bit, maka penyiapannya diverifikasi.

   Mungkin akan ada sedikit penundaan dalam membuat grup log `/dataplane`. Hal ini normal karena grup log ini hanya bisa dibuat ketika Fluent Bit mulai mengirim log untuk grup log tersebut.

Dukungan log multi-baris

Untuk informasi tentang cara menggunakan Fluent Bit dengan log multi-baris, silakan lihat bagian dokumentasi Fluent Bit berikut ini:

- Penguraian Multi-baris
- Multi-baris dan Kontainer (v1.8)
- Inti Multi-baris (v1.8)
- Selalu gunakan multi-baris di input ekor

(Opsional) Mengurangi volume log dari Fluent Bit

Secara default, kami mengirim log aplikasi Fluent Bit dan metadata Kubernetes ke. CloudWatch Jika Anda ingin mengurangi volume data yang dikirim CloudWatch, Anda dapat menghentikan salah satu atau kedua sumber data ini agar tidak dikirim CloudWatch.

Untuk menghentikan log aplikasi Fluent Bit, Anda harus menghapus bagian berikut dari file `Fluent-Bit.yaml`.

```
[INPUT]
        Name                tail
        Tag                 application.*
        Path                /var/log/containers/fluent-bit*
        Parser              docker
        DB                  /fluent-bit/state/flb_log.db
        Mem_Buf_Limit       5MB
        Skip_Long_Lines     On
        Refresh_Interval    10
```

Untuk menghapus metadata Kubernetes agar tidak ditambahkan ke peristiwa log yang dikirim CloudWatch, tambahkan filter berikut ke bagian dalam file. `application-log.conf` `Fluent-Bit.yaml` Ganti <Metadata_1>dan bidang serupa dengan pengidentifikasi metadata yang sebenarnya.

```
application-log.conf: |
    [FILTER]
        Name                nest
        Match               application.*
        Operation           lift
        Nested_under        kubernetes
        Add_prefix          Kube.

    [FILTER]
        Name                modify
        Match               application.*
```

```
        Remove              Kube.<Metadata_1>
        Remove              Kube.<Metadata_2>
        Remove              Kube.<Metadata_3>

    [FILTER]
        Name                nest
        Match               application.*
        Operation           nest
        Wildcard            Kube.*
        Nested_under        kubernetes
        Remove_prefix       Kube.
```

Pemecahan Masalah

Jika Anda tidak melihat grup log ini dan mencarinya di Wilayah yang benar, Anda perlu memeriksa log untuk pod daemonSet Fluent Bit untuk mencari kesalahan.

Jalankan perintah berikut dan pastikan bahwa statusnya `Running`.

```
kubectl get pods -n amazon-cloudwatch
```

Jika log memiliki kesalahan terkait izin IAM, Anda perlu memeriksa peran IAM yang dilampirkan ke klaster simpul. Untuk informasi selengkapnya tentang izin-izin yang diperlukan untuk menjalankan sebuah klaster Amazon EKS, silakan lihat Kebijakan IAM, Peran, dan Izin Amazon EKS dalam Panduan Pengguna Amazon EKS .

Jika status pod tersebut adalah `CreateContainerConfigError`, dapatkan kesalahan yang tepat dengan menjalankan perintah berikut.

```
kubectl describe pod pod_name -n amazon-cloudwatch
```

Dasbor

Anda dapat membuat sebuah dasbor untuk memantau metrik-metrik dari masing-masing plugin yang sedang berjalan. Anda dapat melihat data untuk input dan output byte dan data untuk tingkat pemrosesan catatan serta kesalahan output dan data untuk tingkat coba lagi/gagal. Untuk melihat metrik ini, Anda harus menginstal CloudWatch agen dengan koleksi metrik Prometheus untuk klaster Amazon EKS dan Kubernetes. Untuk informasi selengkapnya tentang cara menyiapkan dasbor, silakan lihat Instal CloudWatch agen dengan koleksi metrik Prometheus di klaster Amazon EKS dan Kubernetes.

> **Note**
>
> Sebelum Anda dapat menyiapkan dasbor ini, Anda harus menyiapkan Wawasan Kontainer untuk metrik Prometheus. Untuk informasi selengkapnya, lihat Pemantauan metrik-metrik Prometheus Wawasan Kontainer.

Cara membuat sebuah dasbor untuk metrik-metrik Fluent Bit Prometheus

1. Membuat variabel lingkungan, yang mengganti nilai yang ada di sebelah kanan di baris-baris berikut agar sesuai dengan deployment Anda.

```
DASHBOARD_NAME=your_cw_dashboard_name
REGION_NAME=your_metric_region_such_as_us-west-1
CLUSTER_NAME=your_kubernetes_cluster_name
```

2. Membuat dasbor dengan menjalankan perintah berikut.

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-
prometheus/sample_cloudwatch_dashboards/fluent-bit/cw_dashboard_fluent_bit.json \
| sed "s/{{YOUR_AWS_REGION}}/${REGION_NAME}/g" \
| sed "s/{{YOUR_CLUSTER_NAME}}/${CLUSTER_NAME}/g" \
| xargs -0 aws cloudwatch put-dashboard --dashboard-name ${DASHBOARD_NAME} --
dashboard-body
```

(Opsional) Siapkan Fluentd sebagai DaemonSet untuk mengirim log ke Log CloudWatch

> **Warning**
>
> Dukungan Container Insights untuk Fluentd sekarang dalam mode pemeliharaan, yang berarti bahwa tidak AWS akan memberikan pembaruan lebih lanjut untuk Fluentd dan kami berencana untuk menghentikannya dalam waktu dekat. Selain itu, konfigurasi Fluentd saat ini untuk Wawasan Kontainer menggunakan Citra Fluentd `fluent/fluentd-kubernetes-daemonset:v1.10.3-debian-cloudwatch-1.0` versi lama yang tidak memiliki patch perbaikan dan keamanan terbaru. Untuk gambar Fluentd terbaru yang didukung oleh komunitas open source, lihat. fluentd-kubernetes-daemonset

Kami sangat menyarankan Anda bermigrasi untuk digunakan FluentBit dengan Wawasan Kontainer bila memungkinkan. Menggunakan FluentBit sebagai log forwarder untuk Container Insights memberikan peningkatan kinerja yang signifikan.
Untuk informasi selengkapnya, silakan lihat Siapkan Fluent Bit sebagai a DaemonSet untuk mengirim log ke CloudWatch Log dan Perbedaan-perbedaan jika Anda sudah menggunakan Fluentd.

Untuk menyiapkan Fluentd untuk mengumpulkan log dari kontainer-kontainer Anda, Anda dapat mengikuti langkah-langkah yang diuraikan dalam Pengaturan Mulai Cepat untuk Wawasan Kontainer di Amazon EKS dan Kubernetes atau Anda dapat mengikuti langkah-langkah yang dijelaskan di bagian ini. Pada langkah-langkah berikut, Anda mengatur Fluentd sebagai DaemonSet untuk mengirim log ke CloudWatch Log. Setelah Anda menyelesaikan langkah ini, Fluentd akan membuat grup log berikut jika belum ada.

Nama grup log	Sumber log
`/aws/containerinsights/` *`Cluster_Name`* `/application`	Semua file log yang ada di `/var/log/containers`
`/aws/containerinsights/` *`Cluster_Name`* `/host`	Log dari `/var/log/dmesg` , `/var/log/secure` , dan `/var/log/messages`
`/aws/containerinsights/` *`Cluster_Name`* `/dataplane`	Log yang ada di `/var/log/journal` untuk `kubelet.service` , `kubeproxy.service` , dan `docker.service` .

Langkah 1: Buat namespace untuk CloudWatch

Gunakan langkah berikut untuk membuat namespace Kubernetes yang dipanggil. `amazon-cloudwatch` CloudWatch Anda dapat melewati langkah ini jika Anda telah membuat namespace ini.

Untuk membuat namespace untuk CloudWatch

- Masukkan perintah berikut.

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/
daemonset/container-insights-monitoring/cloudwatch-namespace.yaml
```

Langkah 2: Melakukan instalasi Fluentd

Mulai proses ini dengan mengunduh Fluentd. Setelah Anda menyelesaikan langkah-langkah ini, deployment tersebut akan membuat sumber daya berikut pada klaster:

- Sebuah akun layanan yang bernama `fluentd` dalam namespace `amazon-cloudwatch`. Akun layanan ini digunakan untuk menjalankan Fluentd DaemonSet. Untuk informasi selengkapnya, silakan lihat Mengelola Akun Layanan dalam Referensi Kubernetes.
- Sebuah peran klaster yang diberi nama `fluentd` dalam namespace `amazon-cloudwatch`. Peran klaster ini akan memberikan izin `get`, `list`, dan `watch` pada log pod ke akun layanan `fluentd`. Untuk informasi selengkapnya, silakan lihat Gambaran umum API di Referensi Kubernetes.
- A ConfigMap bernama `fluentd-config` di `amazon-cloudwatch` namespace. Ini ConfigMap berisi konfigurasi yang akan digunakan oleh Fluentd. Untuk informasi selengkapnya, lihat Mengonfigurasi Pod untuk Digunakan ConfigMap dalam dokumentasi Tugas Kubernetes.

Cara melakukan instalasi Fluentd

1. Buat ConfigMap nama `cluster-info` dengan nama cluster dan AWS Wilayah tempat log akan dikirim. Jalankan perintah berikut, yang akan memperbarui placeholder dengan nama klaster dan Wilayah Anda.

   ```
   kubectl create configmap cluster-info \
   --from-literal=cluster.name=cluster_name \
   --from-literal=logs.region=region_name -n amazon-cloudwatch
   ```

2. Unduh dan terapkan Fluentd DaemonSet ke cluster dengan menjalankan perintah berikut. Anda harus memastikan bahwa Anda menggunakan citra kontainer dengan arsitektur yang benar. Contoh manifes hanya bekerja pada instans x86 dan akan berubah statusnya menjadi `CrashLoopBackOff` jika Anda memiliki instans Advanced RISC Machine (ARM) di klaster Anda. DaemonSet Fluentd tidak memiliki citra Docker multi-arsitektur resmi yang memungkinkan Anda menggunakan satu tanda untuk beberapa citra yang mendasari dan membiarkan runtime

kontainer menarik yang benar. Citra ARM FluentD menggunakan tanda yang berbeda dengan akhiran `arm64`.

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/fluentd/fluentd.yaml
```

> **Note**
>
> Karena perubahan terbaru yang dibuat untuk mengoptimalkan konfigurasi Fluentd dan meminimalkan dampak permintaan API Fluentd pada titik akhir API Kubernetes, opsi "Tonton" untuk filter Kubernetes telah dinonaktifkan secara default. Untuk detail selengkapnya, lihat fluent-plugin-kubernetes_metadata_filter.

3. Lakukan validasi atas deployment dengan menjalankan perintah berikut. Masing-masing simpul harus memiliki satu pod yang bernama `fluentd-cloudwatch-*`.

```
kubectl get pods -n amazon-cloudwatch
```

Langkah 3: Melakukan verifikasi atas pengaturan Fluentd

Untuk melakukan verifikasi atas pengaturan Fluentd Anda, gunakan langkah-langkah berikut.

Cara melakukan verifikasi atas pengaturan Fluentd untuk Wawasan Kontainer

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Grup log. Anda harus memastikan bahwa Anda berada di Wilayah tempat menerapkan Fluentd Anda ke kontainer Anda.

   Dalam daftar grup log yang ada di Wilayah, Anda akan melihat hal-hal berikut:

   - `/aws/containerinsights/`*`Cluster_Name`*`/application`
   - `/aws/containerinsights/`*`Cluster_Name`*`/host`
   - `/aws/containerinsights/`*`Cluster_Name`*`/dataplane`

   Jika Anda melihat grup log ini, artinya penyiapan Fluentd sudah diverifikasi.

Dukungan log multi-baris

Pada tanggal 19 Agustus 2019, kami menambahkan dukungan log multi-baris untuk log yang dikumpulkan oleh Fluentd.

Secara default, awalan entri log multi-baris adalah karakter apa pun tanpa ruang spasi putih. Ini artinya bahwa semua baris log yang dimulai dengan karakter yang tidak memiliki ruang spasi putih akan dianggap sebagai entri log multi-baris baru.

Jika log aplikasi Anda sendiri menggunakan awalan multi-baris yang berbeda, maka Anda dapat mendukungnya dengan melakukan dua perubahan berikut dalam file `fluentd.yaml`.

Pertama, Anda harus mengecualikannya dari dukungan multi-baris bawaan dengan menambahkan nama jalur file log Anda ke bidang `exclude_path` yang ada di bagian `containers` dari `fluentd.yaml`. Berikut ini salah satu contohnya.

```
<source>
  @type tail
  @id in_tail_container_logs
  @label @containers
  path /var/log/containers/*.log
  exclude_path ["full_pathname_of_log_file*", "full_pathname_of_log_file2*"]
```

Berikutnya, tambahkan sebuah blok untuk file log Anda ke file `fluentd.yaml`. Contoh di bawah ini digunakan untuk file log CloudWatch agen, yang menggunakan ekspresi reguler stempel waktu sebagai starter multi-baris. Anda dapat menyalin blok ini dan menambahkannya ke `fluentd.yaml`. Ubah baris yang ditunjukkan untuk mencerminkan nama file log aplikasi dan awalan multi-baris yang ingin Anda gunakan.

```
<source>
  @type tail
  @id in_tail_cwagent_logs
  @label @cwagentlogs
  path /var/log/containers/cloudwatch-agent*
  pos_file /var/log/cloudwatch-agent.log.pos
  tag *
  read_from_head true
 <parse>
    @type json
```

```
    time_format %Y-%m-%dT%H:%M:%S.%NZ
 </parse>
</source>
```

```
<label @cwagentlogs>
  <filter **>
    @type kubernetes_metadata
    @id filter_kube_metadata_cwagent
  </filter>

  <filter **>
    @type record_transformer
    @id filter_cwagent_stream_transformer
    <record>
      stream_name ${tag_parts[3]}
    </record>
 </filter>

  <filter **>
    @type concat
    key log
    multiline_start_regexp /^\d{4}[-/]\d{1,2}[-/]\d{1,2}/
    separator ""
    flush_interval 5
    timeout_label @NORMAL
 </filter>

 <match **>
    @type relabel
    @label @NORMAL
 </match>
</label>
```

(Opsional) Mengurangi volume log dari Fluentd

Secara default, kami mengirim log aplikasi Fluentd dan metadata Kubernetes ke. CloudWatch Jika Anda ingin mengurangi volume data yang dikirim CloudWatch, Anda dapat menghentikan salah satu atau kedua sumber data ini agar tidak dikirim CloudWatch.

Untuk menghentikan log aplikasi Fluentd, Anda harus menghapus bagian berikut dari file `fluentd.yaml`.

```
<source>
  @type tail
  @id in_tail_fluentd_logs
  @label @fluentdlogs
  path /var/log/containers/fluentd*
  pos_file /var/log/fluentd.log.pos
  tag *
  read_from_head true
   <parse>
   @type json
   time_format %Y-%m-%dT%H:%M:%S.%NZ
  </parse>
</source>

<label @fluentdlogs>
  <filter **>
    @type kubernetes_metadata
    @id filter_kube_metadata_fluentd
  </filter>

  <filter **>
    @type record_transformer
    @id filter_fluentd_stream_transformer
    <record>
      stream_name ${tag_parts[3]}
    </record>
   </filter>

   <match **>
     @type relabel
     @label @NORMAL
   </match>
</label>
```

Untuk menghapus metadata Kubernetes agar tidak ditambahkan ke peristiwa log yang dikirim CloudWatch, tambahkan satu baris ke bagian dalam file. `record_transformer` `fluentd.yaml` Di sumber log tempat Anda ingin menghapus metadata ini, Anda harus menambahkan baris berikut.

```
remove_keys $.kubernetes.pod_id, $.kubernetes.master_url,
 $.kubernetes.container_image_id, $.kubernetes.namespace_id
```

Sebagai contoh:

```
<filter **>
  @type record_transformer
  @id filter_containers_stream_transformer
  <record>
    stream_name ${tag_parts[3]}
  </record>
  remove_keys $.kubernetes.pod_id, $.kubernetes.master_url,
 $.kubernetes.container_image_id, $.kubernetes.namespace_id
</filter>
```

Pemecahan Masalah

Jika Anda tidak melihat grup log ini dan mencari di Wilayah yang benar, periksa log untuk DaemonSet pod Fluentd untuk mencari kesalahan.

Jalankan perintah berikut dan pastikan bahwa statusnya `Running`.

```
kubectl get pods -n amazon-cloudwatch
```

Pada hasil perintah sebelumnya, Anda harus memperhatikan nama pod yang dimulai dengan `fluentd-cloudwatch`. Gunakan nama pod ini dalam perintah berikut.

```
kubectl logs pod_name -n amazon-cloudwatch
```

Jika log memiliki kesalahan terkait izin IAM, Anda perlu memeriksa peran IAM yang sudah dilampirkan ke klaster simpul. Untuk informasi selengkapnya tentang izin-izin yang diperlukan untuk menjalankan sebuah klaster Amazon EKS, silakan lihat Kebijakan IAM, Peran, dan Izin Amazon EKS dalam Panduan Pengguna Amazon EKS .

Jika status pod tersebut adalah `CreateContainerConfigError`, dapatkan kesalahan yang tepat dengan menjalankan perintah berikut.

```
kubectl describe pod pod_name -n amazon-cloudwatch
```

Jika status pod adalah `CrashLoopBackOff`, Anda harus memastikan bahwa arsitektur citra kontainer Fluentd sama dengan simpul ketika Anda melakukan instalasi Fluentd. Jika klaster Anda memiliki simpul x86 dan ARM64, maka Anda dapat menggunakan label kubernetes.io/arch untuk

menempatkan citra pada simpul yang semestinya. Untuk informasi selengkapnya, silakan lihat kuberntes.io/arch.

(Opsional) Menyiapkan pencatatan log bidang kontrol Amazon EKS

Jika Anda menggunakan Amazon EKS, Anda dapat mengaktifkan pencatatan pesawat kontrol Amazon EKS secara opsional, untuk menyediakan log audit dan diagnostik langsung dari bidang kontrol Amazon EKS ke CloudWatch Log. Untuk informasi selengkapnya, silakan lihat Pencatatan Log Bidang Kontrol Amazon EKS.

(Opsional) Mengaktifkan log akses App Mesh Envoy

Anda dapat mengatur Container Insights Fluentd untuk mengirim log akses App Mesh Envoy ke Log. CloudWatch Untuk informasi selengkapnya, silakan lihat Pencatatan log.

Agar log akses Utusan dikirim ke Log CloudWatch

1. Menyiapkan Fluentd dalam klaster. Untuk informasi selengkapnya, lihat (Opsional) Siapkan Fluentd sebagai DaemonSet untuk mengirim log ke Log CloudWatch .
2. Mengonfigurasi log akses Utusan untuk simpul virtual Anda. Untuk mendapatkan petunjuknya, silakan lihat Pencatatan log. Anda harus memastikan untuk mengonfigurasi jalur log menjadi **`/dev/stdout`** di masing-masing simpul virtual.

Setelah Anda selesai, log akses utusan akan dikirim ke grup log `/aws/containerinsights/`*`Cluster_Name`*`/application`.

(Opsional) Mengaktifkan fitur Use_Kubelet untuk klaster besar

Secara default, fitur Use_Kubelet dinonaktifkan di plugin Kubernetes. FluentBit Dengan mengaktifkan fitur ini, Anda akan dapat mengurangi lalu lintas ke server API dan mengurangi masalah Server API menjadi hambatan. Kami menyarankan agar Anda mengaktifkan fitur ini untuk klaster besar.

Untuk mengaktifkan Use_Kubelet, pertama-tama Anda harus menambahkan simpul dan izin simpul/proksi ke konfigurasi clusterRole.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: fluent-bit-role
rules:
  - nonResourceURLs:
```

```
      - /metrics
    verbs:
      - get
  - apiGroups: [""]
    resources:
      - namespaces
      - pods
      - pods/logs
      - nodes
      - nodes/proxy
    verbs: ["get", "list", "watch"]
```

Dalam DaemonSet konfigurasi, fitur ini membutuhkan akses jaringan host. Versi citra untuk `amazon/aws-for-fluent-bit` harus versi 2.12.0 atau versi yang lebih baru, atau versi citra bit fluent harus versi 1.7.2 atau versi yang lebih baru.

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluent-bit
  namespace: amazon-cloudwatch
  labels:
    k8s-app: fluent-bit
    version: v1
    kubernetes.io/cluster-service: "true"
spec:
  selector:
    matchLabels:
      k8s-app: fluent-bit
  template:
    metadata:
      labels:
        k8s-app: fluent-bit
        version: v1
        kubernetes.io/cluster-service: "true"
    spec:
      containers:
      - name: fluent-bit
        image: amazon/aws-for-fluent-bit:2.19.0
        imagePullPolicy: Always
        env:
            - name: AWS_REGION
              valueFrom:
```

```
              configMapKeyRef:
                name: fluent-bit-cluster-info
                key: logs.region
          - name: CLUSTER_NAME
            valueFrom:
              configMapKeyRef:
                name: fluent-bit-cluster-info
                key: cluster.name
          - name: HTTP_SERVER
            valueFrom:
              configMapKeyRef:
                name: fluent-bit-cluster-info
                key: http.server
          - name: HTTP_PORT
            valueFrom:
              configMapKeyRef:
                name: fluent-bit-cluster-info
                key: http.port
          - name: READ_FROM_HEAD
            valueFrom:
              configMapKeyRef:
                name: fluent-bit-cluster-info
                key: read.head
          - name: READ_FROM_TAIL
            valueFrom:
              configMapKeyRef:
                name: fluent-bit-cluster-info
                key: read.tail
          - name: HOST_NAME
            valueFrom:
              fieldRef:
                fieldPath: spec.nodeName
          - name: HOSTNAME
            valueFrom:
              fieldRef:
                apiVersion: v1
                fieldPath: metadata.name
          - name: CI_VERSION
            value: "k8s/1.3.8"
        resources:
          limits:
            memory: 200Mi
          requests:
            cpu: 500m
```

```
            memory: 100Mi
        volumeMounts:
        # Please don't change below read-only permissions
        - name: fluentbitstate
          mountPath: /var/fluent-bit/state
        - name: varlog
          mountPath: /var/log
          readOnly: true
        - name: varlibdockercontainers
          mountPath: /var/lib/docker/containers
          readOnly: true
        - name: fluent-bit-config
          mountPath: /fluent-bit/etc/
        - name: runlogjournal
          mountPath: /run/log/journal
          readOnly: true
        - name: dmesg
          mountPath: /var/log/dmesg
          readOnly: true
      terminationGracePeriodSeconds: 10
      hostNetwork: true
      dnsPolicy: ClusterFirstWithHostNet
      volumes:
      - name: fluentbitstate
        hostPath:
          path: /var/fluent-bit/state
      - name: varlog
        hostPath:
          path: /var/log
      - name: varlibdockercontainers
        hostPath:
          path: /var/lib/docker/containers
      - name: fluent-bit-config
        configMap:
          name: fluent-bit-config
      - name: runlogjournal
        hostPath:
          path: /run/log/journal
      - name: dmesg
        hostPath:
          path: /var/log/dmesg
      serviceAccountName: fluent-bit
      tolerations:
      - key: node-role.kubernetes.io/master
```

```
        operator: Exists
        effect: NoSchedule
      - operator: "Exists"
        effect: "NoExecute"
      - operator: "Exists"
        effect: "NoSchedule"
```

Konfigurasi Plugin Kubernetes harus serupa dengan konfigurasi berikut ini:

```
[FILTER]
        Name                kubernetes
        Match               application.*
        Kube_URL            https://kubernetes.default.svc:443
        Kube_Tag_Prefix     application.var.log.containers.
        Merge_Log           On
        Merge_Log_Key       log_processed
        K8S-Logging.Parser  On
        K8S-Logging.Exclude Off
        Labels              Off
        Annotations         Off
        Use_Kubelet         On
        Kubelet_Port        10250
        Buffer_Size         0
```

Memperbarui atau menghapus Wawasan Kontainer pada Amazon EKS dan Kubernetes

Gunakan langkah-langkah di bagian ini untuk memperbarui image container CloudWatch agen Anda, atau untuk menghapus Container Insights dari kluster Amazon EKS atau Kubernetes.

Topik

- Meningkatkan ke Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
- Memperbarui gambar kontainer CloudWatch agen
- Menghapus CloudWatch agen dan Fluentd untuk Wawasan Kontainer

Meningkatkan ke Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS

Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS adalah Wawasan Kontainer versi terbaru. Wawasan Kontainer ini mengumpulkan metrik-metrik terperinci dari klaster yang menjalankan Amazon EKS dan menawarkan dasbor yang dikuratori, dan digunakan langsung untuk menelusuri telemetri aplikasi dan infrastruktur lebih dalam saat itu juga. Untuk informasi

selengkapnya tentang versi Wawasan Kontainer ini, silakan lihat Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS.

Jika Anda telah menyelesaikan instalasi versi asli Wawasan Kontainer di sebuah klaster Amazon EKS dan Anda ingin meningkatkannya ke versi yang lebih baru dengan observabilitas yang disempurnakan, silakan ikuti instruksi-instruksi yang dijelaskan di bagian ini.

Untuk meningkatkan sebuah klaster Amazon EKS ke Wawasan Kontainer dengan kemampuan peningkatan observabilitas untuk Amazon EKS

1. Perbarui versi terbaru CloudWatch agen dengan mengikuti instruksi diMemperbarui gambar kontainer CloudWatch agen.
2. Mengubah agen `configmap` untuk menambahkan `"enhanced_container_insights": true`

   Sebagai contoh, masukkan perintah berikut:

   ```
   kubectl edit configmap/cwagentconfig -n amazon-cloudwatch
   ```

   Sunting file agar terlihat seperti berikut. Anda bisa membiarkan sisa konfigurasi Anda sebelumnya dan tidak mengubahnya.

   ```
   apiVersion: v1
   data:
     cwagentconfig.json: |
       {
         "logs": {
           "metrics_collected": {
             "kubernetes": {
               "cluster_name": "my-cluster-name",
               "enhanced_container_insights": true,
               "metrics_collection_interval": 60
             }
           },
           "force_flush_interval": 5
         }
       }
   kind: ConfigMap
   ```

Memperbarui gambar kontainer CloudWatch agen

Jika Anda perlu melakukan pembaruan citra kontainer Anda ke versi terbaru, gunakan langkah-langkah yang dijelaskan di bagian ini.

Cara memperbarui citra kontainer Anda

1. Menerapkan file `cwagent-serviceaccount.yaml` terbaru dengan memasukkan perintah berikut.

   ```
   kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cwagent/cwagent-serviceaccount.yaml
   ```

2. Langkah ini hanya diperlukan untuk pelanggan yang meningkatkan CloudWatch agen kontainer mereka dari versi lebih awal dari 1.226589.0, yang dirilis pada 20 Agustus 2019.

   Di file Configmap `cwagentconfig`, ganti kata kunci `structuredlogs` untuk `logs`

   a. Pertama, buka `cwagentconfig` dalam mode sunting dengan memasukkan perintah berikut.

      ```
      kubectl edit cm cwagentconfig -n amazon-cloudwatch
      ```

      Dalam file tersebut, jika Anda melihat kata kunci `structuredlogs`, ubah menjadi `logs`

   b. Masukkan **wq** untuk menyimpan file dan keluar dari mode sunting.

3. Menerapkan file `cwagent-daemonset.yaml` terbaru dengan memasukkan perintah berikut.

   ```
   kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cwagent/cwagent-daemonset.yaml
   ```

Anda dapat mencapai pembaruan bergulir dari CloudWatch agen DaemonSet kapan saja Anda mengubah konfigurasi Andacwagent-configmap.yaml. Untuk melakukan hal ini, Anda harus memastikan bagian `.spec.template` dalam file `cwagent-daemonset.yaml` memiliki perubahan. Jika tidak, Kubernetes memperlakukan sebagai tidak berubah. DaemonSet Praktik yang umum adalah menambahkan nilai hash dari ConfigMap ke dalam`.spec.template.metadata.annotations.configHash`, seperti pada contoh berikut.

yq versi 4

```
$HASH=`kubectl get cm/cwagentconfig -n amazon-cloudwatch -o yaml | sha256sum`
 yq -i '.spec.template.metadata.annotations.configHash =  strenv(HASH)' cwagent-
daemonset.yaml
```

```
$cat cwagent-daemonset.yaml |grep Hash
```

```
$kubectl apply -f cwagent-daemonset.yaml
```

yq versi 3

```
yq w -i cwagent-daemonset.yaml spec.template.metadata.annotations.configHash $(kubectl
 get cm/cwagentconfig -n amazon-cloudwatch -o yaml | sha256sum)
```

Hal ini akan menambahkan nilai hash ke dalam `cwagent-daemonset.yaml` seperti dalam contoh berikut.

```
spec:
  selector:
    matchLabels:
      name: cloudwatch-agent
  template:
    metadata:
      labels:
        name: cloudwatch-agent
      annotations:
        configHash: 88915de4cf9c3551a8dc74c0137a3e83569d28c71044b0359c2578d2e0461825
```

Kemudian, jalankan perintah berikut untuk mengambil konfigurasi baru.

```
kubectl apply -f cwagent-daemonset.yaml
```

Untuk informasi selengkapnya tentang yq, silakan lihat yq.

Menghapus CloudWatch agen dan Fluentd untuk Wawasan Kontainer

Jika Anda menginstal Container Insights dengan menginstal add-on CloudWatch Observability untuk Amazon EKS, Anda dapat menghapus Wawasan Kontainer dan CloudWatch agen dengan memasukkan perintah berikut:

```
aws eks delete-addon --cluster-name my-cluster --addon-name amazon-cloudwatch-
observability
```

Jika tidak, untuk menghapus semua sumber daya yang terkait dengan CloudWatch agen dan Fluentd, masukkan perintah berikut. Dalam perintah ini, *Cluster_Name* adalah nama dari klaster Amazon EKS atau Kubernetes Anda, dan *Region* adalah nama dari Wilayah tempat log diterbitkan.

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/
container-insights-monitoring/quickstart/cwagent-fluentd-quickstart.yaml | sed "s/
{{cluster_name}}/Cluster_Name/;s/{{region_name}}/Region/" | kubectl delete -f -
```

Menampilkan metrik-metrik Wawasan Kontainer

Setelah menyiapkan Container Insights dan mengumpulkan metrik, Anda dapat melihat metrik tersebut di konsol. CloudWatch

Agar metrik-metrik Wawasan Kontainer ditampilkan di dasbor Anda, Anda harus menyelesaikan penyiapan Wawasan Kontainer. Untuk informasi selengkapnya, lihat Menyiapkan Wawasan Kontainer.

Prosedur ini akan menjelaskan cara menampilkan metrik-metrik yang dihasilkan secara otomatis oleh Wawasan Kontainer dari data log yang dikumpulkan. Bagian lainnya menjelaskan cara menyelam lebih jauh ke dalam data Anda dan menggunakan Wawasan CloudWatch Log untuk melihat lebih banyak metrik pada tingkat perincian yang lebih banyak.

Cara menampilkan metrik-metrik Wawasan Kontainer

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Wawasan, Wawasan Kontainer.
3. Pada menu geser-turun yang ada di Wawasan Kontainer, pilih Pemantauan performa.
4. Gunakan kotak geser-turun di bagian atas untuk memilih jenis sumber daya yang akan ditampilkan, serta sumber daya tertentu.

Anda dapat menyetel CloudWatch alarm pada metrik apa pun yang dikumpulkan oleh Container Insights. Lihat informasi yang lebih lengkap di Menggunakan CloudWatch alarm Amazon

> **Note**
>
> Jika Anda telah menyiapkan CloudWatch Application Insights untuk memantau aplikasi kontainer Anda, dasbor Application Insights akan muncul di bawah dasbor Container Insights. Jika Anda belum mengaktifkan Wawasan Aplikasi, maka Anda dapat melakukannya dengan memilih Mengonfigurasi Wawasan Aplikasi secara otomatis di bawah tampilan performa di dasbor Wawasan Kontainer.
> Untuk informasi selengkapnya tentang Wawasan Aplikasi dan aplikasi terkontainer, silakan lihat Aktifkan Wawasan Aplikasi untuk melakukan pemantauan sumber daya Amazon ECS dan Amazon EKS.

Menampilkan kontributor teratas

Untuk beberapa tampilan dalam pemantauan performa Wawasan Kontainer, Anda juga dapat melihat kontributor-kontributor teratas berdasarkan memori atau CPU, atau sumber daya yang terakhir aktif. Hal ini akan tersedia bila Anda memilih salah satu dasbor berikut di kotak geser-turun yang ada di sekitar bagian atas halaman:

- Layanan ECS
- Tugas ECS
- Namespace EKS
- Layanan EKS
- Pod EKS

Ketika Anda menampilkan salah satu jenis sumber daya, bagian bawah halaman akan menampilkan sebuah tabel yang awalnya diurutkan berdasarkan penggunaan CPU. Anda dapat mengubahnya untuk melakukan pengurutan berdasarkan penggunaan memori atau aktivitas terbaru. Untuk melihat detail tentang salah satu baris dalam tabel, Anda dapat memilih kotak centang yang ada di sebelah baris tersebut dan kemudian memilih Tindakan dan memilih salah satu opsi yang ada di menu Tindakan.

Menggunakan Wawasan CloudWatch Log untuk melihat data Wawasan Kontainer

Wawasan Kontainer mengumpulkan metrik-metrik dengan menggunakan peristiwa log performa dengan menggunakan format metrik tersemat. Log disimpan di CloudWatch Log. CloudWatch menghasilkan beberapa metrik secara otomatis dari log yang dapat Anda lihat di CloudWatch konsol.

Anda juga dapat melakukan analisis lebih dalam terhadap data kinerja yang dikumpulkan dengan menggunakan kueri Wawasan CloudWatch Log.

Untuk informasi selengkapnya tentang Wawasan CloudWatch Log, lihat Menganalisis Data Log dengan Wawasan CloudWatch Log. Untuk informasi selengkapnya tentang bidang log yang dapat Anda gunakan dalam kueri, silakan lihat Peristiwa log performa Wawasan Kontainer untuk Amazon EKS dan Kubernetes.

Untuk menggunakan Wawasan CloudWatch Log untuk menanyakan data metrik penampung Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Wawasan.

   Di sekitar bagian atas layar bisa Anda temukan editor kueri. Saat pertama kali membuka Wawasan CloudWatch Log, kotak ini berisi kueri default yang menampilkan 20 peristiwa log terbaru.
3. Pada kotak yang ada di atas editor kueri, pilih salah satu grup log Wawasan Kontainer untuk menjalankan kueri. Untuk contoh kueri yang akan digunakan berikut, nama grup log harus diakhiri dengan performance.

   Saat Anda memilih grup CloudWatch log, Wawasan Log secara otomatis mendeteksi bidang dalam data dalam grup log dan menampilkannya di bidang Ditemukan di panel kanan. Ia juga akan menampilkan grafik batang peristiwa log dalam grup log ini dari waktu ke waktu. Grafik batang ini akan menunjukkan distribusi peristiwa dalam grup log yang sesuai dengan kueri dan rentang waktu Anda, tidak hanya peristiwa-peristiwa yang ditampilkan dalam tabel.
4. Dalam editor kueri, Anda harus mengganti kueri bawaan dengan kueri berikut dan pilih Jalankan kueri.

   ```
   STATS avg(node_cpu_utilization) as avg_node_cpu_utilization by NodeName
   | SORT avg_node_cpu_utilization DESC
   ```

   Kueri ini akan menunjukkan daftar simpul, yang diurutkan berdasarkan pemanfaatan CPU simpul rata-rata.
5. Untuk mencoba contoh yang lain, silakan ganti kueri tersebut dengan kueri lain dan kemudian pilih Jalankan kueri. Contoh kueri-kueri lainnya dicantumkan kemudian di halaman ini.

   ```
   STATS avg(number_of_container_restarts) as avg_number_of_container_restarts by
    PodName
   ```

```
| SORT avg_number_of_container_restarts DESC
```

Kueri ini akan menampilkan daftar pod Anda, yang diurutkan berdasarkan jumlah rata-rata mulai ulang kontainer.

6. Apabila Anda ingin mencoba kueri yang lain, Anda harus menggunakan bidang-bidang yang disertakan dalam daftar yang ada di sebelah kanan layar. Untuk informasi selengkapnya tentang sintaks kueri, lihat Sintaks Kueri Wawasan CloudWatch Log.

Cara menampilkan daftar sumber daya Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Sumber Daya.
3. Tampilan bawaannya adalah daftar sumber daya Anda yang dipantau oleh Wawasan Kontainer, dan alarm-alarm yang telah Anda tetapkan pada sumber daya ini. Cara menampilkan peta visual sumber daya, pilih Tampilan peta.
4. Dari tampilan peta, Anda dapat menghentikan kursor Anda di atas sumber daya yang ada di peta tersebut untuk melihat metrik-metrik dasar tentang sumber daya tersebut. Anda dapat memilih sumber daya apa pun untuk melihat grafik terperinci tentang sumber daya tersebut.

Kasus penggunaan: Melihat metrik-metrik tingkat-tugas di kontainer Amazon ECS

Contoh berikut mengilustrasikan cara menggunakan Wawasan CloudWatch Log untuk menyelam lebih dalam ke log Wawasan Kontainer Anda. Untuk contoh lainnya, lihat blog Memperkenalkan CloudWatch Wawasan Kontainer Amazon untuk Amazon ECS.

Wawasan Kontainer tidak akan menghasilkan metrik terperinci pada tingkat Tugas secara otomatis. Kueri berikut akan menampilkan metrik-metrik tingkat tugas untuk pemanfaatan CPU dan memori.

```
stats avg(CpuUtilized) as CPU, avg(MemoryUtilized) as Mem by TaskId, ContainerName
| sort Mem, CPU desc
```

Contoh kueri lainnya untuk Wawasan Kontainer

Daftar pod Anda, yang diurutkan berdasarkan jumlah rata-rata mulai ulang kontainer

```
STATS avg(number_of_container_restarts) as avg_number_of_container_restarts by PodName
| SORT avg_number_of_container_restarts DESC
```

Pod yang diminta vs. pod yang berjalan

```
fields @timestamp, @message
| sort @timestamp desc
| filter Type="Pod"
| stats min(pod_number_of_containers) as requested,
 min(pod_number_of_running_containers) as running, ceil(avg(pod_number_of_containers-
pod_number_of_running_containers)) as pods_missing by kubernetes.pod_name
| sort pods_missing desc
```

Jumlah kegagalan simpul klaster

```
stats avg(cluster_failed_node_count) as CountOfNodeFailures
| filter Type="Cluster"
| sort @timestamp desc
```

Kesalahan log aplikasi berdasarkan nama kontainer

```
stats count() as countoferrors by kubernetes.container_name
| filter stream="stderr"
| sort countoferrors desc
```

Penggunaan disk berdasarkan nama kontainer

```
stats floor(avg(container_filesystem_usage/1024)) as container_filesystem_usage_avg_kb
 by InstanceId, kubernetes.container_name, device
| filter Type="ContainerFS"
| sort container_filesystem_usage_avg_kb desc
```

Penggunaan CPU berdasarkan nama kontainer

```
stats pct(container_cpu_usage_total, 50) as CPUPercMedian by kubernetes.container_name
| filter Type="Container"
```

Metrik-metrik yang dikumpulkan oleh Wawasan Kontainer

Container Insights mengumpulkan satu set metrik untuk Amazon ECS dan Amazon ECS, dan satu set berbeda untuk Amazon EKS, di AWS Fargate Amazon EKS, dan Kubernetes. AWS Fargate

Metrik-metrik tidak terlihat sampai tugas kontainer telah berjalan selama beberapa waktu.

Topik

Metrik-metrik Wawasan Kontainer Amazon ECS

Tabel berikut mencantumkan metrik-metrik dan dimensi yang dikumpulkan oleh Wawasan Kontainer untuk Amazon ECS. Metrik-metrik ini berada di namespace `ECS/ContainerInsights`. Untuk informasi selengkapnya, lihat Metrik.

Jika Anda tidak melihat metrik Wawasan Kontainer di konsol Anda, maka Anda harus memastikan bahwa telah menyelesaikan penyiapan Wawasan Kontainer. Metrik tidak akan ditampilkan sebelum Wawasan Kontainer telah disiapkan sepenuhnya. Untuk informasi selengkapnya, lihat Menyiapkan Wawasan Kontainer.

Metrik-metrik berikut tersedia saat Anda menyelesaikan langkah-langkah yang diuraikan dalam Menyiapkan Wawasan Kontainer di Amazon ECS untuk metrik-metrik tingkat klaster dan layanan

Nama metrik	Dimensi	Deskripsi
`ContainerInstanceCount`	ClusterName	Jumlah instans EC2 yang menjalankan agen Amazon ECS yang terdaftar pada sebuah klaster. Metrik ini dikumpulkan hanya untuk instans kontainer yang menjalankan tugas-tugas Amazon ECS di klaster tersebut. Metrik tidak dikumpulkan untuk instans kontainer kosong yang tidak memiliki tugas-tugas Amazon ECS.

Nama metrik	Dimensi	Deskripsi
		Satuan: Hitungan
CpuUtilized	TaskDefinitionFamily , ClusterName ServiceName , ClusterName ClusterName	Unit CPU yang digunakan berdasarkan tugas-tugas dalam sumber daya yang ditentukan oleh serangkaian dimensi yang Anda gunakan. Metrik ini dikumpulkan hanya untuk tugas-tugas yang memiliki reservasi CPU yang sudah ditentukan dalam penetapan tugas mereka. Satuan: Tidak ada
CpuReserved	TaskDefinitionFamily , ClusterName ServiceName , ClusterName ClusterName	Unit CPU yang direservasi berdasarkan tugas-tugas dalam sumber daya yang ditentukan oleh serangkaian dimensi yang Anda gunakan. Metrik ini dikumpulkan hanya untuk tugas-tugas yang memiliki reservasi CPU yang sudah ditentukan dalam penetapan tugas mereka. Satuan: Tidak ada

Nama metrik	Dimensi	Deskripsi
DeploymentCount	ServiceName , ClusterName	Jumlah deployment dalam sebuah layanan Amazon ECS. Satuan: Hitungan
DesiredTaskCount	ServiceName , ClusterName	Jumlah tugas yang diinginkan untuk sebuah layanan Amazon ECS. Unit: Jumlah
EBSFilesystemSize	VolumeName , TaskDefinitionFamily , ClusterName TaskDefinitionFamily , ClusterName ServiceName , ClusterName	Jumlah total, dalam gigabyte (GB), penyimpanan sistem file Amazon EBS yang dialokasikan ke sumber daya yang ditentukan oleh dimensi yang Anda gunakan. Metrik ini hanya tersedia untuk tugas yang berjalan di infrastruktur Amazon ECS yang berjalan di Fargate menggunakan 1.4.0 versi platform atau instans Amazon EC2 menggunakan versi agen kontainer atau yang lebih baru. 1.79.0 Satuan: Gigabyte (GB)

Nama metrik	Dimensi	Deskripsi
EBSFilesystemUtilized	VolumeName , TaskDefinitionFamily , ClusterName TaskDefinitionFamily , ClusterName ServiceName , ClusterName	Jumlah total, dalam gigabyte (GB), penyimpanan sistem file Amazon EBS yang digunakan oleh sumber daya yang ditentukan oleh dimensi yang Anda gunakan. Metrik ini hanya tersedia untuk tugas yang berjalan di infrastruktur Amazon ECS yang berjalan di Fargate menggunakan 1.4.0 versi platform atau instans Amazon EC2 menggunakan versi agen kontainer atau yang lebih baru. 1.79.0 Untuk tugas yang dijalankan di Fargate, Fargate menyediakan ruang pada disk yang hanya digunakan oleh Fargate. Tidak ada biaya yang terkait dengan ruang yang digunakan Fargate, tetapi Anda akan melihat penyimpanan tambahan ini menggunakan alat seperti. df Satuan: Gigabyte (GB)

Nama metrik	Dimensi	Deskripsi

`EphemeralStorageReserved` [1]	`TaskDefinitionFamily` , `ClusterName`
	
	`ServiceName` , `ClusterName`
	
	`ClusterName`
		Jumlah byte yang dicadangkan dari penyimpanan sementara dalam sumber daya yang ditentukan oleh dimensi-dimensi yang Anda gunakan. Penyimpanan sementara digunakan untuk sistem file root kontainer dan volume host pemasangan terikat apa pun yang ditentukan dalam citra kontainer dan penetapan tugas. Jumlah penyimpanan sementara tidak dapat diubah dalam sebuah tugas yang sedang berjalan.
		
		Metrik ini hanya tersedia untuk tugas yang berjalan di versi platform Fargate Linux versi 1.4.0 atau versi yang lebih baru.
		
		Satuan: Gigabyte (GB)

Nama metrik	Dimensi	Deskripsi

`EphemeralStorageUtilized` [1]	`TaskDefinitionFamily` , `ClusterName`
	
	`ServiceName` , `ClusterName`
	
	`ClusterName`
		Jumlah byte yang digunakan dari penyimpanan sementara dalam sumber daya yang ditentukan oleh dimensi-dimensi yang Anda gunakan. Penyimpanan sementara digunakan untuk sistem file root kontainer dan volume host pemasangan terikat apa pun yang ditentukan dalam citra kontainer dan penetapan tugas. Jumlah penyimpanan sementara tidak dapat diubah dalam sebuah tugas yang sedang berjalan.
		
		Metrik ini hanya tersedia untuk tugas yang berjalan di versi platform Fargate Linux versi 1.4.0 atau versi yang lebih baru.
		
		Satuan: Gigabyte (GB)

Nama metrik	Dimensi	Deskripsi
MemoryUtilized	TaskDefinitionFamily , ClusterName ServiceName , ClusterName ClusterName	Memori yang digunakan oleh tugas-tugas yang ada di sumber daya yang ditentukan oleh serangkaian dimensi yang Anda gunakan. Metrik ini dikumpulkan hanya untuk tugas-tugas yang memiliki reservasi memori yang ditentukan dalam penetapan tugas mereka. Satuan: Megabyte
MemoryReserved	TaskDefinitionFamily , ClusterName ServiceName , ClusterName ClusterName	Memori yang dicadangkan oleh tugas-tugas yang ada di sumber daya yang ditentukan oleh serangkaian dimensi yang Anda gunakan. Metrik ini dikumpulkan hanya untuk tugas-tugas yang memiliki reservasi memori yang ditentukan dalam penetapan tugas mereka. Satuan: Megabyte

Nama metrik	Dimensi	Deskripsi
NetworkRxBytes	TaskDefinitionFamily , ClusterName ServiceName , ClusterName ClusterName	Jumlah byte yang diterima oleh sumber daya yang ditentukan oleh dimensi-dimensi yang sedang Anda gunakan. Metrik ini didapatkan dari runtime Docker. Metrik ini tersedia hanya untuk kontainer yang ada dalam tugas yang menggunakan mode jaringan awsvpc atau bridge. Satuan: Byte/Detik
NetworkTxBytes	TaskDefinitionFamily , ClusterName ServiceName , ClusterName ClusterName	Jumlah byte yang ditransmisikan oleh sumber daya yang ditentukan oleh dimensi-dimensi yang sedang Anda gunakan. Metrik ini didapatkan dari runtime Docker. Metrik ini tersedia hanya untuk kontainer yang ada dalam tugas yang menggunakan mode jaringan awsvpc atau bridge. Satuan: Byte/Detik

Nama metrik	Dimensi	Deskripsi
PendingTaskCount	ServiceName , ClusterName	Jumlah tugas saat ini yang berada dalam status PENDING. Satuan: Hitungan
RunningTaskCount	ServiceName , ClusterName	Jumlah tugas saat ini yang berada dalam status RUNNING. Satuan: Hitungan
ServiceCount	ClusterName	Jumlah layanan yang ada di klaster. Satuan: Hitungan
StorageReadBytes	TaskDefinitionFamily , ClusterName ServiceName , ClusterName ClusterName	Jumlah byte yang dibaca dari penyimpanan pada instans dalam sumber daya yang ditentukan oleh dimensi-dimensi yang Anda gunakan. Ini tidak termasuk byte baca untuk perangkat-perangkat penyimpanan Anda. Metrik ini didapatkan dari runtime Docker. Unit: Bita

Nama metrik	Dimensi	Deskripsi
StorageWriteBytes	TaskDefinitionFamily , ClusterName ServiceName , ClusterName ClusterName	Jumlah byte yang dituliskan ke penyimpanan dalam sumber daya yang ditentukan oleh dimensi-dimensi yang sedang Anda gunakan. Metrik ini didapatkan dari runtime Docker. Unit: Bita
TaskCount	ClusterName	Jumlah tugas yang berjalan di klaster. Satuan: Hitungan
TaskSetCount	ServiceName , ClusterName	Jumlah serangkaian tugas dalam layanan. Satuan: Hitungan

> **Note**
>
> Metrik EphemeralStorageReserved dan EphemeralStorageUtilized hanya tersedia untuk tugas yang berjalan di versi platform Fargate Linux versi 1.4.0 atau versi yang lebih baru.
> Fargate mencadangkan ruang pada disk. Ruang ini hanya digunakan oleh Fargate. Anda tidak akan dikenai biaya untuk ruang tersebut. Ruang ini tidak akan ditampilkan dalam metrik ini. Namun demikian, Anda dapat melihat penyimpanan tambahan ini di alat lain seperti df.

Metrik-metrik berikut tersedia saat Anda menyelesaikan langkah-langkah yang diuraikan dalam Menerapkan CloudWatch agen untuk mengumpulkan metrik tingkat instans EC2 di Amazon ECS

Nama metrik	Dimensi	Deskripsi
instance_cpu_limit	ClusterName	Jumlah unit CPU maksimum yang dapat ditetapkan untuk satu Instans EC2 dalam klaster. Satuan: Tidak ada
instance_cpu_reserved_capacity	ClusterName InstanceId , ContainerInstanceId , ClusterName	Persentase CPU yang saat ini dicadangkan pada satu instans EC2 dalam klaster. Satuan: Persen
instance_cpu_usage_total	ClusterName	Jumlah unit CPU yang sedang digunakan pada satu instans EC2 dalam klaster. Satuan: Tidak ada
instance_cpu_utilization	ClusterName InstanceId , ContainerInstanceId , ClusterName	Persentase total unit CPU yang sedang digunakan pada satu instans EC2 dalam klaster. Satuan: Persen
instance_filesystem_utilization	ClusterName InstanceId , ContainerInstanceId , ClusterName	Persentase total kapasitas sistem file yang sedang digunakan pada satu instans EC2 dalam klaster.

Nama metrik	Dimensi	Deskripsi
		Satuan: Persen
instance_memory_limit	ClusterName	Jumlah memori maksimum, dalam byte, yang dapat ditetapkan ke satu Instans EC2 dalam klaster ini. Satuan: Byte
instance_memory_reserved_capacity	ClusterName InstanceId , ContainerInstanceId , ClusterName	Persentase Memori yang saat ini dicadangkan pada satu instans EC2 dalam klaster. Satuan: Persen
instance_memory_utilization	ClusterName InstanceId , ContainerInstanceId , ClusterName	Persentase total memori yang sedang digunakan pada satu Instans EC2 dalam klaster. Satuan: Persen
instance_memory_working_set	ClusterName	Jumlah memori, dalam byte, yang sedang digunakan pada satu Instans EC2 dalam klaster. Satuan: Byte

Nama metrik	Dimensi	Deskripsi
`instance_network_total_bytes`	`ClusterName`	Jumlah total byte per detik yang ditransmisikan dan diterima melalui jaringan pada satu Instans EC2 tunggal dalam klaster. Satuan: Byte/detik
`instance_number_of_running_tasks`	`ClusterName`	Jumlah tugas berjalan pada satu Instans EC2 tunggal dalam klaster. Satuan: Hitungan

Metrik-metrik Wawasan Kontainer Amazon EKS dan Kubernetes

Tabel berikut mencantumkan metrik dan dimensi yang dikumpulkan Container Insights untuk Amazon EKS dan Kubernetes. Metrik-metrik ini berada di namespace `ContainerInsights`. Untuk informasi selengkapnya, lihat Metrik.

Jika Anda tidak melihat metrik Wawasan Kontainer di konsol Anda, maka Anda harus memastikan bahwa telah menyelesaikan penyiapan Wawasan Kontainer. Metrik tidak akan ditampilkan sebelum Wawasan Kontainer telah disiapkan sepenuhnya. Untuk informasi selengkapnya, lihat Menyiapkan Wawasan Kontainer.

Dengan Wawasan Kontainer versi asli, metrik-metrik tersebut akan dikenai biaya sebagai metrik kustom. Dengan Wawasan Kontainer yang memiliki kemampuan observabilitas yang ditingkatkan untuk Amazon EKS, metrik-metrik Wawasan Kontainer akan dikenakan biaya per observasi, bukan dibebankan per metrik yang disimpan atau log yang diserap. Untuk informasi selengkapnya tentang CloudWatch harga, lihat CloudWatch Harga Amazon.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi
`cluster_failed_node_count`	`ClusterName`		Jumlah simpul pekerja yang mengalami kegagalan dalam klaster. Sebuah simpul dianggap mengalami kegagalan jika mengalami kondisi simpul apa pun. Untuk informasi selengkapnya tentang hal itu, silakan lihat Kondisi dalam dokumentasi Kubernetes.
`cluster_node_count`	`ClusterName`		Jumlah total simpul pekerja yang ada di klaster.

`namespace_number_of_running_pods`	`Namespace` `ClusterName`
	
	`ClusterName`
			Jumlah pod yang berjalan untuk masing-masing namespace dalam sumber daya yang ditentukan oleh dimensi-dimensi yang sedang Anda gunakan.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi
`node_cpu_limit`	`ClusterName`	`ClusterName` , `InstanceId` , `NodeName`	Jumlah maksimum unit CPU yang dapat ditetapkan untuk satu simpul tunggal dalam klaster ini.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`node_cpu_reserved_capacity`	`NodeName`, `ClusterName` , `InstanceId`
	
	`ClusterName`
			Persentase unit CPU yang dicadangkan untuk komponen-komponen simpul, seperti kubelet, kube-proxy, dan Docker.
			
			Rumus: `node_cpu_request / node_cpu_limit`
			
			Note
			`node_cpu_request` tidak dilaporkan secara langsung sebagai sebuah metrik, tetapi merupakan sebuah bidang dalam peristiwa log performa. Untuk informasi selengkapnya, lihat

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi
			Bidang-bidang relevan dalam peristiwa log performa untuk Amazon EKS dan Kubernetes.
`node_cpu_usage_total`	`ClusterName`	`ClusterName` , `InstanceId` , `NodeName`	Jumlah unit CPU yang sedang digunakan pada simpul di klaster.

`node_cpu_utilization`	`NodeName`, `ClusterName` , `InstanceId`
	
	`ClusterName`
			Persentase total unit CPU yang sedang digunakan pada simpul di klaster.
			
			Rumus: `node_cpu_usage_total / node_cpu_limit`

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`node_filesystem_utilization`	`NodeName`, `ClusterName` , `InstanceId`
	
	`ClusterName`
			Persentase total kapasitas sistem file yang sedang digunakan pada satu simpul tunggal dalam klaster.
			
			Rumus: `node_filesystem_usage / node_filesystem_capacity`
			
			Note
			`node_filesystem_usage` dan `node_filesystem_capacity` tidak akan dilaporkan secara langsung sebagai metrik, tetapi sebagai bidang dalam peristiwa log

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi
			performa. Untuk informasi selengkap nya, lihat Bidang-bidang relevan dalam peristiwa log performa untuk Amazon EKS dan Kubernetes.
`node_memory_limit`	`ClusterName`	`ClusterName` , `InstanceId` , `NodeName`	Jumlah memori maksimum, dalam byte, yang dapat ditetapkan ke satu simpul tunggal dalam klaster ini.

`node_filesystem_inodes`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName` , `InstanceId` , `NodeName`
			Jumlah total inode (yang digunakan dan tidak digunakan) pada sebuah simpul.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`node_filesystem_inodes_free`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName` , `InstanceId` , `NodeName`
			Jumlah inode yang tidak digunakan pada sebuah simpul.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`node_memory_reserved_capacity`	`NodeName`, `ClusterName` , `InstanceId`
	
	`ClusterName`
			Persentase memori yang saat ini sedang digunakan pada simpul di klaster.
			
			Rumus: `node_memory_request / node_memory_limit`
			
			Note
			`node_memory_request` tidak dilaporkan secara langsung sebagai sebuah metrik, tetapi merupakan sebuah bidang dalam peristiwa log performa. Untuk informasi selengkapnya, lihat

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi
			Bidang-bidang relevan dalam peristiwa log performa untuk Amazon EKS dan Kubernetes.
node_memory_utilization	NodeName, ClusterName , InstanceId ClusterName		Persentase memori yang saat ini sedang digunakan oleh simpul atau simpul-simpul tersebut. Ini adalah persentase penggunaan memori simpul yang dibagi dengan batasan memori simpul. Rumus: node_memory_working_set / node_memory_limit .
node_memory_working_set	ClusterName	ClusterName , InstanceId , NodeName	Jumlah memori, dalam byte, yang sedang digunakan dalam serangkaian simpul dalam klaster.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi
node_network_total_bytes	NodeName, ClusterName , InstanceId ClusterName		Jumlah total byte per detik yang ditransmisikan dan diterima melalui jaringan untuk setiap simpul dalam sebuah klaster. Rumus: node_network_rx_bytes + node_network_tx_bytes **Note** node_network_rx_bytes dan node_network_tx_bytes tidak akan dilaporkan secara langsung sebagai metrik, tetapi sebagai bidang dalam peristiwa log performa.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi
			Untuk informasi selengkapnya, lihat Bidang-bidang relevan dalam peristiwa log performa untuk Amazon EKS dan Kubernetes.
node_number_of_running_containers	NodeName, ClusterName , InstanceId ClusterName		Jumlah kontainer yang sedang berjalan untuk setiap simpul dalam sebuah klaster.
node_number_of_running_pods	NodeName, ClusterName , InstanceId ClusterName		Jumlah pod yang sedang berjalan untuk setiap simpul dalam sebuah klaster.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`node_status_allocatable_pods`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName`, `InstanceId`, `NodeName`
			Jumlah pod yang dapat ditetapkan ke sebuah simpul berdasarkan sumber daya yang dapat dialokasikan, yang didefinisikan sebagai sisa kapasitas simpul setelah memperhitungkan reservasi daemon sistem dan ambang batas pengosongan keras.

`node_status_capacity_pods`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName`, `InstanceId`, `NodeName`
			Jumlah pod yang dapat ditetapkan ke sebuah simpul berdasarkan kapasitasnya.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`node_status_condition_ready`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName`, `InstanceId`, `NodeName`
			Menunjukkan apakah kondisi status simpul dalam keadaan `Ready` benar.

`node_status_condition_memory_pressure`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName`, `InstanceId`, `NodeName`
			Menunjukkan apakah kondisi status simpul dalam keadaan `MemoryPressure` benar.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`node_status_condition_pid_pressure`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName` , `InstanceId` , `NodeName`
			Menunjukkan apakah kondisi status simpul dalam keadaan `PIDPressure` benar.

`node_status_condition_disk_pressure`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName` , `InstanceId` , `NodeName`
			Menunjukkan apakah kondisi status simpul dalam keadaan `OutOfDisk` benar.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`node_status_condition_unknown`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName` , `InstanceId` , `NodeName`
			Menunjukkan apakah salah satu kondisi status simpul Unknown.

`node_interface_network_rx_dropped`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName` , `InstanceId` , `NodeName`
			Jumlah paket yang diterima dan kemudian dijatuhkan oleh sebuah antarmuka jaringan pada simpul.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`node_interface_network_tx_dropped`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName` , `InstanceId` , `NodeName`
			Jumlah paket yang akan ditransmisikan tetapi dijatuhkan oleh sebuah antarmuka jaringan pada simpul.

`node_diskio_io_service_bytes_total`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName` , `InstanceId` , `NodeName`
			Jumlah total byte yang ditransfer oleh semua operasi I/O pada simpul.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`node_disk io_io_ser viced_total`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName` , `InstanceId` , `NodeName`
			Jumlah total operasi I/O yang ada di simpul.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`pod_cpu_reserved_capacity`	`PodName`, Ruangnama, `ClusterName`
	
	`ClusterName`
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`
		
		`ClusterName` , `Namespace` , `Service`
			Kapasitas CPU yang dicadangkan untuk setiap pod dalam sebuah klaster.
			
			Rumus: `pod_cpu_request / node_cpu_limit`
			
			Note
			`pod_cpu_request` tidak dilaporkan secara langsung sebagai sebuah metrik, tetapi merupakan sebuah bidang dalam peristiwa log performa. Untuk informasi selengkapnya, lihat Bidang-bidang relevan dalam

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi
			peristiwa log performa untuk Amazon EKS dan Kubernetes.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`pod_cpu_utilization`	`PodName`, Namespace, `ClusterName`
	
	Namespace, `ClusterName`
	
	Layanan, Namespace, `ClusterName`
	
	`ClusterName`
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`	Persentase unit CPU yang sedang digunakan oleh pod.
			
			Rumus: `pod_cpu_usage_total / node_cpu_limit`
			
			Note
			`pod_cpu_usage_total` tidak dilaporkan secara langsung sebagai sebuah metrik, tetapi merupakan sebuah bidang dalam peristiwa log performa. Untuk informasi selengkapnya, lihat Bidang-bidang relevan dalam

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi
			peristiwa log performa untuk Amazon EKS dan Kubernetes.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`pod_cpu_utilization_over_pod_limit`	`PodName`, Namespace, `ClusterName`
	
	Namespace, `ClusterName`
	
	Layanan, Namespace, `ClusterName`
	
	`ClusterName`
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`	Persentase unit CPU yang sedang digunakan oleh pod relatif terhadap batas pod.
			
			Rumus: `pod_cpu_usage_total / pod_cpu_limit`
			
			Note
			`pod_cpu_usage_total` dan `pod_cpu_limit` tidak akan dilaporkan secara langsung sebagai metrik, tetapi sebagai bidang dalam peristiwa log performa. Untuk informasi selengkap

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi
			nya, lihat Bidang-bidang relevan dalam peristiwa log performa untuk Amazon EKS dan Kubernetes.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`pod_memory_reserved_capacity`	`PodName`, `Namespace`, `ClusterName`
	
	`ClusterName`
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`
		
		`ClusterName` , `Namespace` , `Service`
			Persentase memori yang dicadangkan untuk pod.
			
			Rumus: `pod_memory_request / node_memory_limit`
			
			Note
			`pod_memory_request` tidak dilaporkan secara langsung sebagai sebuah metrik, tetapi merupakan sebuah bidang dalam peristiwa log performa. Untuk informasi selengkapnya, lihat Bidang-bidang

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi
			relevan dalam peristiwa log performa untuk Amazon EKS dan Kubernetes.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`pod_memory_utilization`	`PodName`, Namespace, `ClusterName`
	
	Namespace, `ClusterName`
	
	Layanan, Namespace, `ClusterName`
	
	`ClusterName`
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`	Persentase memori yang saat ini sedang digunakan oleh satu pod atau banyak pod.
			
			Rumus: `pod_memory_working_set / node_memory_limit`
			
			Note
			`pod_memory_working_set` tidak dilaporkan secara langsung sebagai sebuah metrik, tetapi merupakan sebuah bidang dalam peristiwa log performa. Untuk informasi selengkap

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi
			nya, lihat Bidang-bidang relevan dalam peristiwa log performa untuk Amazon EKS dan Kubernetes.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`pod_memory_utilization_over_pod_limit`	`PodName`, Namespace, `ClusterName`
	
	Namespace, `ClusterName`
	
	Layanan, Namespace, `ClusterName`
	
	`ClusterName`
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`	Persentase memori yang sedang digunakan oleh pod relatif terhadap batas pod. Jika ada kontainer di dalam pod yang tidak memiliki batas memori yang ditentukan, metrik ini tidak akan ditampilkan.
			
			Rumus: `pod_memory_working_set / pod_memory_limit`
			
			Note
			`pod_memory_working_set` tidak dilaporkan secara langsung sebagai sebuah metrik, tetapi merupakan

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi
			sebuah bidang dalam peristiwa log performa. Untuk informasi selengkap nya, lihat Bidang-bidang relevan dalam peristiwa log performa untuk Amazon EKS dan Kubernetes.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`pod_network_rx_bytes`	`PodName`, `Namespace`, `ClusterName`
	
	`Namespace`, `ClusterName`
	
	Layanan, `Namespace`, `ClusterName`
	
	`ClusterName`
		`ClusterName`, `Namespace`, `PodName`, `FullPodName`	Jumlah byte per detik yang sedang diterima melalui jaringan oleh pod.
			
			Rumus: `sum(pod_interface_network_rx_bytes)`
			
			Note
			`pod_interface_network_rx_bytes` tidak dilaporkan secara langsung sebagai sebuah metrik, tetapi merupakan sebuah bidang dalam peristiwa log performa. Untuk informasi selengkap

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi
			nya, lihat Bidang-bidang relevan dalam peristiwa log performa untuk Amazon EKS dan Kubernetes.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`pod_network_tx_bytes`	`PodName`, `Namespace`, `ClusterName`
	
	`Namespace`, `ClusterName`
	
	Layanan, `Namespace`, `ClusterName`
	
	`ClusterName`
		`ClusterName`, `Namespace`, `PodName`, `FullPodName`	Jumlah byte per detik yang sedang ditransmisikan melalui jaringan oleh pod.
			
			Rumus: `sum(pod_interface_network_tx_bytes)`
			
			Note
			`pod_interface_network_tx_bytes` tidak dilaporkan secara langsung sebagai sebuah metrik, tetapi merupakan sebuah bidang dalam peristiwa log performa. Untuk informasi selengkap

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi
			nya, lihat Bidang-bidang relevan dalam peristiwa log performa untuk Amazon EKS dan Kubernetes.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`pod_cpu_request`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
		
		`Namespace` , `ClusterName` , `Service`
		
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`
			Permintaan CPU untuk pod.
			
			Rumus: `sum(container_cpu_request)`
			
			Note
			`pod_cpu_request` tidak dilaporkan secara langsung sebagai sebuah metrik, tetapi merupakan sebuah bidang dalam peristiwa log performa. Untuk informasi selengkapnya, lihat Bidang-bidang relevan dalam peristiwa log performa

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi
			untuk Amazon EKS dan Kubernetes.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`pod_memory_request`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
		
		`Namespace` , `ClusterName` , `Service`
		
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`
			Permintaan memori untuk pod.
			
			Rumus: `sum(container_memory_request)`
			
			Note
			`pod_memory_request` tidak dilaporkan secara langsung sebagai sebuah metrik, tetapi merupakan sebuah bidang dalam peristiwa log performa. Untuk informasi selengkapnya, lihat Bidang-bidang relevan dalam peristiwa log

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi
			performa untuk Amazon EKS dan Kubernetes.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`pod_cpu_limit`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
		
		`Namespace` , `ClusterName` , `Service`
		
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`
			Batas CPU yang ditentukan untuk kontainer-kontainer yang ada dalam pod. Jika ada kontainer di dalam pod yang tidak memiliki batas CPU yang ditentukan, metrik ini tidak akan ditampilkan.
			
			Rumus: `sum(container_cpu_limit)`
			
			Note
			`pod_cpu_limit` tidak dilaporkan secara langsung sebagai sebuah metrik, tetapi merupakan sebuah bidang dalam peristiwa log performa.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi
			Untuk informasi selengkapnya, lihat Bidang-bidang relevan dalam peristiwa log performa untuk Amazon EKS dan Kubernetes.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`pod_memory_limit`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
		
		`Namespace` , `ClusterName` , `Service`
		
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`
			Batas memori yang ditentukan untuk kontainer-kontainer yang ada dalam pod. Jika ada kontainer di dalam pod yang tidak memiliki batas memori yang ditentukan, metrik ini tidak akan ditampilkan.
			
			Rumus: `sum(container_memory_limit)`
			
			Note
			`pod_cpu_limit` tidak dilaporkan secara langsung sebagai sebuah metrik, tetapi merupakan sebuah bidang dalam peristiwa log performa.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi
			Untuk informasi selengkapnya, lihat Bidang-bidang relevan dalam peristiwa log performa untuk Amazon EKS dan Kubernetes.

`pod_status_failed`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
		
		`Namespace` , `ClusterName` , `Service`
		
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`
			Menunjukkan bahwa semua kontainer yang ada dalam pod telah dihentikan, dan setidaknya satu kontainer telah diakhiri dengan status bukan nol atau dihentikan oleh sistem.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`pod_status_ready`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
		
		`Namespace` , `ClusterName` , `Service`
		
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`
			Menunjukkan bahwa semua kontainer yang ada dalam pod sudah siap, setelah mencapai kondisi `ContainerReady` .

`pod_status_running`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
		
		`Namespace` , `ClusterName` , `Service`
		
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`
			Menunjukkan bahwa semua kontainer yang ada dalam pod sedang berjalan.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`pod_status_scheduled`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
		
		`Namespace` , `ClusterName` , `Service`
		
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`
			Menunjukkan bahwa pod telah dijadwalkan untuk sebuah simpul.

`pod_status_unknown`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
		
		`Namespace` , `ClusterName` , `Service`
		
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`
			Menunjukkan bahwa status pod tidak dapat diperoleh.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`pod_status_pending`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
		
		`Namespace` , `ClusterName` , `Service`
		
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`
			Menunjukkan bahwa pod telah diterima oleh klaster tetapi satu atau beberapa kontainer belum siap.

`pod_status_succeeded`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
		
		`Namespace` , `ClusterName` , `Service`
		
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`
			Menunjukkan bahwa semua kontainer yang ada dalam pod telah berhasil dihentikan dan tidak akan dimulai ulang.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`pod_number_of_containers`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
		
		`Namespace` , `ClusterName` , `Service`
		
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`
			Melaporkan jumlah kontainer yang ditentukan dalam spesifikasi pod.

`pod_number_of_running_containers`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
		
		`Namespace` , `ClusterName` , `Service`
		
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`
			Melaporkan jumlah kontainer yang ada dalam pod yang saat ini berada dalam status `Running`.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`pod_container_status_terminated`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
		
		`Namespace` , `ClusterName` , `Service`
		
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`
			Melaporkan jumlah kontainer yang ada dalam pod yang berada dalam status `Terminated` .

`pod_container_status_running`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
		
		`Namespace` , `ClusterName` , `Service`
		
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`
			Melaporkan jumlah kontainer yang ada dalam pod yang berada dalam status `Running`.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`pod_container_status_waiting`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
		
		`Namespace` , `ClusterName` , `Service`
		
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`
			Melaporkan jumlah kontainer yang ada dalam pod yang berada dalam status `Waiting`.

`pod_interface_network_rx_dropped`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
		
		`Namespace` , `ClusterName` , `Service`
		
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`
			Jumlah paket yang diterima dan kemudian menghapus sebuah antarmuka jaringan untuk pod.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`pod_interface_network_tx_dropped`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
		
		`Namespace` , `ClusterName` , `Service`
		
		`ClusterName` , `Namespace` , `PodName`, `FullPodName`
			Jumlah paket yang seharusnya ditransmisikan tetapi dihapus untuk pod.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`container_cpu_utilization`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName` , `ContainerName`
		
		`PodName`, `Namespace` , `ClusterName` , `ContainerName` , `FullPodName`
			Persentase unit CPU yang sedang digunakan oleh kontainer.
			
			Rumus: `container_cpu_usage_total / node_cpu_limit`
			
			Note
			`container_cpu_utilization` tidak dilaporkan secara langsung sebagai sebuah metrik, tetapi merupakan sebuah bidang dalam peristiwa log performa. Untuk informasi selengkapnya, lihat

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi
			Bidang-bidang relevan dalam peristiwa log performa untuk Amazon EKS dan Kubernetes.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`container_cpu_utilization_over_container_limit`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName` , `ContainerName`
		
		`PodName`, `Namespace` , `ClusterName` , `ContainerName` , `FullPodName`
			Persentase unit CPU yang sedang digunakan oleh kontainer relatif terhadap batas kontainer. Jika kontainer tidak memiliki batas CPU yang ditentukan, maka metrik ini tidak akan ditampilkan.
			
			Rumus: `container_cpu_usage_total / container_cpu_limit`
			
			Note
			`container_cpu_utilization_over_container_limit` tidak dilaporkan secara langsung sebagai

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi
			sebuah metrik, tetapi merupakan sebuah bidang dalam peristiwa log performa. Untuk informasi selengkap nya, lihat Bidang-bidang relevan dalam peristiwa log performa untuk Amazon EKS dan Kubernetes.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`container_memory_utilization`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName` , `ContainerName`
		
		`PodName`, `Namespace` , `ClusterName` , `ContainerName` , `FullPodName`
			Persentase unit memori yang sedang digunakan oleh kontainer.
			
			Rumus: `container_memory_working_set / node_memory_limit`
			
			Note
			`container_memory_utilization` tidak dilaporkan secara langsung sebagai sebuah metrik, tetapi merupakan sebuah bidang dalam peristiwa log performa. Untuk informasi

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi
			selengkap nya, lihat Bidang-bidang relevan dalam peristiwa log performa untuk Amazon EKS dan Kubernetes.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`container_memory_utilization_over_container_limit`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName` , `ContainerName`
		
		`PodName`, `Namespace` , `ClusterName` , `ContainerName` , `FullPodName`
			Persentase unit memori yang sedang digunakan oleh kontainer relatif terhadap batas kontainer. Jika kontainer tidak memiliki batas memori yang ditentukan, maka metrik ini tidak akan ditampilkan.
			
			Rumus: `container_memory_working_set / container_memory_limit`
			
			Note
			`container_memory_utilization_over_container_limit` tidak dilaporkan secara langsung sebagai

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi
			sebuah metrik, tetapi merupakan sebuah bidang dalam peristiwa log performa. Untuk informasi selengkap nya, lihat Bidang-bidang relevan dalam peristiwa log performa untuk Amazon EKS dan Kubernetes.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`container_memory_failures_total`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName` , `ContainerName`
		
		`PodName`, `Namespace` , `ClusterName` , `ContainerName` , `FullPodName`
			Jumlah kegagalan alokasi memori yang dialami oleh kontainer .

`container_filesystem_usage`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName` , `ContainerName`
		
		`PodName`, `Namespace` , `ClusterName` , `ContainerName` , `FullPodName`
			Jumlah byte yang dikonsumsi oleh kontainer pada sistem file ini.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`container_filesystem_available`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName` , `ContainerName`
		
		`PodName`, `Namespace` , `ClusterName` , `ContainerName` , `FullPodName`
			Jumlah byte yang tersedia untuk kontainer pada sistem file ini.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`container_filesystem_utilization`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName` , `ContainerName`
		
		`PodName`, `Namespace` , `ClusterName` , `ContainerName` , `FullPodName`
			Persentase sistem file yang sedang dikonsumsi oleh kontainer.
			
			Rumus: `container_memory_working_set / container_memory_limit`
			
			Note
			`container_filesystem_usage / container_filesystem_capacity` tidak dilaporkan secara langsung sebagai sebuah metrik, tetapi merupakan sebuah bidang dalam peristiwa log

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi
			performa. Untuk informasi selengkapnya, lihat Bidang-bidang relevan dalam peristiwa log performa untuk Amazon EKS dan Kubernetes.
pod_number_of_container_restarts	PodName, Namespace , ClusterName		Jumlah total kontainer yang memulai ulang di sebuah pod.
service_number_of_running_pods	Layanan, Namespace , ClusterName ClusterName		Jumlah pod yang menjalankan satu layanan atau banyak layanan di klaster.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`replicas_desired`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
			Jumlah pod yang dikehendaki untuk beban kerja sebagaimana yang ditentukan dalam spesifikasi beban kerja.

`replicas_ready`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
			Jumlah pod untuk sebuah beban kerja yang telah mencapai status siap.

`replicas_available`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
			Jumlah pod untuk sebuah beban kerja yang tersedia. Sebuah pod yang tersedia ketika sudah siap untuk `minReadySeconds` yang ditentukan dalam spesifikasi beban kerja.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`replicas_unavailable`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`PodName`, `Namespace` , `ClusterName`
			Jumlah pod untuk sebuah beban kerja yang tidak tersedia. Sebuah pod yang tersedia ketika sudah siap untuk `minReadySeconds` yang ditentukan dalam spesifikasi beban kerja. Pod tidak tersedia jika pod tersebut belum memenuhi kriteria ini.

`apiserver_storage_objects`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName` , `resource`
			Jumlah objek yang disimpan di etcd pada saat pemeriksaan terakhir.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`apiserver_request_total`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName` , `code`, `verb`
			Jumlah total permintaan API yang dikirimkan ke server API Kubernetes.

`apiserver_request_duration_seconds`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName` , `verb`
			Latensi respons untuk permintaan API ke server API Kubernetes.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`apiserver_admission_controller_admission_duration_seconds`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName , operation`
			Latensi pengendali penerimaan, dalam satuan detik. Pengendali penerimaan adalah kode yang mencegat permintaan ke server API Kubernetes.

`rest_client_request_duration_seconds`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName , operation`
			Latensi respons yang dialami oleh klien yang memanggil server API Kubernetes. Metrik ini bersifat eksperimental dan dapat berubah pada rilis Kubernetes di masa mendatang.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`rest_client_requests_total`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName` , `code`, `method`
			Jumlah total permintaan API yang dikirimkan ke server API Kubernetes yang dibuat oleh klien. Metrik ini bersifat eksperimental dan dapat berubah pada rilis Kubernetes di masa mendatang.

`etcd_request_duration_seconds`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName` , `operation`
			Latensi respons panggilan API ke Etcd. Metrik ini bersifat eksperimental dan dapat berubah pada rilis Kubernetes di masa mendatang.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`apiserver_storage_size_bytes`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName , endpoint`
			Ukuran file basis data penyimpanan yang dialokasikan secara fisik, dalam satuan byte. Metrik ini bersifat eksperimental dan dapat berubah pada rilis Kubernetes di masa mendatang.

`apiserver_longrunning_requests`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName , resource`
			Jumlah permintaan aktif yang berjalan lama yang dikirimkan ke server API Kubernetes.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`apiserver_current_inflight_requests`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName`, `request_kind`
			Jumlah permintaan yang sedang diproses oleh server API Kubernetes.

`apiserver_admission_webhook_admission_duration_seconds`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName`, `name`
			Latensi webhook penerimaan, dalam satuan detik. Webhook penerimaan adalah callback HTTP yang menerima permintaan penerimaan dan melakukan sesuatu dengannya.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`apiserver_admission_step_admission_duration_seconds`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName , operation`
			Latensi sub-langkah penerimaan, dalam satuan detik.

`apiserver_requested_deprecated_apis`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName , group`
			Jumlah permintaan yang dikirimkan ke API yang tidak digunakan lagi di server API Kubernetes.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`apiserver_request_total_5XX`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName`, `code`, `verb`
			Jumlah permintaan yang dikirimkan ke server API Kubernetes yang direspon dengan kode respons HTTP 5XX.

`apiserver_storage_list_duration_seconds`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName`, `resource`
			Latensi respons objek daftar dari Etcd. Metrik ini bersifat eksperimental dan dapat berubah pada rilis Kubernetes di masa mendatang.

Nama metrik	Dimensi-dimensi dengan versi Wawasan Kontainer apa pun	Dimensi tambahan dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	Deskripsi

`apiserver_current_inqueue_requests`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName , request_kind`
			Jumlah permintaan yang diantrekan oleh server API Kubernetes. Metrik ini bersifat eksperimental dan dapat berubah pada rilis Kubernetes di masa mendatang.

`apiserver_flowcontrol_rejected_requests_total`
 	
Metrik ini hanya tersedia dengan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS
		`ClusterName`
		
		`ClusterName , reason`
			Jumlah permintaan yang ditolak oleh subsistem API Priority dan Fairness. Metrik ini bersifat eksperimental dan dapat berubah pada rilis Kubernetes di masa mendatang.

Metrik GPU NVIDIA

Dimulai dengan CloudWatch agen versi 1.300034.0, Container Insights dengan peningkatan observabilitas untuk Amazon EKS mengumpulkan metrik GPU NVIDIA dari beban kerja EKS secara default. CloudWatch Agen harus diinstal menggunakan add-on CloudWatch Observability EKS versi

v1.3.0-eksbuild.1 atau yang lebih baru. Untuk informasi selengkapnya, lihat Instal CloudWatch agen dengan menggunakan add-on Amazon CloudWatch Observability EKS. Metrik GPU NVIDIA yang dikumpulkan tercantum dalam tabel di bagian ini.

Agar Container Insights mengumpulkan metrik GPU NVIDIA, Anda harus memenuhi prasyarat berikut:

- Anda harus menggunakan Container Insights dengan peningkatan observabilitas untuk Amazon EKS, dengan add-on Amazon CloudWatch Observability EKS.
- Plugin perangkat NVIDIA untuk Kubernetes harus diinstal di cluster.
- Toolkit kontainer NVIDIA harus diinstal pada node cluster. Misalnya, AMI akselerasi Amazon EKS yang dioptimalkan dibangun dengan komponen yang diperlukan.

Anda dapat memilih untuk tidak mengumpulkan metrik GPU NVIDIA dengan menyetel `accelerated_compute_metrics` opsi di file konfigurasi CloudWatch agen awal ke. `false` Untuk informasi selengkapnya dan contoh konfigurasi opt-out, lihat. (Opsional) Konfigurasi tambahan

Nama metrik	Dimensi	Deskripsi

`container_gpu_memory_total`	`ClusterName`
	
	`ClusterName` , `Namespace` , `PodName`, `ContainerName`
	
	`ClusterName` , `Namespace` , `PodName`, `FullPodName` , `ContainerName`
	
	`ClusterName` , `Namespace` , `PodName`, `FullPodName` , `ContainerName` , `GpuDevice`
		Ukuran buffer frame total, dalam byte, pada GPU yang dialokasikan ke wadah.

`container_gpu_memory_used`	`ClusterName`
	
	`ClusterName` , `Namespace` , `PodName`, `ContainerName`
		Byte frame buffer yang digunakan pada GPU (s) dialokasikan ke container.

Nama metrik	Dimensi	Deskripsi

	`ClusterName` , `Namespace` , `PodName`, `FullPodName` , `ContainerName`
	
	`ClusterName` , `Namespace` , `PodName`, `FullPodName` , `ContainerName` , `GpuDevice`	

`container_gpu_memory_utilization`	`ClusterName`
	
	`ClusterName` , `Namespace` , `PodName`, `ContainerName`
	
	`ClusterName` , `Namespace` , `PodName`, `FullPodName` , `ContainerName`
	
	`ClusterName` , `Namespace` , `PodName`, `FullPodName` , `ContainerName` , `GpuDevice`
		Persentase buffer bingkai yang digunakan dari GPU yang dialokasikan ke wadah.

`container_gpu_power_draw`	`ClusterName`
	
	`ClusterName` , `Namespace` , `PodName`, `ContainerName`
	
	`ClusterName` , `Namespace` , `PodName`, `FullPodName` , `ContainerName`
	
	`ClusterName` , `Namespace` , `PodName`, `FullPodName` , `ContainerName` , `GpuDevice`
		Penggunaan daya dalam watt GPU dialokasikan ke wadah.

Nama metrik	Dimensi	Deskripsi
container_gpu_temperature	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , GpuDevice	Suhu dalam derajat celcius dari GPU (s) dialokasikan ke wadah.
container_gpu_utilization	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , GpuDevice	Persentase pemanfaatan GPU yang dialokasikan ke kontainer.
node_gpu_memory_total	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, GpuDevice	Ukuran buffer frame total, dalam byte, pada GPU yang dialokasikan ke node.

Nama metrik	Dimensi	Deskripsi
node_gpu_memory_used	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, GpuDevice	Byte frame buffer yang digunakan pada GPU yang dialokasikan ke node.
node_gpu_memory_utilization	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, GpuDevice	Persentase frame buffer yang digunakan pada GPU yang dialokasikan ke node.
node_gpu_power_draw	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, GpuDevice	Penggunaan daya dalam watt GPU dialokasikan ke node.
node_gpu_temperature	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, GpuDevice	Suhu dalam derajat celcius dari GPU (s) dialokasikan ke node.

Nama metrik	Dimensi	Deskripsi
node_gpu_utilization	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, GpuDevice	Persentase pemanfaatan GPU yang dialokasikan ke node.
pod_gpu_memory_total	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName . GpuDevice	Ukuran buffer frame total, dalam byte, pada GPU yang dialokasikan ke pod.

Nama metrik	Dimensi	Deskripsi
pod_gpu_memory_used	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName . GpuDevice	Byte frame buffer yang digunakan pada GPU yang dialokasikan ke pod.
pod_gpu_memory_utilization	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName . GpuDevice	Persentase frame buffer yang digunakan dari GPU yang dialokasikan ke pod.

Nama metrik	Dimensi	Deskripsi
pod_gpu_power_draw	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName . GpuDevice	Penggunaan daya dalam watt GPU dialokasikan ke pod.
pod_gpu_temperature	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName . GpuDevice	Suhu dalam derajat celcius dari GPU (s) dialokasikan ke pod.

Nama metrik	Dimensi	Deskripsi
pod_gpu_utilization	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , GpuDevice	Persentase pemanfaatan GPU yang dialokasikan ke pod.

Referensi log performa Wawasan Kontainer

Bagian ini mencakup informasi referensi tentang bagaimana Wawasan Kontainer menggunakan peristiwa log peforma untuk mengumpulkan metrik. Saat Anda menerapkan Wawasan Kontainer, ia akan secara otomatis membuat suatu grup log untuk peristiwa log performa. Anda tidak harus membuat grup log ini sendiri.

Topik

- Peristiwa log performa Wawasan Kontainer untuk Amazon ECS
- Peristiwa log performa Wawasan Kontainer untuk Amazon EKS dan Kubernetes
- Bidang-bidang relevan dalam peristiwa log performa untuk Amazon EKS dan Kubernetes

Peristiwa log performa Wawasan Kontainer untuk Amazon ECS

Berikut ini adalah contoh-contoh peristiwa log performa yang dikumpulkan oleh Wawasan Kontainer dari Amazon ECS.

Log ini ada di CloudWatch Log, dalam grup log bernama`/aws/ecs/containerinsights/`*`CLUSTER_NAME`*`/performance`. Dalam grup log itu, masing-masing instans kontainer akan memiliki sebuah log stream bernama `AgentTelemetry-`*`CONTAINER_INSTANCE_ID`*.

Anda dapat menjalankan kueri atas log ini dengan menggunakan kueri seperti `{ $.Type = "Container" }` untuk menampilkan semua peristiwa log kontainer.

Tipe: Kontainer


```
{
 "Version":"0",
 "Type":"Container",
 "ContainerName":"sleep",
 "TaskId":"7ac4dfba69214411b4783a3b8189c9ba",
 "TaskDefinitionFamily":"sleep360",
 "TaskDefinitionRevision":"1",
 "ContainerInstanceId":"0d7650e6dec34c1a9200f72098071e8f",
 "EC2InstanceId":"i-0c470579dbcdbd2f3",
 "ClusterName":"MyCluster",
 "Image":"busybox",
 "ContainerKnownStatus":"RUNNING",
 "Timestamp":1623963900000,
 "CpuUtilized":0.0,
 "CpuReserved":10.0,
 "MemoryUtilized":0,
 "MemoryReserved":10,
 "StorageReadBytes":0,
 "StorageWriteBytes":0,
 "NetworkRxBytes":0,
 "NetworkRxDropped":0,
 "NetworkRxErrors":0,
 "NetworkRxPackets":14,
 "NetworkTxBytes":0,
 "NetworkTxDropped":0,
 "NetworkTxErrors":0,
 "NetworkTxPackets":0
}
```


Tipe: Tugas


```
{
    "Version": "0",
    "Type": "Task",
    "TaskId": "7ac4dfba69214411b4783a3b8189c9ba",
    "TaskDefinitionFamily": "sleep360",
    "TaskDefinitionRevision": "1",
    "ContainerInstanceId": "0d7650e6dec34c1a9200f72098071e8f",
```

```
    "EC2InstanceId": "i-0c470579dbcdbd2f3",
    "ClusterName": "MyCluster",
    "AccountID": "637146863587",
    "Region": "us-west-2",
    "AvailabilityZone": "us-west-2b",
    "KnownStatus": "RUNNING",
    "LaunchType": "EC2",
    "PullStartedAt": 1623963608201,
    "PullStoppedAt": 1623963610065,
    "CreatedAt": 1623963607094,
    "StartedAt": 1623963610382,
    "Timestamp": 1623963900000,
    "CpuUtilized": 0.0,
    "CpuReserved": 10.0,
    "MemoryUtilized": 0,
    "MemoryReserved": 10,
    "StorageReadBytes": 0,
    "StorageWriteBytes": 0,
    "NetworkRxBytes": 0,
    "NetworkRxDropped": 0,
    "NetworkRxErrors": 0,
    "NetworkRxPackets": 14,
    "NetworkTxBytes": 0,
    "NetworkTxDropped": 0,
    "NetworkTxErrors": 0,
    "NetworkTxPackets": 0,
    "EBSFilesystemUtilized": 10,
    "EBSFilesystemSize": 20,
    "CloudWatchMetrics": [
        {
            "Namespace": "ECS/ContainerInsights",
            "Metrics": [
                {
                    "Name": "CpuUtilized",
                    "Unit": "None"
                },
                {
                    "Name": "CpuReserved",
                    "Unit": "None"
                },
                {
                    "Name": "MemoryUtilized",
                    "Unit": "Megabytes"
                },
```

```
                {
                    "Name": "MemoryReserved",
                    "Unit": "Megabytes"
                },
                {
                    "Name": "StorageReadBytes",
                    "Unit": "Bytes/Second"
                },
                {
                    "Name": "StorageWriteBytes",
                    "Unit": "Bytes/Second"
                },
                {
                    "Name": "NetworkRxBytes",
                    "Unit": "Bytes/Second"
                },
                {
                    "Name": "NetworkTxBytes",
                    "Unit": "Bytes/Second"
                },
                {
                    "Name": "EBSFilesystemSize",
                    "Unit": "Gigabytes"
                },
                {
                    "Name": "EBSFilesystemUtilzed",
                    "Unit": "Gigabytes"
                }
            ],
            "Dimensions": [
                ["ClusterName"],
                [
                    "ClusterName",
                    "TaskDefinitionFamily"
                ]
            ]
        }
    ]
}
```

Tipe: Layanan

```
{
```

```
    "Version": "0",
    "Type": "Service",
    "ServiceName": "myCIService",
    "ClusterName": "myCICluster",
    "Timestamp": 1561586460000,
    "DesiredTaskCount": 2,
    "RunningTaskCount": 2,
    "PendingTaskCount": 0,
    "DeploymentCount": 1,
    "TaskSetCount": 0,
    "CloudWatchMetrics": [
        {
            "Namespace": "ECS/ContainerInsights",
            "Metrics": [
                {
                    "Name": "DesiredTaskCount",
                    "Unit": "Count"
                },
                {
                    "Name": "RunningTaskCount",
                    "Unit": "Count"
                },
                {
                    "Name": "PendingTaskCount",
                    "Unit": "Count"
                },
                {
                    "Name": "DeploymentCount",
                    "Unit": "Count"
                },
                {
                    "Name": "TaskSetCount",
                    "Unit": "Count"
                }
            ],
            "Dimensions": [
                [
                    "ServiceName",
                    "ClusterName"
                ]
            ]
        }
    ]
```

```
}
```

Jenis: Volume

```
{
    "Version": "0",
    "Type": "Volume",
    "TaskDefinitionFamily": "myCITaskDef",
    "TaskId": "7ac4dfba69214411b4783a3b8189c9ba",
    "ClusterName": "myCICluster",
    "ServiceName": "myCIService",
    "VolumeId": "vol-1233436545ff708cb",
    "InstanceId": "i-0c470579dbcdbd2f3",
    "LaunchType": "EC2",
    "VolumeName": "MyVolumeName",
    "EBSFilesystemUtilized": 10,
    "EBSFilesystemSize": 20,
    "CloudWatchMetrics": [
        {
            "Namespace": "ECS/ContainerInsights",
            "Metrics": [
                {
                    "Name": "EBSFilesystemSize",
                    "Unit": "Gigabytes"
                },
                {
                    "Name": "EBSFilesystemUtilzed",
                    "Unit": "Gigabytes"
                }
            ],
            "Dimensions": [
                ["ClusterName"],
                [
                    "VolumeName",
                    "TaskDefinitionFamily",
                    "ClusterName"
                ],
                [
                    "ServiceName",
                    "ClusterName"
                ]
            ]
        }
```

```
    ]
}
```

Tipe: Klaster

```
{
    "Version": "0",
    "Type": "Cluster",
    "ClusterName": "myCICluster",
    "Timestamp": 1561587300000,
    "TaskCount": 5,
    "ContainerInstanceCount": 5,
    "ServiceCount": 2,
    "CloudWatchMetrics": [
        {
            "Namespace": "ECS/ContainerInsights",
            "Metrics": [
                {
                    "Name": "TaskCount",
                    "Unit": "Count"
                },
                {
                    "Name": "ContainerInstanceCount",
                    "Unit": "Count"
                },
                {
                    "Name": "ServiceCount",
                    "Unit": "Count"
                }
            ],
            "Dimensions": [
                [
                    "ClusterName"
                ]
            ]
        }
    ]
}
```

Peristiwa log performa Wawasan Kontainer untuk Amazon EKS dan Kubernetes

Berikut ini adalah contoh-contoh peristiwa log performa yang dikumpulkan oleh Wawasan Kontainer dari klaster Amazon EKS dan Kubernetes.

Tipe: Simpul

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Percent",
          "Name": "node_cpu_utilization"
        },
        {
          "Unit": "Percent",
          "Name": "node_memory_utilization"
        },
        {
          "Unit": "Bytes/Second",
          "Name": "node_network_total_bytes"
        },
        {
          "Unit": "Percent",
          "Name": "node_cpu_reserved_capacity"
        },
        {
          "Unit": "Percent",
          "Name": "node_memory_reserved_capacity"
        },
        {
          "Unit": "Count",
          "Name": "node_number_of_running_pods"
        },
        {
          "Unit": "Count",
          "Name": "node_number_of_running_containers"
        }
      ],
      "Dimensions": [
        [
          "NodeName",
          "InstanceId",
          "ClusterName"
        ]
      ],
```

```
      "Namespace": "ContainerInsights"
    },
    {
      "Metrics": [
        {
          "Unit": "Percent",
          "Name": "node_cpu_utilization"
        },
        {
          "Unit": "Percent",
          "Name": "node_memory_utilization"
        },
        {
          "Unit": "Bytes/Second",
          "Name": "node_network_total_bytes"
        },
        {
          "Unit": "Percent",
          "Name": "node_cpu_reserved_capacity"
        },
        {
          "Unit": "Percent",
          "Name": "node_memory_reserved_capacity"
        },
        {
          "Unit": "Count",
          "Name": "node_number_of_running_pods"
        },
        {
          "Unit": "Count",
          "Name": "node_number_of_running_containers"
        },
        {
          "Name": "node_cpu_usage_total"
        },
        {
          "Name": "node_cpu_limit"
        },
        {
          "Unit": "Bytes",
          "Name": "node_memory_working_set"
        },
        {
          "Unit": "Bytes",
```

```
          "Name": "node_memory_limit"
        }
      ],
      "Dimensions": [
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    }
  ],
  "ClusterName": "myCICluster",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "Sources": [
    "cadvisor",
    "/proc",
    "pod",
    "calculated"
  ],
  "Timestamp": "1567096682364",
  "Type": "Node",
  "Version": "0",
  "kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal"
  },
  "node_cpu_limit": 4000,
  "node_cpu_request": 1130,
  "node_cpu_reserved_capacity": 28.249999999999996,
  "node_cpu_usage_system": 33.794636630852764,
  "node_cpu_usage_total": 136.47852169244098,
  "node_cpu_usage_user": 71.67075111567326,
  "node_cpu_utilization": 3.4119630423110245,
  "node_memory_cache": 3103297536,
  "node_memory_failcnt": 0,
  "node_memory_hierarchical_pgfault": 0,
  "node_memory_hierarchical_pgmajfault": 0,
  "node_memory_limit": 16624865280,
  "node_memory_mapped_file": 406646784,
  "node_memory_max_usage": 4230746112,
  "node_memory_pgfault": 0,
  "node_memory_pgmajfault": 0,
  "node_memory_request": 1115684864,
```

```
  "node_memory_reserved_capacity": 6.7109407818311055,
  "node_memory_rss": 798146560,
  "node_memory_swap": 0,
  "node_memory_usage": 3901444096,
  "node_memory_utilization": 6.601302600149552,
  "node_memory_working_set": 1097457664,
  "node_network_rx_bytes": 35918.392817386324,
  "node_network_rx_dropped": 0,
  "node_network_rx_errors": 0,
  "node_network_rx_packets": 157.67565245448117,
  "node_network_total_bytes": 68264.20276554905,
  "node_network_tx_bytes": 32345.80994816272,
  "node_network_tx_dropped": 0,
  "node_network_tx_errors": 0,
  "node_network_tx_packets": 154.21455923431654,
  "node_number_of_running_containers": 16,
  "node_number_of_running_pods": 13
}
```

Tipe: NodeFS

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Percent",
          "Name": "node_filesystem_utilization"
        }
      ],
      "Dimensions": [
        [
          "NodeName",
          "InstanceId",
          "ClusterName"
        ],
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
```

```
    }
  ],
  "ClusterName": "myCICluster",
  "EBSVolumeId": "aws://us-west-2b/vol-0a53108976d4a2fda",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "Sources": [
    "cadvisor",
    "calculated"
  ],
  "Timestamp": "1567097939726",
  "Type": "NodeFS",
  "Version": "0",
  "device": "/dev/nvme0n1p1",
  "fstype": "vfs",
  "kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal"
  },
  "node_filesystem_available": 17298395136,
  "node_filesystem_capacity": 21462233088,
  "node_filesystem_inodes": 10484720,
  "node_filesystem_inodes_free": 10367158,
  "node_filesystem_usage": 4163837952,
  "node_filesystem_utilization": 19.400767547940255
}
```

Jenis: NodeDisk IO

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "ClusterName": "myCICluster",
  "EBSVolumeId": "aws://us-west-2b/vol-0a53108976d4a2fda",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "Sources": [
    "cadvisor"
  ],
  "Timestamp": "1567096928131",
  "Type": "NodeDiskIO",
  "Version": "0",
```

```
  "device": "/dev/nvme0n1",
  "kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal"
  },
  "node_diskio_io_service_bytes_async": 9750.505814277016,
  "node_diskio_io_service_bytes_read": 0,
  "node_diskio_io_service_bytes_sync": 230.6174506688036,
  "node_diskio_io_service_bytes_total": 9981.123264945818,
  "node_diskio_io_service_bytes_write": 9981.123264945818,
  "node_diskio_io_serviced_async": 1.153087253344018,
  "node_diskio_io_serviced_read": 0,
  "node_diskio_io_serviced_sync": 0.03603397666700056,
  "node_diskio_io_serviced_total": 1.1891212300110185,
  "node_diskio_io_serviced_write": 1.1891212300110185
}
```


Jenis: NodeNet


```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "ClusterName": "myCICluster",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "Sources": [
    "cadvisor",
    "calculated"
  ],
  "Timestamp": "1567096928131",
  "Type": "NodeNet",
  "Version": "0",
  "interface": "eni972f6bfa9a0",
  "kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal"
  },
  "node_interface_network_rx_bytes": 3163.008420864309,
  "node_interface_network_rx_dropped": 0,
  "node_interface_network_rx_errors": 0,
  "node_interface_network_rx_packets": 16.575629266820258,
  "node_interface_network_total_bytes": 3518.3935157426017,
  "node_interface_network_tx_bytes": 355.385094878293,
  "node_interface_network_tx_dropped": 0,
```

```
  "node_interface_network_tx_errors": 0,
  "node_interface_network_tx_packets": 3.9997714100370625
}
```

Tipe: Pod

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Percent",
          "Name": "pod_cpu_utilization"
        },
        {
          "Unit": "Percent",
          "Name": "pod_memory_utilization"
        },
        {
          "Unit": "Bytes/Second",
          "Name": "pod_network_rx_bytes"
        },
        {
          "Unit": "Bytes/Second",
          "Name": "pod_network_tx_bytes"
        },
        {
          "Unit": "Percent",
          "Name": "pod_cpu_utilization_over_pod_limit"
        },
        {
          "Unit": "Percent",
          "Name": "pod_memory_utilization_over_pod_limit"
        }
      ],
      "Dimensions": [
        [
          "PodName",
          "Namespace",
          "ClusterName"
        ],
```

```
        [
          "Service",
          "Namespace",
          "ClusterName"
        ],
        [
          "Namespace",
          "ClusterName"
        ],
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    },
    {
      "Metrics": [
        {
          "Unit": "Percent",
          "Name": "pod_cpu_reserved_capacity"
        },
        {
          "Unit": "Percent",
          "Name": "pod_memory_reserved_capacity"
        }
      ],
      "Dimensions": [
        [
          "PodName",
          "Namespace",
          "ClusterName"
        ],
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    },
    {
      "Metrics": [
        {
          "Unit": "Count",
          "Name": "pod_number_of_container_restarts"
        }
```

```
      ],
      "Dimensions": [
        [
          "PodName",
          "Namespace",
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    }
  ],
  "ClusterName": "myCICluster",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "Namespace": "amazon-cloudwatch",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "PodName": "cloudwatch-agent-statsd",
  "Service": "cloudwatch-agent-statsd",
  "Sources": [
    "cadvisor",
    "pod",
    "calculated"
  ],
  "Timestamp": "1567097351092",
  "Type": "Pod",
  "Version": "0",
  "kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal",
    "labels": {
      "app": "cloudwatch-agent-statsd",
      "pod-template-hash": "df44f855f"
    },
    "namespace_name": "amazon-cloudwatch",
    "pod_id": "2f4ff5ac-c813-11e9-a31d-06e9dde32928",
    "pod_name": "cloudwatch-agent-statsd-df44f855f-ts4q2",
    "pod_owners": [
      {
        "owner_kind": "Deployment",
        "owner_name": "cloudwatch-agent-statsd"
      }
    ],
    "service_name": "cloudwatch-agent-statsd"
  },
  "pod_cpu_limit": 200,
```

```
  "pod_cpu_request": 200,
  "pod_cpu_reserved_capacity": 5,
  "pod_cpu_usage_system": 1.4504841104992765,
  "pod_cpu_usage_total": 5.817016867430125,
  "pod_cpu_usage_user": 1.1281543081661038,
  "pod_cpu_utilization": 0.14542542168575312,
  "pod_cpu_utilization_over_pod_limit": 2.9085084337150624,
  "pod_memory_cache": 8192,
  "pod_memory_failcnt": 0,
  "pod_memory_hierarchical_pgfault": 0,
  "pod_memory_hierarchical_pgmajfault": 0,
  "pod_memory_limit": 104857600,
  "pod_memory_mapped_file": 0,
  "pod_memory_max_usage": 25268224,
  "pod_memory_pgfault": 0,
  "pod_memory_pgmajfault": 0,
  "pod_memory_request": 104857600,
  "pod_memory_reserved_capacity": 0.6307275170893897,
  "pod_memory_rss": 22777856,
  "pod_memory_swap": 0,
  "pod_memory_usage": 25141248,
  "pod_memory_utilization": 0.10988455961791709,
  "pod_memory_utilization_over_pod_limit": 17.421875,
  "pod_memory_working_set": 18268160,
  "pod_network_rx_bytes": 9880.697124714186,
  "pod_network_rx_dropped": 0,
  "pod_network_rx_errors": 0,
  "pod_network_rx_packets": 107.80005532263283,
  "pod_network_total_bytes": 10158.829201483635,
  "pod_network_tx_bytes": 278.13207676944796,
  "pod_network_tx_dropped": 0,
  "pod_network_tx_errors": 0,
  "pod_network_tx_packets": 1.146027574644318,
  "pod_number_of_container_restarts": 0,
  "pod_number_of_containers": 1,
  "pod_number_of_running_containers": 1,
  "pod_status": "Running"
}
```


Jenis: PodNet


```
{
```

```
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "ClusterName": "myCICluster",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "Namespace": "amazon-cloudwatch",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "PodName": "cloudwatch-agent-statsd",
  "Service": "cloudwatch-agent-statsd",
  "Sources": [
    "cadvisor",
    "calculated"
  ],
  "Timestamp": "1567097351092",
  "Type": "PodNet",
  "Version": "0",
  "interface": "eth0",
  "kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal",
    "labels": {
      "app": "cloudwatch-agent-statsd",
      "pod-template-hash": "df44f855f"
    },
    "namespace_name": "amazon-cloudwatch",
    "pod_id": "2f4ff5ac-c813-11e9-a31d-06e9dde32928",
    "pod_name": "cloudwatch-agent-statsd-df44f855f-ts4q2",
    "pod_owners": [
      {
        "owner_kind": "Deployment",
        "owner_name": "cloudwatch-agent-statsd"
      }
    ],
    "service_name": "cloudwatch-agent-statsd"
  },
  "pod_interface_network_rx_bytes": 9880.697124714186,
  "pod_interface_network_rx_dropped": 0,
  "pod_interface_network_rx_errors": 0,
  "pod_interface_network_rx_packets": 107.80005532263283,
  "pod_interface_network_total_bytes": 10158.829201483635,
  "pod_interface_network_tx_bytes": 278.13207676944796,
  "pod_interface_network_tx_dropped": 0,
  "pod_interface_network_tx_errors": 0,
  "pod_interface_network_tx_packets": 1.146027574644318
```

```
}
```

Tipe: Kontainer


```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-NodeGroup-
sample",
  "ClusterName": "myCICluster",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "Namespace": "amazon-cloudwatch",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "PodName": "cloudwatch-agent-statsd",
  "Service": "cloudwatch-agent-statsd",
  "Sources": [
    "cadvisor",
    "pod",
    "calculated"
  ],
  "Timestamp": "1567097399912",
  "Type": "Container",
  "Version": "0",
  "container_cpu_limit": 200,
  "container_cpu_request": 200,
  "container_cpu_usage_system": 1.87958283771964,
  "container_cpu_usage_total": 6.159993652997942,
  "container_cpu_usage_user": 1.6707403001952357,
  "container_cpu_utilization": 0.15399984132494854,
  "container_memory_cache": 8192,
  "container_memory_failcnt": 0,
  "container_memory_hierarchical_pgfault": 0,
  "container_memory_hierarchical_pgmajfault": 0,
  "container_memory_limit": 104857600,
  "container_memory_mapped_file": 0,
  "container_memory_max_usage": 24580096,
  "container_memory_pgfault": 0,
  "container_memory_pgmajfault": 0,
  "container_memory_request": 104857600,
  "container_memory_rss": 22736896,
  "container_memory_swap": 0,
  "container_memory_usage": 24453120,
  "container_memory_utilization": 0.10574541028701798,
  "container_memory_working_set": 17580032,
```

```
  "container_status": "Running",
  "kubernetes": {
    "container_name": "cloudwatch-agent",
    "docker": {
      "container_id":
 "8967b6b37da239dfad197c9fdea3e5dfd35a8a759ec86e2e4c3f7b401e232706"
    },
    "host": "ip-192-168-75-26.us-west-2.compute.internal",
    "labels": {
      "app": "cloudwatch-agent-statsd",
      "pod-template-hash": "df44f855f"
    },
    "namespace_name": "amazon-cloudwatch",
    "pod_id": "2f4ff5ac-c813-11e9-a31d-06e9dde32928",
    "pod_name": "cloudwatch-agent-statsd-df44f855f-ts4q2",
    "pod_owners": [
      {
        "owner_kind": "Deployment",
        "owner_name": "cloudwatch-agent-statsd"
      }
    ],
    "service_name": "cloudwatch-agent-statsd"
  },
  "number_of_container_restarts": 0
}
```

Tipe: ContainerFS

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "ClusterName": "myCICluster",
  "EBSVolumeId": "aws://us-west-2b/vol-0a53108976d4a2fda",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "Namespace": "amazon-cloudwatch",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "PodName": "cloudwatch-agent-statsd",
  "Service": "cloudwatch-agent-statsd",
  "Sources": [
    "cadvisor",
    "calculated"
  ],
```

```
  "Timestamp": "1567097399912",
  "Type": "ContainerFS",
  "Version": "0",
  "container_filesystem_available": 0,
  "container_filesystem_capacity": 21462233088,
  "container_filesystem_usage": 24576,
  "container_filesystem_utilization": 0.0001145081217748071,
  "device": "/dev/nvme0n1p1",
  "fstype": "vfs",
  "kubernetes": {
    "container_name": "cloudwatch-agent",
    "docker": {
      "container_id":
 "8967b6b37da239dfad197c9fdea3e5dfd35a8a759ec86e2e4c3f7b401e232706"
    },
    "host": "ip-192-168-75-26.us-west-2.compute.internal",
    "labels": {
      "app": "cloudwatch-agent-statsd",
      "pod-template-hash": "df44f855f"
    },
    "namespace_name": "amazon-cloudwatch",
    "pod_id": "2f4ff5ac-c813-11e9-a31d-06e9dde32928",
    "pod_name": "cloudwatch-agent-statsd-df44f855f-ts4q2",
    "pod_owners": [
      {
        "owner_kind": "Deployment",
        "owner_name": "cloudwatch-agent-statsd"
      }
    ],
    "service_name": "cloudwatch-agent-statsd"
  }
}
```

Tipe: Klaster

```
{
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Count",
          "Name": "cluster_node_count"
        },
```

```
        {
          "Unit": "Count",
          "Name": "cluster_failed_node_count"
        }
      ],
      "Dimensions": [
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    }
  ],
  "ClusterName": "myCICluster",
  "Sources": [
    "apiserver"
  ],
  "Timestamp": "1567097534160",
  "Type": "Cluster",
  "Version": "0",
  "cluster_failed_node_count": 0,
  "cluster_node_count": 3
}
```

Jenis: ClusterService

```
{
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Count",
          "Name": "service_number_of_running_pods"
        }
      ],
      "Dimensions": [
        [
          "Service",
          "Namespace",
          "ClusterName"
        ],
        [
          "ClusterName"
```

```
        ]
      ],
      "Namespace": "ContainerInsights"
    }
  ],
  "ClusterName": "myCICluster",
  "Namespace": "amazon-cloudwatch",
  "Service": "cloudwatch-agent-statsd",
  "Sources": [
    "apiserver"
  ],
  "Timestamp": "1567097534160",
  "Type": "ClusterService",
  "Version": "0",
  "kubernetes": {
    "namespace_name": "amazon-cloudwatch",
    "service_name": "cloudwatch-agent-statsd"
  },
  "service_number_of_running_pods": 1
}
```

Jenis: ClusterNamespace

```
{
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Count",
          "Name": "namespace_number_of_running_pods"
        }
      ],
      "Dimensions": [
        [
          "Namespace",
          "ClusterName"
        ],
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    }
```

```
  ],
  "ClusterName": "myCICluster",
  "Namespace": "amazon-cloudwatch",
  "Sources": [
    "apiserver"
  ],
  "Timestamp": "1567097594160",
  "Type": "ClusterNamespace",
  "Version": "0",
  "kubernetes": {
    "namespace_name": "amazon-cloudwatch"
  },
  "namespace_number_of_running_pods": 7
}
```

Bidang-bidang relevan dalam peristiwa log performa untuk Amazon EKS dan Kubernetes

Untuk Amazon EKS dan Kubernetes, CloudWatch agen containerized memancarkan data sebagai peristiwa log kinerja. Ini memungkinkan CloudWatch untuk menelan dan menyimpan data kardinalitas tinggi. CloudWatch menggunakan data dalam peristiwa log kinerja untuk membuat CloudWatch metrik agregat di level cluster, node, dan pod tanpa perlu kehilangan detail granular.

Tabel berikut mencantumkan bidang-bidang dalam peristiwa log performa yang relevan dengan pengumpulan data metrik Wawasan Kontainer. Anda dapat menggunakan Wawasan CloudWatch Log untuk menanyakan salah satu bidang ini guna mengumpulkan data atau menyelidiki masalah. Untuk informasi selengkapnya, lihat Menganalisis Data Log Dengan Wawasan CloudWatch Log.

Tipe	Bidang log	Sumber	Rumus atau catatan
Pod	`pod_cpu_utilization`	Terkalkulasi	Rumus: `pod_cpu_usage_total / node_cpu_limit`
Pod	`pod_cpu_usage_total`	cadvisor	

Tipe	Bidang log		Sumber	Rumus atau catatan
	`pod_cpu_usage_total` dalam milicore.	dilaporkan		
Pod	`pod_cpu_limit`		Terkalkulasi	Rumus: `sum(container_cpu_limit)` `sum(container_cpu_limit)` menyertakan pod-pod yang sudah selesai. Jika ada kontainer di dalam pod yang tidak memiliki batas CPU yang ditentukan, bidang ini tidak akan ditampilkan di peristiwa log. Hal ini termasuk kontainer init.

Tipe	Bidang log	Sumber	Rumus atau catatan
Pod	`pod_cpu_request`	Terkalkulasi	Rumus: `sum(container_cpu_request)` `container_cpu_request` tidak dijamin akan ditetapkan. Hanya yang sudah dimasukkan saja yang sudah ada dalam jumlah tersebut.
Pod	`pod_cpu_utilization_over_pod_limit`	Terkalkulasi	Rumus: `pod_cpu_usage_total / pod_cpu_limit`
Pod	`pod_cpu_reserved_capacity`	Terkalkulasi	Rumus: `pod_cpu_request / node_cpu_limit`

Tipe	Bidang log	Sumber	Rumus atau catatan
Pod	`pod_memory_utilization`	Terkalkulasi	Rumus: `pod_memory_working_set / node_memory_limit` Ini adalah persentase penggunaan memori pod di atas batasan memori simpul.
Pod	`pod_memory_working_set`	cadvisor	
Pod	`pod_memory_limit`	Terkalkulasi	Rumus: `sum(container_memory_limit)` Jika ada kontainer di dalam pod yang tidak memiliki batas memori yang ditentukan, bidang ini tidak akan ditampilkan di peristiwa log. Hal ini termasuk kontainer init.

Tipe	Bidang log	Sumber	Rumus atau catatan
Pod	`pod_memory_request`	Terkalkulasi	Rumus: `sum(container_memory_request)` `container_memory_request` tidak dijamin akan ditetapkan. Hanya yang sudah dimasukkan saja yang sudah ada dalam jumlah tersebut.

Tipe	Bidang log	Sumber	Rumus atau catatan
Pod	`pod_memory_utilization_over_pod_limit`	Terkalkulasi	Rumus: `pod_memory_working_set / pod_memory_limit` Jika ada kontainer di dalam pod yang tidak memiliki batas memori yang ditentukan, bidang ini tidak akan ditampilkan di peristiwa log. Hal ini termasuk kontainer init.
Pod	`pod_memory_reserved_capacity`	Terkalkulasi	Rumus: `pod_memory_request / node_memory_limit`

Tipe	Bidang log	Sumber	Rumus atau catatan
Pod	`pod_network_tx_bytes`	Terkalkulasi	Rumus: `sum(pod_interface_network_tx_bytes)` Data ini tersedia untuk semua antarmuka jaringan untuk masing-masing pod. CloudWatch Agen menghitung total dan menambahkan aturan ekstraksi metrik.
Pod	`pod_network_rx_bytes`	Terkalkulasi	Rumus: `sum(pod_interface_network_rx_bytes)`

Tipe	Bidang log	Sumber	Rumus atau catatan
Pod	`pod_network_total_bytes`	Terkalkulasi	Formula: `pod_network_rx_bytes + pod_network_tx_bytes`
PodNet	`pod_interface_network_rx_bytes`	cadvisor	Data ini adalah rx bytes jaringan per detik dari antarmuka jaringan pod.
PodNet	`pod_interface_network_tx_bytes`	kadvisor	Data ini adalah byte tx jaringan per detik dari sebuah antarmuka jaringan pod.
Kontainer	`container_cpu_usage_total`	cadvisor	
Kontainer	`container_cpu_limit`	cadvisor	Tidak dijamin akan ditetapkan. Ini tidak akan dipancarkan jika tidak diatur.

Tipe	Bidang log	Sumber	Rumus atau catatan
Kontainer	`container_cpu_request`	cadvisor	Tidak dijamin akan ditetapkan. Ini tidak akan dipancarkan jika tidak diatur.
Kontainer	`container_memory_working_set`	cadvisor	
Kontainer	`container_memory_limit`	pod	Tidak dijamin akan ditetapkan. Ini tidak akan dipancarkan jika tidak diatur.
Kontainer	`container_memory_request`	pod	Tidak dijamin akan ditetapkan. Ini tidak akan dipancarkan jika tidak diatur.
ContainerFS	`container_filesystem_capacity`	pod	Data ini tersedia untuk setiap perangkat disk.
ContainerFS	`container_filesystem_usage`	pod	Data ini tersedia untuk setiap perangkat disk.

Tipe	Bidang log	Sumber	Rumus atau catatan
ContainerFS	`container_filesystem_utilization`	Terkalkulasi	Rumus: `container_filesystem_usage / container_filesystem_capacity` Data ini tersedia untuk masing-masing nama perangkat.
Simpul	`node_cpu_utilization`	Terkalkulasi	Rumus: `node_cpu_usage_total / node_cpu_limit`
Simpul	`node_cpu_usage_total`	cadvisor	
Simpul	`node_cpu_limit`	/proc	

Tipe	Bidang log	Sumber	Rumus atau catatan
Simpul	`node_cpu_request`	Terkalkulasi	Rumus: `sum(pod_cpu_request)` Untuk cronjobs, `node_cpu_request` juga akan menyertakan permintaan dari pod yang sudah selesai. Hal ini dapat menyebabkan nilai yang tinggi untuk `node_cpu_reserved_capacity` .
Simpul	`node_cpu_reserved_capacity`	Terkalkulasi	Rumus: `node_cpu_request / node_cpu_limit`
Simpul	`node_memory_utilization`	Terkalkulasi	Rumus: `node_memory_working_set / node_memory_limit`

Tipe	Bidang log	Sumber	Rumus atau catatan
Simpul	node_memory_working_set	cadvisor	
Simpul	node_memory_limit	/proc	
Simpul	node_memory_request	Terkalkulasi	Rumus: sum(pod_memory_request)
Simpul	node_memory_reserved_capacity	Terkalkulasi	Rumus: node_memory_request / node_memory_limit
Simpul	node_network_rx_bytes	Terkalkulasi	Rumus: sum(node_interface_network_rx_bytes)
Simpul	node_network_tx_bytes	Terkalkulasi	Rumus: sum(node_interface_network_tx_bytes)
Simpul	node_network_total_bytes	Terkalkulasi	Rumus: node_network_rx_bytes + node_network_tx_bytes

Tipe	Bidang log	Sumber	Rumus atau catatan
Simpul	`node_number_of_running_pods`	Daftar Pod	
Simpul	`node_number_of_running_containers`	Daftar Kelompok Kecil	
NodeNet	`node_interface_network_rx_bytes`	kadvisor	Data ini adalah rx bytes jaringan per detik dari antarmuka jaringan node pekerja.
NodeNet	`node_interface_network_tx_bytes`	kadvisor	Data ini adalah byte tx jaringan per detik dari sebuah antarmuka jaringan simpul pekerja.
NodeFS	`node_filesystem_capacity`	cadvisor	
NodeFS	`node_filesystem_usage`	cadvisor	

Tipe	Bidang log	Sumber	Rumus atau catatan
NodeFS	`node_filesystem_utilization`	Terkalkulasi	Rumus: `node_filesystem_usage / node_filesystem_capacity` Data ini tersedia untuk masing-masing nama perangkat.
Klaster	`cluster_failed_node_count`	Server API	
Klaster	`cluster_node_count`	Server API	
Layanan	`service_number_of_running_pods`	Server API	
`Namespace`	`namespace_number_of_running_pods`	Server API	

Contoh-contoh perhitungan metrik

Bagian ini mencakup contoh-contoh yang menunjukkan bagaimana beberapa nilai dalam tabel sebelumnya dikalkulasikan.

Bayangkan Anda memiliki sebuah klaster dalam status berikut.

```
Node1
   node_cpu_limit = 4
   node_cpu_usage_total = 3

   Pod1
```

```
      pod_cpu_usage_total = 2

      Container1
         container_cpu_limit = 1
         container_cpu_request = 1
         container_cpu_usage_total = 0.8

      Container2
         container_cpu_limit = null
         container_cpu_request = null
         container_cpu_usage_total = 1.2

   Pod2
      pod_cpu_usage_total = 0.4

      Container3
         container_cpu_limit = 1
         container_cpu_request = 0.5
         container_cpu_usage_total = 0.4

Node2
   node_cpu_limit = 8
   node_cpu_usage_total = 1.5

   Pod3
      pod_cpu_usage_total = 1

      Container4
         container_cpu_limit = 2
         container_cpu_request = 2
         container_cpu_usage_total = 1
```

Tabel berikut menunjukkan bagaimana metrik CPU pod dikalkulasikan dengan menggunakan data ini.

Metrik	Rumus .	Pod1	Pod2	Pod3
`pod_cpu_utilization`	`pod_cpu_usage_total / node_cpu_limit`	2 / 4 = 50%	0,4 / 4 = 10%	1 / 8 = 12.5%

Metrik	Rumus .	Pod1	Pod2	Pod3
pod_cpu_utilization_over_pod_limit	pod_cpu_usage_total / sum(container_cpu_limit)	Tidak berlaku karena batas CPU untuk Container 2 tidak ditentukan	0,4 / 1 = 40%	1 / 2 = 50%
pod_cpu_reserved_capacity	sum(container_cpu_request) / node_cpu_limit	(1 + 0) / 4 = 25%	0,5 / 4 = 12.5%	2 / 8 = 25%

Tabel berikut menunjukkan bagaimana metrik-metrik CPU simpul dikalkulasikan dengan menggunakan data ini.

Metrik	Rumus .	Node1	Node2
node_cpu_utilization	node_cpu_usage_total / node_cpu_limit	3 / 4 = 75%	1,5 / 8 = 18.75%
node_cpu_reserved_capacity	sum(pod_cpu_request) / node_cpu_limit	1,5 / 4 = 37.5%	2 / 8 = 25%

Pemantauan metrik-metrik Prometheus Wawasan Kontainer

CloudWatch Pemantauan Wawasan Kontainer untuk Prometheus mengotomatiskan penemuan metrik Prometheus dari sistem dan beban kerja dalam peti kemas. Prometheus adalah sebuah alat pemantauan dan peringatan sistem sumber terbuka. Untuk informasi selengkapnya, silakan lihat Apa itu Prometheus? dalam dokumentasi Prometheus.

Menemukan metrik-metrik Prometheus didukung untuk klaster Amazon Elastic Container Service, Amazon Elastic Kubernetes Service dan Kubernetes yang sedang berjalan di instans Amazon EC2.

Penghitung, pengukur, dan tipe-tipe metrik ringkasan Prometheus dikumpulkan. Dukungan untuk metrik-metrik histogram direncanakan untuk rilis mendatang.

Untuk klaster Amazon ECS dan Amazon EKS, baik tipe-tipe peluncuran EC2 dan Fargate didukung. Wawasan Kontainer secara otomatis mengumpulkan metrik-metrik dari beberapa beban kerja, dan Anda dapat mengonfigurasinya untuk mengumpulkan metrik-metrik dari beban kerja mana pun.

Anda dapat mengadopsi Prometheus sebagai metode open-source dan standar terbuka untuk memasukkan metrik khusus. CloudWatch CloudWatch Agen dengan dukungan Prometheus menemukan dan mengumpulkan metrik Prometheus untuk memantau, memecahkan masalah, dan alarm tentang penurunan kinerja aplikasi dan kegagalan lebih cepat. Agen CloudWatch ini juga mengurangi jumlah alat-alat pemantauan yang diperlukan untuk meningkatkan observabilitas.

Container Insights Dukungan Prometheus pay-per-use melibatkan metrik dan log, termasuk mengumpulkan, menyimpan, dan menganalisis. Untuk informasi selengkapnya, lihat CloudWatch Harga Amazon.

Dasbor yang sudah dibangun sebelumnya untuk beberapa beban kerja

Solusi Wawasan Kontainer Prometheus mencakup dasbor yang sudah dibangun sebelumnya untuk beban kerja populer yang dicantumkan dalam bagian ini. Untuk konfigurasi sampel untuk beban kerja ini, silakan lihat (Opsional) Menyiapkan sampel beban kerja Amazon ECS terkontainer untuk pengujian metrik Prometheus dan (Opsional) Menyiapkan sampel beban kerja Amazon EKS terkontainer untuk pengujian metrik Prometheus.

Anda juga dapat mengonfigurasi Wawasan Kontainer untuk mengumpulkan metrik-metrik Prometheus dari layanan-layanan dan aplikasi-aplikasi terkontainer lainnya dengan menyunting file konfigurasi agen.

Beban kerja dengan dasbor yang sudah dibangun sebelumnya untuk klaster Amazon EKS dan klaster Kubernetes yang berjalan di instans Amazon EC2:

- AWS App Mesh
- NGINX
- Memcached
- Java/JMX
- HAProxy

Beban kerja dengan dasbor yang sudah dibangun sebelumnya untuk klaster-klaster Amazon ECS:

- AWS App Mesh
- Java/JMX
- NGINX
- NGINX Plus

Mengatur dan mengonfigurasikan koleksi metrik-metrik Prometheus di klaster Amazon ECS

Untuk mengumpulkan metrik Prometheus dari cluster Amazon ECS, Anda dapat CloudWatch menggunakan agen sebagai kolektor atau menggunakan Distro untuk kolektor. AWS OpenTelemetry Untuk informasi tentang menggunakan AWS Distro untuk OpenTelemetry kolektor, lihat https://aws-otel.github.io/docs/getting-started/container-insights/ecs-prometheus.

Bagian berikut menjelaskan cara menggunakan CloudWatch agen sebagai kolektor untuk mengambil metrik Prometheus. Anda menginstal CloudWatch agen dengan pemantauan Prometheus pada cluster yang menjalankan Amazon ECS, dan Anda dapat mengonfigurasi agen secara opsional untuk mengikis target tambahan. Bagian-bagian ini juga akan menyediakan tutorial opsional untuk menyiapkan beban kerja sampel untuk digunakan dalam pengujian dengan pemantauan Prometheus.

Wawasan Kontainer di Amazon ECS mendukung tipe-tipe peluncuran dan kombinasi mode jaringan berikut untuk metrik Prometheus:

Tipe peluncuran Amazon ECS	Mode jaringan yang didukung
EC2 (Linux)	jembatan, host, dan awsvpc
Fargate	awsvpc

Persyaratan grup keamanan VPC

Aturan masuknya kelompok keamanan untuk beban kerja Prometheus harus membuka port CloudWatch Prometheus ke agen untuk mengikis metrik Prometheus oleh IP pribadi.

Aturan keluar dari grup keamanan untuk CloudWatch agen harus memungkinkan agen untuk terhubung ke port CloudWatch beban kerja Prometheus dengan IP pribadi.

Topik

- Instal CloudWatch agen dengan koleksi metrik Prometheus di klaster Amazon ECS
- Melakukan scraping atas sumber Prometheus tambahan dan mengimpor metrik-metrik tersebut
- (Opsional) Menyiapkan sampel beban kerja Amazon ECS terkontainer untuk pengujian metrik Prometheus

Instal CloudWatch agen dengan koleksi metrik Prometheus di klaster Amazon ECS

Bagian ini menjelaskan cara mengatur CloudWatch agen dengan pemantauan Prometheus di cluster yang menjalankan Amazon ECS. Setelah melakukan hal tersebut, agen secara otomatis melakukan scraping terhadap dan mengimpor metrik-metrik untuk beban kerja berikut yang berjalan di klaster tersebut.

- AWS App Mesh
- Java/JMX

Anda juga dapat mengonfigurasi agen tersebut untuk melakukan scraping atas dan mengimpor metrik-metrik dari beban kerja dan sumber Prometheus tambahan.

Menyiapkan peran IAM

Anda memerlukan dua peran IAM untuk definisi tugas CloudWatch agen. Jika Anda menentukan **`CreateIAMRoles=True`** dalam AWS CloudFormation tumpukan agar Wawasan Kontainer membuat peran ini untuk Anda, peran akan dibuat dengan izin yang benar. Jika Anda ingin membuat peran-peran itu sendiri atau menggunakan peran-peran yang sudah ada, maka peran dan izin berikut diperlukan.

- CloudWatch peran tugas agen ECS - Wadah CloudWatch agen menggunakan peran ini. Ini harus menyertakan CloudWatchAgentServerPolicykebijakan dan kebijakan yang dikelola pelanggan yang berisi izin hanya-baca berikut:
  - `ec2:DescribeInstances`
  - `ecs:ListTasks`
  - `ecs:ListServices`
  - `ecs:DescribeContainerInstances`
  - `ecs:DescribeServices`
  - `ecs:DescribeTasks`
  - `ecs:DescribeTaskDefinition`

- CloudWatch peran eksekusi tugas agen ECS — Ini adalah peran yang diperlukan Amazon ECS untuk meluncurkan dan menjalankan kontainer Anda. Pastikan peran eksekusi tugas Anda memiliki AmazonSSM, ReadOnlyAccess AmazonECS, dan kebijakan yang dilampirkan. TaskExecutionRolePolicy CloudWatchAgentServerPolicy Jika Anda ingin menyimpan data yang lebih sensitif untuk digunakan oleh Amazon ECS, silakan lihat Menentukan data sensitif.

Instal CloudWatch agen dengan pemantauan Prometheus dengan menggunakan AWS CloudFormation

Anda gunakan AWS CloudFormation untuk menginstal CloudWatch agen dengan pemantauan Prometheus untuk klaster Amazon ECS. Daftar berikut menunjukkan parameter-parameter yang akan Anda gunakan dalam template AWS CloudFormation .

- ECS ClusterName - Menentukan target klaster Amazon ECS.
- CreateIAMRoles— Menentukan **`True`** untuk menciptakan peran-peran baru untuk peran tugas Amazon ECS dan peran eksekusi tugas Amazon ECS. Tentukan **`False`** untuk menggunakan kembali peran-peran yang ada.
- TaskRoleName— Jika Anda menentukan **`True`** CreateIamRoles, ini menentukan nama yang akan digunakan untuk peran tugas Amazon ECS yang baru. Jika Anda sudah menentukan **`False`** untuk CreateIAMRoles, maka hal ini akan menentukan peran yang sudah ada untuk digunakan sebagai peran tugas Amazon ECS.
- ExecutionRoleName— Jika Anda menentukan **`True`** CreateIamRoles, ini menentukan nama yang akan digunakan untuk peran eksekusi tugas Amazon ECS yang baru. Jika Anda sudah menentukan **`False`** untuk CreateIAMRoles, maka hal ini akan menentukan peran yang sudah ada untuk digunakan sebagai peran eksekusi tugas Amazon ECS.
- ECS NetworkMode - Jika Anda menggunakan tipe peluncuran EC2, tentukan mode jaringan di sini. Harus **`bridge`** atau **`host`**.
- ECS LaunchType — Tentukan salah satu **`fargate`** atau**`EC2`**.
- SecurityGroupID — Jika ECS NetworkMode adalah**`awsvpc`**, tentukan ID grup keamanan di sini.
- SubnetID — Jika ECS NetworkMode adalah**`awsvpc`**, tentukan subnet ID di sini.

Sampel-sampel perintah

Bagian ini mencakup contoh AWS CloudFormation perintah untuk menginstal Wawasan Kontainer dengan pemantauan Prometheus dalam berbagai skenario.

Buat AWS CloudFormation tumpukan untuk cluster Amazon ECS dalam mode jaringan jembatan

```
export AWS_PROFILE=your_aws_config_profile_eg_default
export AWS_DEFAULT_REGION=your_aws_region_eg_ap-southeast-1
export ECS_CLUSTER_NAME=your_ec2_ecs_cluster_name
export ECS_NETWORK_MODE=bridge
export CREATE_IAM_ROLES=True
export ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
export ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-
prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-bridge-host.yaml

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=${ECS_CLUSTER_NAME} \
                 ParameterKey=CreateIAMRoles,ParameterValue=${CREATE_IAM_ROLES} \
                 ParameterKey=ECSNetworkMode,ParameterValue=${ECS_NETWORK_MODE} \
                 ParameterKey=TaskRoleName,ParameterValue=${ECS_TASK_ROLE_NAME} \
                 ParameterKey=ExecutionRoleName,ParameterValue=
${ECS_EXECUTION_ROLE_NAME} \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${AWS_DEFAULT_REGION} \
    --profile ${AWS_PROFILE}
```

Buat AWS CloudFormation tumpukan untuk cluster Amazon ECS dalam mode jaringan host

```
export AWS_PROFILE=your_aws_config_profile_eg_default
export AWS_DEFAULT_REGION=your_aws_region_eg_ap-southeast-1
export ECS_CLUSTER_NAME=your_ec2_ecs_cluster_name
export ECS_NETWORK_MODE=host
export CREATE_IAM_ROLES=True
export ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
export ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name


curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-
prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-bridge-host.yaml
```

```
aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=${ECS_CLUSTER_NAME} \
                 ParameterKey=CreateIAMRoles,ParameterValue=${CREATE_IAM_ROLES} \
                 ParameterKey=ECSNetworkMode,ParameterValue=${ECS_NETWORK_MODE} \
                 ParameterKey=TaskRoleName,ParameterValue=${ECS_TASK_ROLE_NAME} \
                 ParameterKey=ExecutionRoleName,ParameterValue=
${ECS_EXECUTION_ROLE_NAME} \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${AWS_DEFAULT_REGION} \
    --profile ${AWS_PROFILE}
```

Buat AWS CloudFormation tumpukan untuk cluster Amazon ECS dalam mode jaringan awsvpc

```
export AWS_PROFILE=your_aws_config_profile_eg_default
export AWS_DEFAULT_REGION=your_aws_region_eg_ap-southeast-1
export ECS_CLUSTER_NAME=your_ec2_ecs_cluster_name
export ECS_LAUNCH_TYPE=EC2
export CREATE_IAM_ROLES=True
export ECS_CLUSTER_SECURITY_GROUP=your_security_group_eg_sg-xxxxxxxxxx
export ECS_CLUSTER_SUBNET=your_subnet_eg_subnet-xxxxxxxxxx
export ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
export ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-
prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-awsvpc.yaml

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-${ECS_LAUNCH_TYPE}-awsvpc \
    --template-body file://cwagent-ecs-prometheus-metric-for-awsvpc.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=${ECS_CLUSTER_NAME} \
                 ParameterKey=CreateIAMRoles,ParameterValue=${CREATE_IAM_ROLES} \
                 ParameterKey=ECSLaunchType,ParameterValue=${ECS_LAUNCH_TYPE} \
                 ParameterKey=SecurityGroupID,ParameterValue=
${ECS_CLUSTER_SECURITY_GROUP} \
                 ParameterKey=SubnetID,ParameterValue=${ECS_CLUSTER_SUBNET} \
                 ParameterKey=TaskRoleName,ParameterValue=${ECS_TASK_ROLE_NAME} \
                 ParameterKey=ExecutionRoleName,ParameterValue=
${ECS_EXECUTION_ROLE_NAME} \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${AWS_DEFAULT_REGION} \
```

```
    --profile ${AWS_PROFILE}
```

Buat AWS CloudFormation tumpukan untuk cluster Fargate dalam mode jaringan awsvpc

```
export AWS_PROFILE=your_aws_config_profile_eg_default
export AWS_DEFAULT_REGION=your_aws_region_eg_ap-southeast-1
export ECS_CLUSTER_NAME=your_ec2_ecs_cluster_name
export ECS_LAUNCH_TYPE=FARGATE
export CREATE_IAM_ROLES=True
export ECS_CLUSTER_SECURITY_GROUP=your_security_group_eg_sg-xxxxxxxxxx
export ECS_CLUSTER_SUBNET=your_subnet_eg_subnet-xxxxxxxxxx
export ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
export ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-
prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-awsvpc.yaml

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-${ECS_LAUNCH_TYPE}-awsvpc \
    --template-body file://cwagent-ecs-prometheus-metric-for-awsvpc.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=${ECS_CLUSTER_NAME} \
                 ParameterKey=CreateIAMRoles,ParameterValue=${CREATE_IAM_ROLES} \
                 ParameterKey=ECSLaunchType,ParameterValue=${ECS_LAUNCH_TYPE} \
                 ParameterKey=SecurityGroupID,ParameterValue=
${ECS_CLUSTER_SECURITY_GROUP} \
                 ParameterKey=SubnetID,ParameterValue=${ECS_CLUSTER_SUBNET} \
                 ParameterKey=TaskRoleName,ParameterValue=${ECS_TASK_ROLE_NAME} \
                 ParameterKey=ExecutionRoleName,ParameterValue=
${ECS_EXECUTION_ROLE_NAME} \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${AWS_DEFAULT_REGION} \
    --profile ${AWS_PROFILE}
```

AWS sumber daya yang dibuat oleh AWS CloudFormation tumpukan

Tabel berikut mencantumkan AWS sumber daya yang dibuat saat Anda menggunakan AWS CloudFormation untuk menyiapkan Wawasan Kontainer dengan pemantauan Prometheus di klaster Amazon ECS.

Tipe sumber daya	Nama sumber daya	Komentar
AWS::SSM::Parameter	AmazonCloudWatch-*CW -$ ECS_CLUSTER_NAME AgentConfig -$ ECS_LAUNCH_TYPE -$ ECS_NETWORK_MODE*	Ini adalah CloudWatch agen dengan definisi format metrik tertanam App Mesh dan Java/JMX default.
AWS::SSM::Parameter	AmazonCloudWatch-*- - $ PrometheusConfigName ECS_CLUSTER_NAME -$ ECS_LAUNCH_TYPE -$ ECS_NETWORK_MODE*	Ini adalah konfigurasi scraping Prometheus.
AWS::IAM::Role	$ECS_TASK_ROLE_NAME.	Peran tugas Amazon ECS. Ini hanya dibuat jika Anda menentukan **`True`** untuk `CREATE_IAM_ROLES` .
AWS::IAM::Role	${ECS_EXECUTION_ROLE_NAME}	Peran eksekusi tugas Amazon ECS. Ini hanya dibuat jika Anda sudah menentukan **`True`** untuk `CREATE_IAM_ROLES` .
AWS::ECS::TaskDefinition	cwagent-prometheus-$*ECS_CLUSTER_NAME* -$*ECS_LAUNCH_TYPE* -$*ECS_NETWORK_MODE*	
AWS::ECS::Service	cwagent-prometheus-replica-service-*$ ECS_LAUNCH_TYPE -$ ECS_NETWORK_MODE*	

Menghapus AWS CloudFormation tumpukan untuk CloudWatch agen dengan pemantauan Prometheus

Untuk menghapus CloudWatch agen dari cluster Amazon ECS, masukkan perintah ini.

```
export AWS_PROFILE=your_aws_config_profile_eg_default
```

```
export AWS_DEFAULT_REGION=your_aws_region_eg_ap-southeast-1
export CLOUDFORMATION_STACK_NAME=your_cloudformation_stack_name

aws cloudformation delete-stack \
--stack-name ${CLOUDFORMATION_STACK_NAME} \
--region ${AWS_DEFAULT_REGION} \
--profile ${AWS_PROFILE}
```

Melakukan scraping atas sumber Prometheus tambahan dan mengimpor metrik-metrik tersebut

CloudWatch Agen dengan pemantauan Prometheus membutuhkan dua konfigurasi untuk mengikis metrik Prometheus. Salah satunya adalah konfigurasi standar Prometheus sebagaimana yang didokumentasikan dalam <scrape_config> dalam dokumentasi Prometheus. Yang lainnya adalah untuk konfigurasi CloudWatch agen.

Untuk klaster-klaster Amazon ECS, konfigurasi diintegrasikan dengan Parameter Store AWS Systems Manager oleh rahasia yang ada dalam penetapan tugas Amazon ECS:

- `PROMETHEUS_CONFIG_CONTENT` rahasia adalah untuk konfigurasi scraping Prometheus.
- Rahasianya `CW_CONFIG_CONTENT` adalah untuk konfigurasi CloudWatch agen.

Untuk mengikis sumber metrik Prometheus tambahan dan mengimpor metrik tersebut, Anda memodifikasi CloudWatch konfigurasi scrape Prometheus dan konfigurasi agen, lalu menerapkan kembali agen dengan konfigurasi yang diperbarui. CloudWatch

Persyaratan grup keamanan VPC

Aturan masuknya kelompok keamanan untuk beban kerja Prometheus harus membuka port CloudWatch Prometheus ke agen untuk mengikis metrik Prometheus oleh IP pribadi.

Aturan keluar dari grup keamanan untuk CloudWatch agen harus memungkinkan agen untuk terhubung ke port CloudWatch beban kerja Prometheus dengan IP pribadi.

Konfigurasi scraping Prometheus

CloudWatch <scrape_config>Agen mendukung konfigurasi scrape Prometheus standar seperti yang didokumentasikan dalam dokumentasi Prometheus.https://prometheus.io/docs/prometheus/latest/configuration/configuration/#scrape_config Anda dapat mengedit bagian ini untuk memperbarui konfigurasi yang sudah ada dalam file ini, dan menambahkan target-target scraping Prometheus tambahan. Secara bawaan, file konfigurasi sampel berisi baris-baris konfigurasi global berikut ini:

```
global:
  scrape_interval: 1m
  scrape_timeout: 10s
```

- scrape_interval— Mendefinisikan seberapa sering scraping akan dilakukan ke target.
- scrape_timeout— Mendefinisikan berapa lama waktu tunggu sebelum permintaan scraping habis.

Anda juga dapat menentukan nilai-nilai yang berbeda untuk pengaturan ini di level pekerjaan, untuk mengganti konfigurasi global.

Pekerjaan scraping Prometheus

File CloudWatch agen YAMAL sudah memiliki beberapa pekerjaan pengikisan default yang dikonfigurasi. Sebagai contoh, dalam file YAML untuk Amazon ECS seperti `cwagent-ecs-prometheus-metric-for-bridge-host.yaml`, pekerjaan scraping bawaan dikonfigurasi di bagian `ecs_service_discovery`.

```
"ecs_service_discovery": {
                  "sd_frequency": "1m",
                  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
                  "docker_label": {
                  },
                  "task_definition_list": [
                    {
                      "sd_job_name": "ecs-appmesh-colors",
                      "sd_metrics_ports": "9901",
                      "sd_task_definition_arn_pattern": ".*:task-definition\/.*-
ColorTeller-(white):[0-9]+",
                      "sd_metrics_path": "/stats/prometheus"
                    },
                    {
                      "sd_job_name": "ecs-appmesh-gateway",
                      "sd_metrics_ports": "9901",
                      "sd_task_definition_arn_pattern": ".*:task-definition/.*-
ColorGateway:[0-9]+",
                      "sd_metrics_path": "/stats/prometheus"
                    }
                  ]
                }
```

Masing-masing target default ini dikikis, dan metrik dikirim ke CloudWatch peristiwa log menggunakan format metrik yang disematkan. Untuk informasi selengkapnya, lihat Menyematkan metrik dalam log.

Peristiwa log dari klaster Amazon ECS disimpan di grup log /aws/ecs/containerinsights/***cluster_name***/prometheus.

Setiap pekerjaan scraping dimuat dalam log stream yang berbeda di grup log ini.

Untuk menambahkan sebuah target scraping baru, Anda harus menambahkan sebuah entri baru di bagian `task_definition_list` pada bagian `ecs_service_discovery` file YAML, dan kemudian mulai ulang agen tersebut. Untuk contoh proses ini, silakan lihat Tutorial untuk menambahkan sebuah target scraping Prometheus baru: metrik Server API Prometheus.

CloudWatch konfigurasi agen untuk Prometheus

File konfigurasi CloudWatch agen memiliki `prometheus` bagian di bawah `metrics_collected` untuk konfigurasi pengikisan Prometheus. File konfigurasi tersebut mencakup opsi-opsi konfigurasi berikut:

- cluster_name— menentukan nama klaster yang akan ditambahkan sebagai label pada peristiwa log. Bidang ini bersifat opsional. Jika Anda menghilangkannya, maka agen akan dapat mendeteksi nama klaster Amazon ECS.
- log_group_name— menentukan nama grup log untuk metrik-metrik Prometheus yang di-scraping. Bidang ini bersifat opsional. *Jika Anda menghilangkannya, CloudWatch gunakan* ***/aws/ecs/containerinsights/ cluster_name /prometheus*** *untuk log dari cluster Amazon ECS.*
- prometheus_config_path— menentukan jalur file konfigurasi scraping Prometheus. Jika nilai bidang ini dimulai dengan `env:`, maka konten file konfigurasi Prometheus scraping akan diambil dari variabel lingkungan kontainer. Jangan ubah bidang ini.
- ecs_service_discovery— adalah bagian untuk menentukan konfigurasi fungsi penemuan otomatis target Amazon ECS Prometheus. Dua mode didukung untuk menemukan target-target Prometheus: penemuan berdasarkan label docker kontainer atau penemuan berdasarkan ekspresi reguler ARN penetapan tugas Amazon ECS. Anda dapat menggunakan dua mode bersama-sama dan CloudWatch agen akan menghapus duplikasi target yang ditemukan berdasarkan: {private_ip}: {port}/{metrics_path}.

  Bagian `ecs_service_discovery` dapat berisi bidang-bidang berikut:

- `sd_frequency` adalah frekuensi untuk menemukan pengekspor Prometheus. Tentukan sebuah angka dan sebuah akhiran unit. Sebagai contoh, `1m` satu kali per menit atau `30s` satu kali per 30 detik. Akhiran unit yang valid adalah `ns`, `us`, `ms`, `s`, `m`, dan `h`.

  Bidang ini bersifat opsional. Bawaannya adalah 60 detik (1 menit).

- `sd_target_cluster` adalah nama klaster Amazon ECS target untuk penemuan otomatis. Bidang ini bersifat opsional. Defaultnya adalah nama cluster Amazon ECS tempat CloudWatch agen diinstal.

- `sd_cluster_region` adalah Wilayah klaster Amazon ECS target. Bidang ini bersifat opsional. Defaultnya adalah Wilayah cluster Amazon ECS tempat CloudWatch agen diinstal.

- `sd_result_file` adalah jalur file YAML untuk hasil-hasil target Prometheus. Konfigurasi scraping Prometheus akan merujuk pada file ini.

- `docker_label` adalah sebuah bagian opsional yang dapat Anda gunakan untuk menentukan konfigurasi untuk penemuan layanan berbasis label docker. Jika Anda menghilangkan bagian ini, maka penemuan berbasis label docker tidak akan digunakan. Bagian ini dapat berisi bidang-bidang berikut:

  - `sd_port_label` adalah nama label docker dari kontainer yang menentukan port kontainer untuk metrik-metrik Prometheus. Nilai bawaannya adalah `ECS_PROMETHEUS_EXPORTER_PORT`. Jika wadah tidak memiliki label docker ini, CloudWatch agen akan melewatkannya.

  - `sd_metrics_path_label` adalah nama label docker dari kontainer yang menentukan jalur metrik-metrik Prometheus. Nilai bawaannya adalah `ECS_PROMETHEUS_METRICS_PATH`. Jika kontainer tidak memiliki label docker ini, maka agen akan mengasumsikan jalur default `/metrics`.

  - `sd_job_name_label` adalah nama label docker dari kontainer yang menentukan nama pekerjaan scraping Prometheus. Nilai bawaannya adalah `job`. Jika wadah tidak memiliki label docker ini, CloudWatch agen menggunakan nama pekerjaan dalam konfigurasi scrape Prometheus.

- `task_definition_list` adalah sebuah bagian opsional yang dapat Anda gunakan untuk menentukan konfigurasi penemuan layanan berbasis penetapan tugas. Jika Anda menghilangkan bagian ini, maka penemuan berbasis penetapan tugas tidak akan digunakan. Bagian ini dapat berisi bidang-bidang berikut:

  - `sd_task_definition_arn_pattern` adalah pola yang akan digunakan untuk menentukan penetapan tugas Amazon ECS yang harus ditemukan. Ini adalah ekspresi biasa.

    - `sd_metrics_ports` mencantumkan containerPort untuk metrik-metrik Prometheus. Pisahkan containerPorts dengan titik koma.
    - `sd_container_name_pattern` menentukan nama-nama kontainer tugas Amazon ECS. Ini adalah ekspresi biasa.
    - `sd_metrics_path` menentukan jalur metrik Prometheus. Jika Anda menghilangkannya, maka agen tersebut akan mengasumsikan jalur bawaan `/metrics`
    - `sd_job_name` menyebutkan nama pekerjaan scraping Prometheus. Jika Anda menghilangkan bidang ini, CloudWatch agen menggunakan nama pekerjaan dalam konfigurasi scrape Prometheus.
- `service_name_list_for_tasks` adalah sebuah bagian opsional yang dapat Anda gunakan untuk menentukan konfigurasi penemuan berbasis nama layanan. Jika Anda menghilangkan bagian ini, maka penemuan berbasis nama layanan tidak akan digunakan. Bagian ini dapat berisi bidang-bidang berikut:
    - `sd_service_name_pattern` adalah pola yang digunakan untuk menentukan layanan Amazon ECS tempat di mana tugas ditemukan. Ini adalah ekspresi biasa.
    - `sd_metrics_ports` Mencantumkan `containerPort` untuk metrik-metrik Prometheus. Pisahkan beberapa `containerPorts` dengan titik koma.
    - `sd_container_name_pattern` menentukan nama-nama kontainer tugas Amazon ECS. Ini adalah ekspresi biasa.
    - `sd_metrics_path` menentukan jalur metrik-metrik Prometheus. Jika Anda menghilangkannya, maka agen tersebut akan mengasumsikan jalur bawaannya adalah `/metrics`
    - `sd_job_name` menyebutkan nama pekerjaan scraping Prometheus. Jika Anda menghilangkan bidang ini, CloudWatch agen menggunakan nama pekerjaan dalam konfigurasi scrape Prometheus.
- metric_declaration— adalah bagian-bagian yang menentukan larik log dengan format metrik tersemat yang akan dihasilkan. Ada `metric_declaration` bagian untuk setiap sumber Prometheus yang diimpor agen secara default CloudWatch. Masing-masing bagian ini mencakup bidang-bidang berikut:
    - `label_matcher` adalah ekspresi reguler yang memeriksa nilai dari label-label yang tercantum dalam `source_labels`. Metrik yang cocok diaktifkan untuk dimasukkan dalam format metrik tertanam yang dikirim ke CloudWatch.

Jika Anda memiliki beberapa label yang ditentukan dalam `source_labels`, maka kami menyarankan Anda untuk tidak menggunakan karakter ^ atau $ dalam ekspresi reguler untuk `label_matcher`.

- `source_labels` menentukan nilai dari label-label yang diperiksa oleh baris `label_matcher`.
- `label_separator` menentukan pemisah yang akan digunakan dalam baris `label_matcher` jika ada beberapa `source_labels` yang ditetapkan. Bawaannya adalah `;`. Anda dapat melihat nilai-nilai bawaan ini digunakan di `label_matcher` dalam contoh berikut.
- `metric_selectors`adalah ekspresi reguler yang menentukan metrik yang akan dikumpulkan dan dikirim ke. CloudWatch
- `dimensions`adalah daftar label yang akan digunakan sebagai CloudWatch dimensi untuk setiap metrik yang dipilih.

Lihat contoh `metric_declaration` berikut ini.

```
"metric_declaration": [
  {
     "source_labels":[ "Service", "Namespace"],
     "label_matcher":"(.*node-exporter.*|.*kube-dns.*);kube-system$",
     "dimensions":[
        ["Service", "Namespace"]
     ],
     "metric_selectors":[
        "^coredns_dns_request_type_count_total$"
     ]
  }
]
```

Contoh ini mengonfigurasikan sebuah bagian format metrik tersemat yang akan dikirim sebagai sebuah peristiwa log jika kondisi-kondisi berikut dipenuhi:

- Nilai dari `Service` berisi `node-exporter` atau `kube-dns`.
- Nilai dari `Namespace` adalah `kube-system`.
- Metrik Prometheus `coredns_dns_request_type_count_total` memuat label `Service` dan `Namespace`.

Peristiwa log yang dikirim mencakup bagian yang disorot berikut ini:

```
{
   "CloudWatchMetrics":[
      {
         "Metrics":[
            {
               "Name":"coredns_dns_request_type_count_total"
            }
         ],
         "Dimensions":[
            [
               "Namespace",
               "Service"
            ]
         ],
         "Namespace":"ContainerInsights/Prometheus"
      }
   ],
   "Namespace":"kube-system",
   "Service":"kube-dns",
   "coredns_dns_request_type_count_total":2562,
   "eks_amazonaws_com_component":"kube-dns",
   "instance":"192.168.61.254:9153",
   "job":"kubernetes-service-endpoints",
   ...
}
```

Panduan mendetail untuk penemuan otomatis di klaster-klaster Amazon ECS

Prometheus menyediakan berlusin-lusin mekanisme penemuan layanan dinamis seperti yang dijelaskan di <scrape_config>. Namun demikian, tidak ada penemuan layanan bawaan untuk Amazon ECS. CloudWatch Agen menambahkan mekanisme ini.

Saat penemuan layanan Amazon ECS Prometheus diaktifkan, CloudWatch agen secara berkala melakukan panggilan API berikut ke frontend Amazon ECS dan Amazon EC2 untuk mengambil metadata tugas ECS yang sedang berjalan di cluster ECS target.

```
EC2:DescribeInstances
ECS:ListTasks
ECS:ListServices
ECS:DescribeContainerInstances
ECS:DescribeServices
ECS:DescribeTasks
```

```
ECS:DescribeTaskDefinition
```

Metadata digunakan oleh CloudWatch agen untuk memindai target Prometheus dalam cluster ECS. CloudWatch Agen mendukung tiga mode penemuan layanan:

- Penemuan layanan berbasis label docker kontainer
- Penemuan layanan berbasis ekspresi reguler ARN penetapan tugas ARC
- Penemuan layanan berbasis ekspresi reguler nama layanan ECS

Semua mode dapat digunakan bersama. CloudWatch agen de-duplikat target yang ditemukan berdasarkan:. `{private_ip}:{port}/{metrics_path}`

Semua target yang ditemukan ditulis ke dalam file hasil yang ditentukan oleh bidang `sd_result_file` konfigurasi dalam wadah CloudWatch agen. Berikut ini adalah sebuah file hasil sampel:

```
- targets:
  - 10.6.1.95:32785
  labels:
    __metrics_path__: /metrics
    ECS_PROMETHEUS_EXPORTER_PORT: "9406"
    ECS_PROMETHEUS_JOB_NAME: demo-jar-ec2-bridge-dynamic
    ECS_PROMETHEUS_METRICS_PATH: /metrics
    InstanceType: t3.medium
    LaunchType: EC2
    SubnetId: subnet-123456789012
    TaskDefinitionFamily: demo-jar-ec2-bridge-dynamic-port
    TaskGroup: family:demo-jar-ec2-bridge-dynamic-port
    TaskRevision: "7"
    VpcId: vpc-01234567890
    container_name: demo-jar-ec2-bridge-dynamic-port
    job: demo-jar-ec2-bridge-dynamic
- targets:
  - 10.6.3.193:9404
  labels:
    __metrics_path__: /metrics
    ECS_PROMETHEUS_EXPORTER_PORT_SUBSET_B: "9404"
    ECS_PROMETHEUS_JOB_NAME: demo-tomcat-ec2-bridge-mapped-port
    ECS_PROMETHEUS_METRICS_PATH: /metrics
    InstanceType: t3.medium
    LaunchType: EC2
```

```
    SubnetId: subnet-123456789012
    TaskDefinitionFamily: demo-tomcat-ec2-bridge-mapped-port
    TaskGroup: family:demo-jar-tomcat-bridge-mapped-port
    TaskRevision: "12"
    VpcId: vpc-01234567890
    container_name: demo-tomcat-ec2-bridge-mapped-port
    job: demo-tomcat-ec2-bridge-mapped-port
```

Anda dapat mengintegrasikan file hasil ini secara langsung dengan penemuan layanan berbasis file Prometheus. Untuk informasi selengkapnya tentang penemuan layanan berbasis file Prometheus, silakan lihat <file_sd_config>.

Misalkan file hasil ditulis ke `/tmp/cwagent_ecs_auto_sd.yaml` Konfigurasi scraping Prometheus berikut akan menggunakannya.

```
global:
  scrape_interval: 1m
  scrape_timeout: 10s
scrape_configs:
  - job_name: cwagent-ecs-file-sd-config
    sample_limit: 10000
    file_sd_configs:
      - files: [ "/tmp/cwagent_ecs_auto_sd.yaml" ]
```

CloudWatch Agen juga menambahkan label tambahan berikut untuk target yang ditemukan.

- `container_name`
- `TaskDefinitionFamily`
- `TaskRevision`
- `TaskGroup`
- `StartedBy`
- `LaunchType`
- `job`
- `__metrics_path__`
- Label docker

Saat klaster memiliki tipe peluncuran EC2, tiga label berikut ditambahkan.

- `InstanceType`
- `VpcId`
- `SubnetId`

> **Note**
>
> Label docker yang tidak cocok dengan ekspresi reguler `[a-zA-Z_][a-zA-Z0-9_]*` akan difilter keluar. Ini cocok dengan konvensi Prometheus sebagaimana tercantum di `label_name` dalam File konfigurasi dalam dokumentasi Prometheus.

Contoh konfigurasi penemuan layanan ECS

Bagian ini mencakup contoh-contoh yang menunjukkan penemuan layanan ECS.

Contoh 1

```
"ecs_service_discovery": {
  "sd_frequency": "1m",
  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
  "docker_label": {
  }
}
```

Contoh ini memungkinkan penemuan layanan berbasis label docker. CloudWatch Agen akan menanyakan metadata tugas ECS sekali per menit dan menulis target yang ditemukan ke dalam `/tmp/cwagent_ecs_auto_sd.yaml` file di dalam wadah agen. CloudWatch

Nilai bawaan dari `sd_port_label` di bagian `docker_label` adalah `ECS_PROMETHEUS_EXPORTER_PORT`. Jika ada wadah yang berjalan dalam tugas ECS yang memiliki label `ECS_PROMETHEUS_EXPORTER_PORT` docker, CloudWatch agen menggunakan `container port` nilainya untuk memindai semua port kontainer yang terbuka. Jika ada kecocokan, maka port host yang dipetakan ditambah IP privat kontainer akan digunakan untuk membangun konsep target pengekspor Prometheus dalam format berikut: `private_ip:host_port`.

Nilai bawaan dari `sd_metrics_path_label` di bagian `docker_label` adalah `ECS_PROMETHEUS_METRICS_PATH`. Jika kontainer tersebut memiliki label docker ini, maka nilainya akan digunakan sebagai `__metrics_path__`. Jika kontainer tidak memiliki label ini, maka nilai bawaannya `/metrics` akan digunakan.

Nilai bawaan dari `sd_job_name_label` di bagian `docker_label` adalah `job`. Jika kontainer memiliki label docker ini, maka nilainya akan ditambahkan sebagai salah satu label target untuk menggantikan nama pekerjaan bawaan yang ditentukan dalam konfigurasi Prometheus. Nilai label docker ini digunakan sebagai nama aliran log di grup CloudWatch log Log.

Contoh 2

```
"ecs_service_discovery": {
  "sd_frequency": "15s",
  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
  "docker_label": {
    "sd_port_label": "ECS_PROMETHEUS_EXPORTER_PORT_SUBSET_A",
    "sd_job_name_label": "ECS_PROMETHEUS_JOB_NAME"
  }
}
```

Contoh ini memungkinkan penemuan layanan berbasis label docker. CloudWatch Agen akan menanyakan metadata tugas ECS setiap 15 detik dan menulis target yang ditemukan ke dalam `/tmp/cwagent_ecs_auto_sd.yaml` file di dalam wadah agen. CloudWatch Kontainer yang memiliki label docker `ECS_PROMETHEUS_EXPORTER_PORT_SUBSET_A` akan dipindai. Nilai label docker `ECS_PROMETHEUS_JOB_NAME` akan digunakan sebagai nama pekerjaan.

Contoh 3

```
"ecs_service_discovery": {
  "sd_frequency": "5m",
  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
  "task_definition_list": [
    {
      "sd_job_name": "java-prometheus",
      "sd_metrics_path": "/metrics",
      "sd_metrics_ports": "9404; 9406",
      "sd_task_definition_arn_pattern": ".*:task-definition/.*javajmx.*:[0-9]+"
    },
    {
      "sd_job_name": "envoy-prometheus",
      "sd_metrics_path": "/stats/prometheus",
      "sd_container_name_pattern": "^envoy$",
      "sd_metrics_ports": "9901",
      "sd_task_definition_arn_pattern": ".*:task-definition/.*appmesh.*:23"
    }
```

```
  ]
}
```

Contoh ini memungkinkan penemuan layanan berbasis ekspresi reguler ARN penetapan tugas ECS. CloudWatch Agen akan menanyakan metadata tugas ECS setiap lima menit dan menulis target yang ditemukan ke dalam `/tmp/cwagent_ecs_auto_sd.yaml` file di dalam wadah agen. CloudWatch

Dua bagian ekspresi reguler ARN penetapan tugas ditentukan:

- Untuk bagian pertama, tugas ECS dengan `javajmx` dalam ARN penetapan tugas ECS difilter untuk pemindaian port kontainer. Jika kontainer-kontainer yang ada dalam tugas ECS membuka port kontainer di 9404 atau 9406, port host yang dipetakan bersama dengan IP privat kontainer digunakan untuk membuat target-target pengekspor Prometheus. Nilai dari `sd_metrics_path` mengatur `__metrics_path__` ke `/metrics`. Jadi CloudWatch agen akan mengikis metrik Prometheus `private_ip:host_port/metrics` dari, metrik yang dikikis dikirim ke aliran log di Log di grup log. `java-prometheus` CloudWatch `/aws/ecs/containerinsights/cluster_name/prometheus`
- Untuk bagian kedua, tugas ECS dengan `appmesh` dalam ARN penetapan tugas ECS mereka dan dengan `version` dari `:23` difilter untuk pemindaian port kontainer. Jika kontainer-kontainer yang bernama `envoy` membuka port kontainer di `9901`, port host yang dipetakan bersama dengan IP privat kontainer digunakan untuk membuat target-target pengekspor Prometheus. Nilai yang ada dalam tugas ECS membuka port kontainer di 9404 atau 9406, port host yang dipetakan bersama dengan IP privat kontainer digunakan untuk membuat target-target pengekspor Prometheus. Nilai dari `sd_metrics_path` mengatur `__metrics_path__` menjadi `/stats/prometheus`. Jadi CloudWatch agen akan mengikis metrik Prometheus `private_ip:host_port/stats/prometheus` dari, dan mengirim metrik yang tergores ke aliran log di Log di grup log. `envoy-prometheus` CloudWatch `/aws/ecs/containerinsights/cluster_name/prometheus`

Contoh 4

```
"ecs_service_discovery": {
  "sd_frequency": "5m",
  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
  "service_name_list_for_tasks": [
    {
      "sd_job_name": "nginx-prometheus",
      "sd_metrics_path": "/metrics",
      "sd_metrics_ports": "9113",
      "sd_service_name_pattern": "^nginx-.*"
```

```
    },
    {
      "sd_job_name": "haproxy-prometheus",
      "sd_metrics_path": "/stats/metrics",
      "sd_container_name_pattern": "^haproxy$",
      "sd_metrics_ports": "8404",
      "sd_service_name_pattern": ".*haproxy-service.*"
    }
  ]
}
```

Contoh ini memungkinkan penemuan layanan berbasis ekspresi reguler nama layanan ECS. CloudWatch Agen akan menanyakan metadata layanan ECS setiap lima menit dan menulis target yang ditemukan ke dalam `/tmp/cwagent_ecs_auto_sd.yaml` file di dalam wadah agen. CloudWatch

Dua bagian ekspresi reguler nama layanan ditentukan:

- Untuk bagian pertama, tugas ECS yang terkait dengan layanan-layanan ECS yang memiliki nama yang cocok dengan ekspresi reguler `^nginx-.*` akan difilter untuk pemindaian port kontainer. Jika kontainer-kontainer yang ada dalam tugas ECS membuka port kontainer di 9113, port host yang dipetakan bersama dengan IP privat kontainer digunakan untuk membuat target-target pengekspor Prometheus. Nilai dari `sd_metrics_path` mengatur `__metrics_path__` menjadi `/metrics`. Jadi CloudWatch agen akan mengikis metrik Prometheus `private_ip:host_port/metrics` dari, dan metrik yang dikikis dikirim ke aliran log di Log di grup log. `nginx-prometheus` CloudWatch `/aws/ecs/containerinsights/cluster_name/prometheus`
- atau bagian kedua, tugas ECS yang terkait dengan layanan-layanan ECS yang memiliki nama yang cocok dengan ekspresi reguler `.*haproxy-service.*` akan difilter untuk pemindaian port kontainer. Jika kontainer-kontainer yang bernama `haproxy` membuka port kontainer di 8404, port host yang dipetakan bersama dengan IP privat kontainer digunakan untuk membuat target-target pengekspor Prometheus. Nilai dari `sd_metrics_path` mengatur `__metrics_path__` menjadi `/stats/metrics`. Jadi CloudWatch agen akan mengikis metrik Prometheus `private_ip:host_port/stats/metrics` dari, dan metrik yang dikikis dikirim ke aliran log di Log di grup log. `haproxy-prometheus` CloudWatch `/aws/ecs/containerinsights/cluster_name/prometheus`

Contoh 5

```
"ecs_service_discovery": {
```

```
  "sd_frequency": "1m30s",
  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
  "docker_label": {
    "sd_port_label": "MY_PROMETHEUS_EXPORTER_PORT_LABEL",
    "sd_metrics_path_label": "MY_PROMETHEUS_METRICS_PATH_LABEL",
    "sd_job_name_label": "MY_PROMETHEUS_METRICS_NAME_LABEL"
  }
  "task_definition_list": [
    {
      "sd_metrics_ports": "9150",
      "sd_task_definition_arn_pattern": "*memcached.*"
    }
  ]
}
```

Contoh ini memungkinkan kedua mode penemuan layanan ECS. CloudWatch Agen akan menanyakan metadata tugas ECS setiap 90 detik dan menulis target yang ditemukan ke dalam `/tmp/cwagent_ecs_auto_sd.yaml` file di dalam wadah agen. CloudWatch

Untuk konfigurasi penemuan layanan berbasis docker:

- Tugas-tugas ECS dengan label docker `MY_PROMETHEUS_EXPORTER_PORT_LABEL` akan difilter untuk pemindaian port Prometheus. Port kontainer Prometheus target ditentukan berdasarkan nilai label `MY_PROMETHEUS_EXPORTER_PORT_LABEL`.
- Nilai label docker `MY_PROMETHEUS_EXPORTER_PORT_LABEL` digunakan untuk `__metrics_path__`. Jika kontainer tidak memiliki label docker ini, maka nilai bawaannya `/metrics` akan digunakan.
- Nilai label docker dari `MY_PROMETHEUS_EXPORTER_PORT_LABEL` akan digunakan sebagai nama pekerjaan. Jika kontainer tidak memiliki label docker ini, maka nama pekerjaan yang ditentukan dalam konfigurasi Prometheus akan digunakan.

Untuk konfigurasi penemuan layanan berbasis ekspresi reguler ARN penetapan tugas ECS:

- Tugas-tugas ECS dengan `memcached` dalam ARN penetapan tugas ECS difilter untuk pemindaian port kontainer. Port kontainer Prometheus target adalah 9150 sebagaimana yang ditentukan oleh `sd_metrics_ports`. Jalur metrik bawaan `/metrics` digunakan. Nama pekerjaan yang ditentukan dalam konfigurasi Prometheus digunakan.

(Opsional) Menyiapkan sampel beban kerja Amazon ECS terkontainer untuk pengujian metrik Prometheus

Untuk menguji dukungan metrik Prometheus CloudWatch di Container Insights, Anda dapat menyiapkan satu atau beberapa beban kerja kontainer berikut. CloudWatch Agen dengan dukungan Prometheus secara otomatis mengumpulkan metrik dari masing-masing beban kerja ini. Untuk melihat metrik-metrik yang dikumpulkan secara bawaan, silakan lihat Metrik Prometheus dikumpulkan oleh agen CloudWatch .

Topik

- Contoh beban kerja App Mesh untuk klaster-klaster Amazon ECS
- Contoh beban kerja Java/JMX untuk klaster-klaster Amazon ECS
- Beban kerja NGINX sampel untuk klaster-klaster Amazon ECS
- Beban kerja NGINX Plus sampel untuk klaster-klaster Amazon ECS
- Tutorial untuk menambahkan sebuah target scraping Prometheus baru: Memcached di Amazon ECS
- Tutorial untuk melakukan scraping pada metrik-metrik Redis Prometheus di Amazon ECS Fargate

Contoh beban kerja App Mesh untuk klaster-klaster Amazon ECS

Untuk mengumpulkan metrik-metrik dari sampel beban kerja Prometheus untuk Amazon ECS, Anda harus menjalankan Wawasan Kontainer di klaster. Untuk informasi tentang cara melakukan instalasi Wawasan Kontainer, silakan lihat Menyiapkan Wawasan Kontainer di Amazon ECS.

Pertama, ikuti panduan ini untuk menerapkan aplikasi warna sampel di klaster Amazon ECS Anda. Setelah selesai, Anda akan memiliki metrik-metrik App Mesh Prometheus di port 9901.

Selanjutnya, ikuti langkah-langkah ini untuk menginstal CloudWatch agen dengan pemantauan Prometheus pada cluster Amazon ECS yang sama tempat Anda menginstal aplikasi warna. Langkah-langkah di bagian ini menginstal CloudWatch agen dalam mode jaringan jembatan.

Variabel lingkungan `ENVIRONMENT_NAME`, `AWS_PROFILE`, dan `AWS_DEFAULT_REGION` yang Anda tetapkan dalam panduan juga akan digunakan dalam langkah-langkah berikut.

Untuk menginstal CloudWatch agen dengan pemantauan Prometheus untuk pengujian

1. Unduh AWS CloudFormation template dengan memasukkan perintah berikut.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/
cwagent-prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-
bridge-host.yaml
```

2. Atur mode jaringan dengan memasukkan perintah berikut.

```
export ECS_CLUSTER_NAME=${ENVIRONMENT_NAME}
export ECS_NETWORK_MODE=bridge
```

3. Buat AWS CloudFormation tumpukan dengan memasukkan perintah berikut.

```
aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=${ECS_CLUSTER_NAME} \
                 ParameterKey=CreateIAMRoles,ParameterValue=True \
                 ParameterKey=ECSNetworkMode,ParameterValue=${ECS_NETWORK_MODE} \
                 ParameterKey=TaskRoleName,ParameterValue=CWAgent-Prometheus-
TaskRole-${ECS_CLUSTER_NAME} \
                 ParameterKey=ExecutionRoleName,ParameterValue=CWAgent-Prometheus-
ExecutionRole-${ECS_CLUSTER_NAME} \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${AWS_DEFAULT_REGION} \
    --profile ${AWS_PROFILE}
```

4. (Opsional) Saat AWS CloudFormation tumpukan dibuat, Anda melihat `CREATE_COMPLETE` pesan. Jika Anda memeriksa status sebelum melihat pesan tersebut, silakan masukkan perintah berikut.

```
aws cloudformation describe-stacks \
--stack-name CWAgent-Prometheus-ECS-${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
--query 'Stacks[0].StackStatus' \
--region ${AWS_DEFAULT_REGION} \
--profile ${AWS_PROFILE}
```

Pemecahan Masalah

Langkah-langkah dalam panduan menggunakan jq untuk melakukan penguraian atas hasil output AWS CLI. Untuk informasi selengkapnya tentang melakukan instalasi jq, silakan lihat jq. Gunakan

perintah berikut untuk mengatur format output bawaan AWS CLI Anda menjadi JSON sehingga jq dapat menguraikannya dengan benar.

```
$ aws configure
```

Saat tanggapan menampilkan `Default output format`, masukkan **`json`**.

Copot pemasangan CloudWatch agen dengan pemantauan Prometheus

Setelah Anda selesai menguji, masukkan perintah berikut untuk menghapus instalan CloudWatch agen dengan menghapus tumpukan. AWS CloudFormation

```
aws cloudformation delete-stack \
--stack-name CWAgent-Prometheus-ECS-${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
--region ${AWS_DEFAULT_REGION} \
--profile ${AWS_PROFILE}
```

Contoh beban kerja Java/JMX untuk klaster-klaster Amazon ECS

JMX Exporter adalah sebuah pengekspor Prometheus resmi yang dapat melakukan scraping dan mengekspos JMX mBeans sebagai metrik-metrik Prometheus. Untuk informasi selengkapnya, silakan lihat prometheus/jmx_exporter.

CloudWatch Agen dengan dukungan Prometheus menggores metrik Java/JMX Prometheus berdasarkan konfigurasi penemuan layanan di cluster Amazon ECS. Anda dapat mengonfigurasi JMX Exporter untuk mengekspos metrik-metrik sebuah pada port atau metrik_path yang berbeda. Jika Anda mengubah port atau jalur, perbarui `ecs_service_discovery` bagian default dalam konfigurasi CloudWatch agen.

Untuk mengumpulkan metrik-metrik dari sampel beban kerja Prometheus untuk Amazon ECS, Anda harus menjalankan Wawasan Kontainer di klaster. Untuk informasi tentang cara melakukan instalasi Wawasan Kontainer, silakan lihat Menyiapkan Wawasan Kontainer di Amazon ECS.

Untuk melakukan instalasi beban kerja sampel Java/JMX untuk klaster-klaster Amazon ECS

1. Ikuti langkah-langkah yang ada di bagian ini untuk membuat citra Docker Anda.

   - Contoh: Citra Docker Aplikasi Jar Java dengan metrik-metrik Prometheus
   - Contoh: Citra Docker Apache Tomcat dengan metrik-metrik Prometheus

2. Tentukan dua label docker berikut di file penetapan tugas Amazon ECS. Kemudian, Anda akan dapat menjalankan penetapan tugas sebagai sebuah layanan Amazon ECS atau tugas Amazon ECS dalam klaster.

   - Tetapkan `ECS_PROMETHEUS_EXPORTER_PORT` untuk menunjuk ke containerPort tempat metrik Prometheus dibuka.
   - Atur `Java_EMF_Metrics` menjadi `true`. CloudWatch Agen menggunakan bendera ini untuk menghasilkan format metrik yang disematkan dalam peristiwa log.

   Berikut ini adalah contohnya:

```
{
  "family": "workload-java-ec2-bridge",
  "taskRoleArn": "{{task-role-arn}}",
  "executionRoleArn": "{{execution-role-arn}}",
  "networkMode": "bridge",
  "containerDefinitions": [
    {
      "name": "tomcat-prometheus-workload-java-ec2-bridge-dynamic-port",
      "image": "your_docker_image_tag_for_tomcat_with_prometheus_metrics",
      "portMappings": [
        {
          "hostPort": 0,
          "protocol": "tcp",
          "containerPort": 9404
        }
      ],
      "dockerLabels": {
        "ECS_PROMETHEUS_EXPORTER_PORT": "9404",
        "Java_EMF_Metrics": "true"
      }
    }
  ],
  "requiresCompatibilities": [
    "EC2"  ],
  "cpu": "256",
  "memory": "512"
  }
```

Pengaturan default CloudWatch agen dalam AWS CloudFormation template memungkinkan penemuan layanan berbasis label docker dan penemuan layanan berbasis ARN definisi tugas. Untuk melihat setelan default ini, lihat baris 65 dari file konfigurasi YAMM CloudWatch agen. Kontainer-kontainer yang memiliki label `ECS_PROMETHEUS_EXPORTER_PORT` akan ditemukan secara otomatis berdasarkan port kontainer yang ditentukan untuk scraping Prometheus.

Pengaturan default CloudWatch agen juga memiliki `metric_declaration` pengaturan untuk java/JMX pada baris 112 dari file yang sama. Semua label docker dari wadah target akan ditambahkan sebagai label tambahan dalam metrik Prometheus dan dikirim ke Log. CloudWatch Untuk kontainer-kontainer Java/JMX yang memiliki label docker `Java_EMF_Metrics="true"`, format metrik tersemat akan dibuat.

Beban kerja NGINX sampel untuk klaster-klaster Amazon ECS

Eksportir NGINX Prometheus dapat melakukan scraping dan mengekspos data NGINX sebagai metrik Prometheus. Contoh ini menggunakan pengekspor tersebut bersama dengan layanan proksi terbalik NGINX untuk Amazon ECS.

Untuk informasi lebih lanjut tentang eksportir Prometheus NGINX, lihat di Github. nginx-prometheus-exporter Untuk informasi selengkapnya tentang proxy terbalik NGINX, lihat ecs-nginx-reverse-proxydi Github.

CloudWatch Agen dengan dukungan Prometheus mengikis metrik NGINX Prometheus berdasarkan konfigurasi penemuan layanan di cluster Amazon ECS. Anda dapat mengonfigurasi NGINX Prometheus Exporter untuk mengekspos metrik-metrik sebuah pada port atau jalur yang berbeda. Jika Anda mengubah port atau jalur, perbarui `ecs_service_discovery` bagian dalam file konfigurasi CloudWatch agen.

Melakukan instalasi beban kerja sampel proksi terbalik NGINX untuk klaster-klaster Amazon ECS

Ikuti langkah-langkah berikut ini untuk melakukan instalasi beban kerja sampel proksi terbalik NGINX.

Membuat citra Docker

Cara membuat citra Docker untuk beban kerja sampel proksi terbalik NGINX

1. Unduh folder berikut dari repo proxy terbalik NGINX: https://github.com/awslabs/ /tree/master/reverse-proxy/. ecs-nginx-reverse-proxy
2. Temukan direktori app dan bangun sebuah citra dari direktori tersebut:

```
docker build -t web-server-app ./path-to-app-directory
```

3. Buat sebuah citra kustom untuk NGINX. Pertama, buat sebuah direktori dengan dua file berikut:

   - Dockerfile sampel:

     ```
     FROM nginx
     COPY nginx.conf /etc/nginx/nginx.conf
     ```

   - `nginx.conf`File, dimodifikasi dari https://github.com/awslabs/ ecs-nginx-reverse-proxy /tree/master/reverse-proxy/:

     ```
     events {
       worker_connections 768;
     }

     http {
       # Nginx will handle gzip compression of responses from the app server
       gzip on;
       gzip_proxied any;
       gzip_types text/plain application/json;
       gzip_min_length 1000;

       server{
         listen 8080;
         location /stub_status {
             stub_status   on;
         }
       }

       server {
         listen 80;

         # Nginx will reject anything not matching /api
         location /api {
           # Reject requests with unsupported HTTP method
           if ($request_method !~ ^(GET|POST|HEAD|OPTIONS|PUT|DELETE)$) {
             return 405;
           }

           # Only requests matching the whitelist expectations will
           # get sent to the application server
     ```

```
        proxy_pass http://app:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_cache_bypass $http_upgrade;
    }
  }
}
```

> **Note**
>
> `stub_status` harus diaktifkan pada port yang sama yang `nginx-prometheus-exporter` dikonfigurasi untuk mengikis metrik-metrik darinya. Dalam penetapan tugas contoh kita, `nginx-prometheus-exporter` dikonfigurasi untuk melakukan scraping pada metrik-metrik dari port 8080.

4. Membangun sebuah citra dari file yang ada dalam direktori baru Anda:

```
docker build -t nginx-reverse-proxy ./path-to-your-directory
```

5. Unggah citra baru Anda ke sebuah repositori citra yang akan digunakan nanti.

Buat penetapan tugas untuk menjalankan NGINX dan aplikasi web server di Amazon ECS

Berikutnya, Anda menyiapkan penetapan tugas.

Penetapan tugas ini memungkinkan pengumpulan dan ekspor metrik-metrik NGINX Prometheus. Kontainer NGINX melacak masukan dari aplikasi, dan membuka data itu ke port 8080, sebagaimana diatur dalam `nginx.conf`. Wadah eksportir prometheus NGINX menggores metrik ini, dan mempostingnya ke port 9113, untuk digunakan di. CloudWatch

Cara menyiapkan penetapan tugas untuk beban kerja Amazon ECS sampel NGINX

1. Buat sebuah penetapan tugas file JSON dengan konten berikut. Ganti *`your-customized-nginx-iamge`*dengan URI gambar untuk image NGINX Anda yang disesuaikan, dan ganti *`your-web-server-app-image`* dengan URI image untuk image aplikasi server web Anda.

```
{
```

```
  "containerDefinitions": [
    {
      "name": "nginx",
      "image": "your-customized-nginx-image",
      "memory": 256,
      "cpu": 256,
      "essential": true,
      "portMappings": [
        {
          "containerPort": 80,
          "protocol": "tcp"
        }
      ],
      "links": [
        "app"
      ]
    },
    {
      "name": "app",
      "image": "your-web-server-app-image",
      "memory": 256,
      "cpu": 256,
      "essential": true
    },
    {
      "name": "nginx-prometheus-exporter",
      "image": "docker.io/nginx/nginx-prometheus-exporter:0.8.0",
      "memory": 256,
      "cpu": 256,
      "essential": true,
      "command": [
        "-nginx.scrape-uri",
        "http://nginx:8080/stub_status"
    ],
    "links":[
      "nginx"
    ],
      "portMappings":[
        {
          "containerPort": 9113,
          "protocol": "tcp"
        }
      ]
    }
```

```
  ],
  "networkMode": "bridge",
  "placementConstraints": [],
  "family": "nginx-sample-stack"
}
```

2. Daftarkan penetapan tugas dengan memasukkan perintah berikut.

```
aws ecs register-task-definition --cli-input-json file://path-to-your-task-definition-json
```

3. Untuk membuat sebuah layanan untuk menjalankan tugas dengan memasukkan perintah berikut:

   Pastikan untuk tidak mengubah nama layanan. Kami akan menjalankan layanan CloudWatch agen menggunakan konfigurasi yang mencari tugas menggunakan pola nama layanan yang memulainya. Misalnya, bagi CloudWatch agen untuk menemukan tugas yang diluncurkan oleh perintah ini, Anda dapat menentukan nilai `sd_service_name_pattern` menjadi`^nginx-service$`. Bagian selanjutnya menyediakan detail yang lebih lengkap.

```
aws ecs create-service \
 --cluster your-cluster-name \
 --service-name nginx-service \
 --task-definition nginx-sample-stack:1 \
 --desired-count 1
```

Konfigurasikan CloudWatch agen untuk mengikis metrik Prometheus NGINX

Langkah terakhir adalah mengonfigurasi CloudWatch agen untuk mengikis metrik NGINX. Dalam contoh ini, CloudWatch agen menemukan tugas melalui pola nama layanan, dan port 9113, di mana eksportir mengekspos metrik prometheus untuk NGINX. Dengan tugas yang ditemukan dan metrik yang tersedia, CloudWatch agen mulai memposting metrik yang dikumpulkan ke aliran log. nginx-prometheus-exporter

Untuk mengonfigurasi CloudWatch agen untuk mengikis metrik NGINX

1. Unduh file YAML versi terbaru yang diperlukan dengan memasukkan perintah berikut.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/
```

```
cwagent-prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-
bridge-host.yaml
```

2. Buka file dengan editor teks, dan temukan konfgurasi CloudWatch agen lengkap di `value` kunci di bagian`resource:CWAgentConfigSSMParameter`. Kemudian, di bagian `ecs_service_discovery`, tambahkan bagian `service_name_list_for_tasks` berikut.

```
"service_name_list_for_tasks": [
  {
    "sd_job_name": "nginx-prometheus-exporter",
    "sd_metrics_path": "/metrics",
    "sd_metrics_ports": "9113",
    "sd_service_name_pattern": "^nginx-service$"
   }
],
```

3. Dalam file yang sama, tambahkan bagian berikut di bagian `metric_declaration` untuk mengizinkan metrik NGINX. Pastikan untuk mengikuti pola indentasi yang ada saat itu.

```
{
  "source_labels": ["job"],
  "label_matcher": ".*nginx.*",
  "dimensions": [["ClusterName", "TaskDefinitionFamily", "ServiceName"]],
  "metric_selectors": [
    "^nginx_.*$"
  ]
},
```

4. Jika Anda belum memiliki CloudWatch agen yang digunakan di cluster ini, lewati ke langkah 8.

   Jika Anda sudah memiliki CloudWatch agen yang disebarkan di cluster Amazon ECS dengan menggunakan AWS CloudFormation, Anda dapat membuat set perubahan dengan memasukkan perintah berikut:

```
ECS_CLUSTER_NAME=your_cluster_name
AWS_REGION=your_aws_region
ECS_NETWORK_MODE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name
```

```
aws cloudformation create-change-set --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSNetworkMode,ParameterValue=$ECS_NETWORK_MODE \
                 ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION \
    --change-set-name nginx-scraping-support
```

5. Buka AWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation.
6. Mengungkap changeset yang baru dibuat. nginx-scraping-support Anda akan melihat satu perubahan diterapkan ke sumber daya CW AgentConfig SSMParameter. Jalankan changeset dan restart tugas CloudWatch agen dengan memasukkan perintah berikut:

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 0 \
--service cwagent-prometheus-replica-service-EC2-$ECS_NETWORK_MODE \
--region $AWS_REGION
```

7. Tunggu sekitar 10 detik, kemudian masukkan perintah berikut.

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 1 \
--service cwagent-prometheus-replica-service-EC2-$ECS_NETWORK_MODE \
--region $AWS_REGION
```

8. Jika Anda menginstal CloudWatch agen dengan pengumpulan metrik Prometheus di cluster untuk pertama kalinya, masukkan perintah berikut.

```
ECS_CLUSTER_NAME=your_cluster_name
AWS_REGION=your_aws_region
ECS_NETWORK_MODE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
```

```
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSNetworkMode,ParameterValue=$ECS_NETWORK_MODE \
                 ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION
```

Menampilkan metrik dan log NGINX Anda

Sekarang Anda dapat menampilkan metrik NGINX yang sedang dikumpulkan.

Cara menampilkan metrik untuk sampel beban kerja NGINX Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di Wilayah tempat klaster Anda berjalan, pilih Metrik pada panel navigasi yang ada sebelah kiri. Temukan namespace ContainerInsights/Prometheus untuk melihat metriknya.
3. Untuk melihat peristiwa CloudWatch Log, pilih Grup log di panel navigasi. *Peristiwa ada di grup log **/aws/containerinsights/ your_cluster_name /prometheus**, di aliran log. nginx-prometheus-exporter*

Beban kerja NGINX Plus sampel untuk klaster-klaster Amazon ECS

NGINX Plus adalah NGINX versi komersial. Anda harus memiliki sebuah lisensi untuk menggunakannya. Untuk informasi selengkapnya, silakan lihat NGINX Plus

Eksportir NGINX Prometheus dapat melakukan scraping dan mengekspos data NGINX sebagai metrik Prometheus. Contoh ini menggunakan pengekspor tersebut bersama dengan layanan proksi terbalik NGINX Plus untuk Amazon ECS.

Untuk informasi lebih lanjut tentang eksportir Prometheus NGINX, lihat di Github. nginx-prometheus-exporter Untuk informasi selengkapnya tentang proxy terbalik NGINX, lihat ecs-nginx-reverse-proxydi Github.

CloudWatch Agen dengan dukungan Prometheus menggores metrik NGINX Plus Prometheus berdasarkan konfigurasi penemuan layanan di cluster Amazon ECS. Anda dapat mengonfigurasi NGINX Prometheus Exporter untuk mengekspos metrik-metrik sebuah pada port atau jalur yang

berbeda. Jika Anda mengubah port atau jalur, perbarui `ecs_service_discovery` bagian dalam file konfigurasi CloudWatch agen.

Melakukan instalasi beban kerja sampel proksi terbalik NGINX Plus untuk klaster-klaster Amazon ECS

Ikuti langkah-langkah berikut ini untuk melakukan instalasi beban kerja sampel proksi terbalik NGINX.

Membuat citra Docker

Cara membuat citra Docker untuk beban kerja sampel proksi terbalik NGINX Plus

1. Unduh folder berikut dari repo proxy terbalik NGINX: https://github.com/awslabs/ /tree/master/reverse-proxy/. ecs-nginx-reverse-proxy
2. Temukan direktori `app` dan bangun sebuah citra dari direktori tersebut:

   ```
   docker build -t web-server-app ./path-to-app-directory
   ```

3. Buat sebuah citra kustom untuk NGINX Plus. Sebelum Anda dapat membangun citra untuk NGINX Plus, Anda harus mendapatkan kunci bernama `nginx-repo.key` dan sertifikat SSL `nginx-repo.crt` untuk NGINX Plus berlisensi Anda. Buat sebuah direktori dan simpan file `nginx-repo.key` dan `nginx-repo.crt` di dalamnya.

   Dalam direktori yang baru saja Anda buat, buatlah dua file berikut:

   - Sebuah Dockerfile sampel dengan konten berikut. File docker ini diadopsi dari file sampel yang disediakan di https://docs.nginx.com/nginx/admin-guide/installing-nginx/installing-nginx-docker/#docker_plus_image. Perubahan penting yang kita lakukan adalah kita memuat file terpisah, yang disebut `nginx.conf`, yang akan dibuat pada langkah berikutnya.

     ```
     FROM debian:buster-slim

     LABEL maintainer="NGINX Docker Maintainers <docker-maint@nginx.com>“

     # Define NGINX versions for NGINX Plus and NGINX Plus modules
     # Uncomment this block and the versioned nginxPackages block in the main RUN
     # instruction to install a specific release
     # ENV NGINX_VERSION 21
     # ENV NJS_VERSION 0.3.9
     # ENV PKG_RELEASE 1~buster
     ```

```
# Download certificate and key from the customer portal (https://cs.nginx.com
 (https://cs.nginx.com/))
# and copy to the build context
COPY nginx-repo.crt /etc/ssl/nginx/
COPY nginx-repo.key /etc/ssl/nginx/
# COPY nginx.conf /etc/ssl/nginx/nginx.conf

RUN set -x \
# Create nginx user/group first, to be consistent throughout Docker variants
&& addgroup --system --gid 101 nginx \
&& adduser --system --disabled-login --ingroup nginx --no-create-home --home /
nonexistent --gecos "nginx user" --shell /bin/false --uid 101 nginx \
&& apt-get update \
&& apt-get install --no-install-recommends --no-install-suggests -y ca-
certificates gnupg1 \
&& \
NGINX_GPGKEY=573BFD6B3D8FBC641079A6ABABF5BD827BD9BF62; \
found=''; \
for server in \
ha.pool.sks-keyservers.net (http://ha.pool.sks-keyservers.net/) \
hkp://keyserver.ubuntu.com:80 \
hkp://p80.pool.sks-keyservers.net:80 \
pgp.mit.edu (http://pgp.mit.edu/) \
; do \
echo "Fetching GPG key $NGINX_GPGKEY from $server"; \
apt-key adv --keyserver "$server" --keyserver-options timeout=10 --recv-keys
 "$NGINX_GPGKEY" && found=yes && break; \
done; \
test -z "$found" && echo >&2 "error: failed to fetch GPG key $NGINX_GPGKEY" &&
 exit 1; \
apt-get remove --purge --auto-remove -y gnupg1 && rm -rf /var/lib/apt/lists/* \
# Install the latest release of NGINX Plus and/or NGINX Plus modules
# Uncomment individual modules if necessary
# Use versioned packages over defaults to specify a release
&& nginxPackages=" \
nginx-plus \
# nginx-plus=${NGINX_VERSION}-${PKG_RELEASE} \
# nginx-plus-module-xslt \
# nginx-plus-module-xslt=${NGINX_VERSION}-${PKG_RELEASE} \
# nginx-plus-module-geoip \
# nginx-plus-module-geoip=${NGINX_VERSION}-${PKG_RELEASE} \
# nginx-plus-module-image-filter \
# nginx-plus-module-image-filter=${NGINX_VERSION}-${PKG_RELEASE} \
# nginx-plus-module-perl \
```

```
# nginx-plus-module-perl=${NGINX_VERSION}-${PKG_RELEASE} \
# nginx-plus-module-njs \
# nginx-plus-module-njs=${NGINX_VERSION}+${NJS_VERSION}-${PKG_RELEASE} \
" \
&& echo "Acquire::https::plus-pkgs.nginx.com::Verify-Peer \"true\";" >> /etc/apt/
apt.conf.d/90nginx \
&& echo "Acquire::https::plus-pkgs.nginx.com::Verify-Host \"true\";" >> /etc/apt/
apt.conf.d/90nginx \
&& echo "Acquire::https::plus-pkgs.nginx.com::SslCert \"/etc/ssl/nginx/nginx-
repo.crt\";" >> /etc/apt/apt.conf.d/90nginx \
&& echo "Acquire::https::plus-pkgs.nginx.com::SslKey \"/etc/ssl/nginx/nginx-
repo.key\";" >> /etc/apt/apt.conf.d/90nginx \
&& printf "deb https://plus-pkgs.nginx.com/debian buster nginx-plus\n" > /etc/
apt/sources.list.d/nginx-plus.list \
&& apt-get update \
&& apt-get install --no-install-recommends --no-install-suggests -y \
$nginxPackages \
gettext-base \
curl \
&& apt-get remove --purge --auto-remove -y && rm -rf /var/lib/apt/lists/* /etc/
apt/sources.list.d/nginx-plus.list \
&& rm -rf /etc/apt/apt.conf.d/90nginx /etc/ssl/nginx

# Forward request logs to Docker log collector
RUN ln -sf /dev/stdout /var/log/nginx/access.log \
&& ln -sf /dev/stderr /var/log/nginx/error.log

COPY nginx.conf /etc/nginx/nginx.conf

EXPOSE 80

STOPSIGNAL SIGTERM

CMD ["nginx", "-g", "daemon off;"]
```

- Sebuah `nginx.conf` file, dimodifikasi dari [https://github.com/awslabs/ ecs-nginx-reverse-proxy /tree/master/reverse-proxy/nginx](https://github.com/awslabs/ecs-nginx-reverse-proxy/tree/master/reverse-proxy/nginx).

```
events {
  worker_connections 768;
}

http {
```

```
# Nginx will handle gzip compression of responses from the app server
gzip on;
gzip_proxied any;
gzip_types text/plain application/json;
gzip_min_length 1000;

upstream backend {
  zone name 10m;
  server app:3000    weight=2;
  server app2:3000    weight=1;
}

server{
  listen 8080;
  location /api {
    api write=on;
  }
}

match server_ok {
  status 100-599;
}

server {
  listen 80;
  status_zone zone;
  # Nginx will reject anything not matching /api
  location /api {
    # Reject requests with unsupported HTTP method
    if ($request_method !~ ^(GET|POST|HEAD|OPTIONS|PUT|DELETE)$) {
      return 405;
    }

    # Only requests matching the whitelist expectations will
    # get sent to the application server
    proxy_pass http://backend;
    health_check uri=/lorem-ipsum match=server_ok;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_cache_bypass $http_upgrade;
  }
```

```
  }
}
```

4. Membangun sebuah citra dari file yang ada dalam direktori baru Anda:

```
docker build -t nginx-plus-reverse-proxy ./path-to-your-directory
```

5. Unggah citra baru Anda ke sebuah repositori citra yang akan digunakan nanti.

Buat penetapan tugas untuk menjalankan NGINX Plus dan aplikasi web server di Amazon ECS

Berikutnya, Anda menyiapkan penetapan tugas.

Penetapan tugas ini memungkinkan dilakukannya pengumpulan dan ekspor metrik NGINX Prometheus. Kontainer NGINX melacak masukan dari aplikasi, dan membuka data itu ke port 8080, sebagaimana diatur dalam `nginx.conf`. Wadah eksportir prometheus NGINX menggores metrik ini, dan mempostingnya ke port 9113, untuk digunakan di. CloudWatch

Cara menyiapkan penetapan tugas untuk beban kerja Amazon ECS sampel NGINX

1. Buat sebuah penetapan tugas file JSON dengan konten berikut. Ganti *your-customized-nginx-plus-image* dengan URI image untuk image NGINX Plus Anda yang disesuaikan, dan ganti *your-web-server-app-image dengan URI image* untuk image aplikasi server web Anda.

```
{
  "containerDefinitions": [
    {
      "name": "nginx",
      "image": "your-customized-nginx-plus-image",
      "memory": 256,
      "cpu": 256,
      "essential": true,
      "portMappings": [
        {
          "containerPort": 80,
          "protocol": "tcp"
        }
      ],
      "links": [
        "app",
        "app2"
```

```
      ]
    },
    {
      "name": "app",
      "image": "your-web-server-app-image",
      "memory": 256,
      "cpu": 128,
      "essential": true
    },
    {
      "name": "app2",
      "image": "your-web-server-app-image",
      "memory": 256,
      "cpu": 128,
      "essential": true
    },
    {
      "name": "nginx-prometheus-exporter",
      "image": "docker.io/nginx/nginx-prometheus-exporter:0.8.0",
      "memory": 256,
      "cpu": 256,
      "essential": true,
      "command": [
        "-nginx.plus",
        "-nginx.scrape-uri",
         "http://nginx:8080/api"
    ],
    "links":[
      "nginx"
    ],
      "portMappings":[
        {
          "containerPort": 9113,
          "protocol": "tcp"
        }
      ]
    }
  ],
  "networkMode": "bridge",
  "placementConstraints": [],
  "family": "nginx-plus-sample-stack"
}
```

2. Mendaftarkan penetapan tugas:

```
aws ecs register-task-definition --cli-input-json file://path-to-your-task-definition-json
```

3. Untuk membuat sebuah layanan untuk menjalankan tugas dengan memasukkan perintah berikut:

```
aws ecs create-service \
 --cluster your-cluster-name \
 --service-name nginx-plus-service \
 --task-definition nginx-plus-sample-stack:1 \
 --desired-count 1
```

Pastikan untuk tidak mengubah nama layanan. Kami akan menjalankan layanan CloudWatch agen menggunakan konfigurasi yang mencari tugas menggunakan pola nama layanan yang memulainya. Misalnya, bagi CloudWatch agen untuk menemukan tugas yang diluncurkan oleh perintah ini, Anda dapat menentukan nilai `sd_service_name_pattern` menjadi`^nginx-plus-service$`. Bagian selanjutnya menyediakan detail yang lebih lengkap.

Konfigurasikan CloudWatch agen untuk mengikis metrik Prometheus NGINX Plus

Langkah terakhir adalah mengonfigurasi CloudWatch agen untuk mengikis metrik NGINX. Dalam contoh ini, CloudWatch agen menemukan tugas melalui pola nama layanan, dan port 9113, di mana eksportir mengekspos metrik prometheus untuk NGINX. Dengan tugas yang ditemukan dan metrik yang tersedia, CloudWatch agen mulai memposting metrik yang dikumpulkan ke aliran log. nginx-prometheus-exporter

Untuk mengonfigurasi CloudWatch agen untuk mengikis metrik NGINX

1. Unduh file YAML versi terbaru yang diperlukan dengan memasukkan perintah berikut.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-bridge-host.yaml
```

2. Buka file dengan editor teks, dan temukan konfgurasi CloudWatch agen lengkap di `value` kunci di bagian`resource:CWAgentConfigSSMParameter`. Kemudian, di bagian `ecs_service_discovery`, tambahkan bagian `service_name_list_for_tasks` berikut.

```
"service_name_list_for_tasks": [
  {
    "sd_job_name": "nginx-plus-prometheus-exporter",
    "sd_metrics_path": "/metrics",
    "sd_metrics_ports": "9113",
    "sd_service_name_pattern": "^nginx-plus.*"
   }
],
```

3. Dalam file yang sama, tambahkan bagian berikut di bagian `metric_declaration` untuk mengizinkan metrik NGINX Plus. Pastikan untuk mengikuti pola indentasi yang ada saat itu.

```
{
  "source_labels": ["job"],
  "label_matcher": "^nginx-plus.*",
  "dimensions": [["ClusterName", "TaskDefinitionFamily", "ServiceName"]],
  "metric_selectors": [
    "^nginxplus_connections_accepted$",
    "^nginxplus_connections_active$",
    "^nginxplus_connections_dropped$",
    "^nginxplus_connections_idle$",
    "^nginxplus_http_requests_total$",
    "^nginxplus_ssl_handshakes$",
    "^nginxplus_ssl_handshakes_failed$",
    "^nginxplus_up$",
    "^nginxplus_upstream_server_health_checks_fails$"
  ]
},
{
  "source_labels": ["job"],
  "label_matcher": "^nginx-plus.*",
  "dimensions": [["ClusterName", "TaskDefinitionFamily", "ServiceName",
 "upstream"]],
  "metric_selectors": [
    "^nginxplus_upstream_server_response_time$"
  ]
},
{
  "source_labels": ["job"],
  "label_matcher": "^nginx-plus.*",
  "dimensions": [["ClusterName", "TaskDefinitionFamily", "ServiceName", "code"]],
  "metric_selectors": [
```

```
    "^nginxplus_upstream_server_responses$",
    "^nginxplus_server_zone_responses$"
  ]
},
```

4. Jika Anda belum memiliki CloudWatch agen yang digunakan di cluster ini, lewati ke langkah 8.

   Jika Anda sudah memiliki CloudWatch agen yang disebarkan di cluster Amazon ECS dengan menggunakan AWS CloudFormation, Anda dapat membuat set perubahan dengan memasukkan perintah berikut:

```
ECS_CLUSTER_NAME=your_cluster_name
AWS_REGION=your_aws_region
ECS_NETWORK_MODE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-change-set --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSNetworkMode,ParameterValue=$ECS_NETWORK_MODE \
                 ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION \
    --change-set-name nginx-plus-scraping-support
```

5. Buka AWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation.

6. Mengungkap changeset yang baru dibuat. nginx-plus-scraping-support Anda akan melihat satu perubahan diterapkan ke sumber daya CW AgentConfig SSMParameter. Jalankan changeset dan tahan tugas CloudWatch agen dengan memasukkan perintah berikut:

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 0 \
--service cwagent-prometheus-replica-service-EC2-$ECS_NETWORK_MODE \
--region $AWS_REGION
```

7. Tunggu sekitar 10 detik, kemudian masukkan perintah berikut.

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 1 \
--service cwagent-prometheus-replica-service-EC2-$ECS_NETWORK_MODE \
--region $AWS_REGION
```

8. Jika Anda menginstal CloudWatch agen dengan pengumpulan metrik Prometheus di cluster untuk pertama kalinya, masukkan perintah berikut.

```
ECS_CLUSTER_NAME=your_cluster_name
AWS_REGION=your_aws_region
ECS_NETWORK_MODE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSNetworkMode,ParameterValue=$ECS_NETWORK_MODE \
                 ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION
```

Menampilkan metrik dan log NGINX Plus Anda

Sekarang Anda dapat menampilkan metrik NGINX Plus yang sedang dikumpulkan.

Cara menampilkan metrik untuk sampel beban kerja NGINX Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di Wilayah tempat klaster Anda berjalan, pilih Metrik pada panel navigasi yang ada sebelah kiri. Temukan namespace ContainerInsights/Prometheus untuk melihat metriknya.
3. Untuk melihat peristiwa CloudWatch Log, pilih Grup log di panel navigasi. *Peristiwa ada di grup log* ***/aws/containerinsights/ your_cluster_name /prometheus****, di aliran log. nginx-plus-prometheus-exporter*

Tutorial untuk menambahkan sebuah target scraping Prometheus baru: Memcached di Amazon ECS

Tutorial ini akan memberikan Anda pengenalan langsung tentang cara melakukan scraping pada metrik Prometheus dari aplikasi Memcached sampel di klaster Amazon ECS dengan tipe peluncuran EC2. Target eksportir Prometheus Memcached akan ditemukan secara otomatis oleh agen oleh penemuan layanan berbasis definisi tugas ECS. CloudWatch

Memcached adalah sebuah sistem cache memori terdistribusi serba guna. Memcached ini sering digunakan untuk mempercepat situs web yang didorong oleh basis data dinamis dengan menyimpan data dan objek di RAM untuk mengurangi jumlah sumber data eksternal (seperti basis data atau API) yang harus dibaca. Untuk informasi selengkapnya, silakan lihat Apa itu Memcached?

memchached_exporter (Apache License 2.0) adalah salah satu pengekspor resmi Prometheus. Secara bawaan, memcache_exporter berfungsi pada port 0.0.0.0:9150 di `/metrics`.

Citra Docker dalam dua repositori Docker Hub berikut digunakan dalam tutorial ini:

- Memcached
- prom/memcached-exporter

Prasyarat

Untuk mengumpulkan metrik-metrik dari sampel beban kerja Prometheus untuk Amazon ECS, Anda harus menjalankan Wawasan Kontainer di klaster. Untuk informasi tentang cara melakukan instalasi Wawasan Kontainer, silakan lihat Menyiapkan Wawasan Kontainer di Amazon ECS.

Topik

- Menetapkan variabel lingkungan klaster Amazon ECS EC2
- Instal beban kerja Memcached sampel
- Konfigurasikan CloudWatch agen untuk mengikis metrik Prometheus Memcached
- Menampilkan metrik-metrik Memcached Anda

Menetapkan variabel lingkungan klaster Amazon ECS EC2

Untuk mengatur variabel lingkungan klaster Amazon ECS EC2

1. Instal CLI Amazon ECS jika Anda belum melakukan instalasinya. Untuk informasi selengkapnya, silakan lihat Menginstal CLI Amazon ECS.

2. Atur nama klaster dan Wilayah Amazon ECS baru. Sebagai contoh:

```
ECS_CLUSTER_NAME=ecs-ec2-memcached-tutorial
AWS_DEFAULT_REGION=ca-central-1
```

3. (Opsional) Jika Anda belum memiliki cluster Amazon ECS dengan tipe peluncuran EC2 tempat Anda ingin menginstal sampel beban kerja dan CloudWatch agen Memcached, Anda dapat membuatnya dengan memasukkan perintah berikut.

```
ecs-cli up --capability-iam --size 1 \
--instance-type t3.medium \
--cluster $ECS_CLUSTER_NAME \
--region $AWS_REGION
```

Hasil yang diharapkan dari perintah ini adalah sebagai berikut:

```
WARN[0000] You will not be able to SSH into your EC2 instances without a key pair.
INFO[0000] Using recommended Amazon Linux 2 AMI with ECS Agent 1.44.4 and Docker
 version 19.03.6-ce
INFO[0001] Created cluster                               cluster=ecs-ec2-memcached-
tutorial region=ca-central-1
INFO[0002] Waiting for your cluster resources to be created...
INFO[0002] Cloudformation stack status
 stackStatus=CREATE_IN_PROGRESS
INFO[0063] Cloudformation stack status
 stackStatus=CREATE_IN_PROGRESS
INFO[0124] Cloudformation stack status
 stackStatus=CREATE_IN_PROGRESS
VPC created: vpc-xxxxxxxxxxxxxxxxx
Security Group created: sg-xxxxxxxxxxxxxxxxx
Subnet created: subnet-xxxxxxxxxxxxxxxxx
Subnet created: subnet-xxxxxxxxxxxxxxxxx
Cluster creation succeeded.
```

Instal beban kerja Memcached sampel

Untuk melakukan instalasi beban kerja Memcached sampel yang membuka metrik Prometheus

1. Unduh AWS CloudFormation template Memcached dengan memasukkan perintah berikut.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/
cwagent-prometheus/sample_traffic/memcached/memcached-traffic-sample.yaml
```

2. Tetapkan nama-nama peran IAM yang akan dibuat untuk Memcached dengan memasukkan perintah-perintah berikut.

```
MEMCACHED_ECS_TASK_ROLE_NAME=memcached-prometheus-demo-ecs-task-role-name
MEMCACHED_ECS_EXECUTION_ROLE_NAME=memcached-prometheus-demo-ecs-execution-role-name
```

3. Instal beban kerja Memcached sampel dengan memasukkan perintah berikut. Sampel ini akan melakukan instalasi beban kerja di mode jaringan `host`.

```
MEMCACHED_ECS_NETWORK_MODE=host

aws cloudformation create-stack --stack-name Memcached-Prometheus-Demo-ECS-
$ECS_CLUSTER_NAME-EC2-$MEMCACHED_ECS_NETWORK_MODE \
    --template-body file://memcached-traffic-sample.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=ECSNetworkMode,ParameterValue=
$MEMCACHED_ECS_NETWORK_MODE \
                 ParameterKey=TaskRoleName,ParameterValue=
$MEMCACHED_ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$MEMCACHED_ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION
```

AWS CloudFormation Tumpukan menciptakan empat sumber daya:

- Satu peran tugas ECS
- Satu peran eksekusi tugas ECS
- Satu penetapan tugas Memcached
- Satu layanan Memcached

Dalam penetapan tugas Memcached, ada dua kontainer yang ditentukan:

- Kontainer utama menjalankan sebuah aplikasi Memcached sederhana dan membuka port 11211 agar bisa diakses.
- Kontainer lainnya yang menjalankan proses pengekspor Redis untuk membuka metrik Prometheus di port 9150. Ini adalah wadah yang akan ditemukan dan dikikis oleh agen. CloudWatch

Konfigurasikan CloudWatch agen untuk mengikis metrik Prometheus Memcached

Untuk mengonfigurasi CloudWatch agen untuk mengikis metrik Prometheus Memcached

1. Unduh `cwagent-ecs-prometheus-metric-for-awsvpc.yaml` versi terbaru dengan memasukkan perintah berikut.

   ```
   curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-awsvpc.yaml
   ```

2. Buka file dengan editor teks, dan temukan konfigurasi CloudWatch agen lengkap di belakang `value` kunci di `resource:CWAgentConfigSSMParameter` bagian.

   Kemudian, di bagian `ecs_service_discovery`, tambahkan konfigurasi berikut ke bagian `task_definition_list`.

   ```
   {
       "sd_job_name": "ecs-memcached",
       "sd_metrics_ports": "9150",
       "sd_task_definition_arn_pattern": ".*:task-definition/memcached-prometheus-demo.*:[0-9]+"
   },
   ```

   Untuk bagian `metric_declaration`, pengaturan bawaan tidak mengizinkan metrik Memcached. Tambahkan bagian berikut untuk mengizinkan metrik Memcached. Pastikan untuk mengikuti pola indentasi yang ada.

   ```
   {
     "source_labels": ["container_name"],
     "label_matcher": "memcached-exporter-.*",
     "dimensions": [["ClusterName", "TaskDefinitionFamily"]],
     "metric_selectors": [
       "^memcached_current_(bytes|items|connections)$",
   ```

```
    "^memcached_items_(reclaimed|evicted)_total$",
    "^memcached_(written|read)_bytes_total$",
    "^memcached_limit_bytes$",
    "^memcached_commands_total$"
  ]
},
{
  "source_labels": ["container_name"],
  "label_matcher": "memcached-exporter-.*",
  "dimensions": [["ClusterName", "TaskDefinitionFamily","status","command"],
 ["ClusterName", "TaskDefinitionFamily","command"]],
  "metric_selectors": [
    "^memcached_commands_total$"
  ]
},
```

3. Jika Anda sudah memiliki CloudWatch agen yang digunakan di cluster Amazon ECS oleh AWS CloudFormation, Anda dapat membuat set perubahan dengan memasukkan perintah berikut.

```
ECS_NETWORK_MODE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-change-set --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSNetworkMode,ParameterValue=$ECS_NETWORK_MODE \
                 ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION \
    --change-set-name memcached-scraping-support
```

4. Buka AWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation.
5. Meninjau changeset `memcached-scraping-support` yang baru saja Anda buat. Anda seharusnya melihat satu perubahan diterapkan ke sumber daya `CWAgentConfigSSMParameter`. Jalankan changeset dan restart tugas CloudWatch agen dengan memasukkan perintah berikut.

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 0 \
--service cwagent-prometheus-replica-service-EC2-$ECS_NETWORK_MODE \
--region $AWS_REGION
```

6. Tunggu sekitar 10 detik, kemudian masukkan perintah berikut.

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 1 \
--service cwagent-prometheus-replica-service-EC2-$ECS_NETWORK_MODE \
--region $AWS_REGION
```

7. Jika Anda menginstal CloudWatch agen dengan pengumpulan metrik Prometheus untuk cluster untuk pertama kalinya, masukkan perintah berikut:

```
ECS_NETWORK_MODEE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSNetworkMode,ParameterValue=$ECS_NETWORK_MODE \
                 ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION
```

Menampilkan metrik-metrik Memcached Anda

Tutorial ini mengirimkan metrik berikut ke namespace ContainerInsightsECS//Prometheus di. CloudWatch Anda dapat menggunakan CloudWatch konsol untuk melihat metrik di namespace tersebut.

Nama metrik	Dimensi
memcached_current_items	ClusterName , TaskDefinitionFamily
memcached_current_connections	ClusterName , TaskDefinitionFamily
memcached_limit_bytes	ClusterName , TaskDefinitionFamily
memcached_current_bytes	ClusterName , TaskDefinitionFamily
memcached_written_bytes_total	ClusterName , TaskDefinitionFamily
memcached_read_bytes_total	ClusterName , TaskDefinitionFamily
memcached_items_evicted_total	ClusterName , TaskDefinitionFamily
memcached_items_reclaimed_total	ClusterName , TaskDefinitionFamily
memcached_commands_total	ClusterName , TaskDefinitionFamily ClusterName , TaskDefinitionFamily, perintah ClusterName , TaskDefinitionFamily, status, perintah

> **Note**
>
> Nilai dari dimensi perintah bisa berupa: `delete`, `get`, `cas`, `set`, `decr`, `touch`, `incr`, atau `flush`.
> Nilai dari dimensi status bisa berupa `hit`, `miss`, atau `badval`.

Anda juga dapat membuat CloudWatch dasbor untuk metrik Prometheus Memcached Anda.

Cara membuat sebuah dasbor untuk metrik-metrik Memcached Prometheus

1. Buat variabel lingkungan, yang menggantikan nilai di bawah ini untuk menyesuaikan dengan deployment Anda.

```
DASHBOARD_NAME=your_memcached_cw_dashboard_name
ECS_TASK_DEF_FAMILY=memcached-prometheus-demo-$ECS_CLUSTER_NAME-EC2-
$MEMCACHED_ECS_NETWORK_MOD
```

2. Masukkan perintah berikut untuk membuat dasbor tersebut.

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/ecs-task-definition-templates/deployment-mode/
replica-service/cwagent-prometheus/sample_cloudwatch_dashboards/memcached/
cw_dashboard_memcached.json \
| sed "s/{{YOUR_AWS_REGION}}/$AWS_REGION/g" \
| sed "s/{{YOUR_CLUSTER_NAME}}/$ECS_CLUSTER_NAME/g" \
| sed "s/{{YOUR_TASK_DEF_FAMILY}}/$ECS_TASK_DEF_FAMILY/g" \
| xargs -0 aws cloudwatch put-dashboard --dashboard-name ${DASHBOARD_NAME} --region
 $AWS_REGION --dashboard-body
```

Tutorial untuk melakukan scraping pada metrik-metrik Redis Prometheus di Amazon ECS Fargate

Tutorial ini menyediakan Anda dengan pengenalan langsung untuk melakukan scraping pada metrik-metrik Prometheus dari sebuah aplikasi Redis sampel dalam sebuah klaster Amazon ECS Fargate. Target eksportir Redis Prometheus akan ditemukan secara otomatis oleh CloudWatch agen dengan dukungan metrik Prometheus berdasarkan label docker kontainer.

Redis (https://redis.io/) adalah sebuah penyimpanan struktur data di dalam memori yang bersifat sumber terbuka (dilisensikan BSD), yang digunakan sebagai basis data, cache, dan broker pesan. Untuk informasi selengkapnya, silakan lihat redis.

redis_exporter (dilisensikan di bawah Lisensi MIT) digunakan untuk mengekspos metrik-metrik Prometheus Redis pada port yang ditentukan (bawaan: 0.0.0.0:9121) Untuk informasi selengkapnya, silakan lihat redis_exporter.

Citra Docker dalam dua repositori Docker Hub berikut digunakan dalam tutorial ini:

- redis
- redis_exporter

Prasyarat

Untuk mengumpulkan metrik-metrik dari sampel beban kerja Prometheus untuk Amazon ECS, Anda harus menjalankan Wawasan Kontainer di klaster. Untuk informasi tentang cara melakukan instalasi Wawasan Kontainer, silakan lihat Menyiapkan Wawasan Kontainer di Amazon ECS.

Topik

- Menetapkan variabel lingkungan klaster Amazon ECS Fargate
- Menetapkan variabel lingkungan jaringan untuk klaster Amazon ECS Fargate
- Instal beban kerja Redis sampel
- Konfigurasikan CloudWatch agen untuk mengikis metrik Redis Prometheus
- Menampilkan metrik Redis

Menetapkan variabel lingkungan klaster Amazon ECS Fargate

Cara menetapkan variabel lingkungan klaster Amazon ECS Fargate

1. Instal CLI Amazon ECS jika Anda belum melakukan instalasinya. Untuk informasi selengkapnya, silakan lihat Menginstal CLI Amazon ECS.
2. Atur nama klaster dan Wilayah Amazon ECS baru. Sebagai contoh:

```
ECS_CLUSTER_NAME=ecs-fargate-redis-tutorial
AWS_DEFAULT_REGION=ca-central-1
```

3. (Opsional) Jika Anda belum memiliki cluster Amazon ECS Fargate tempat Anda ingin menginstal contoh beban kerja CloudWatch dan agen Redis, Anda dapat membuatnya dengan memasukkan perintah berikut.

```
ecs-cli up --capability-iam \
--cluster $ECS_CLUSTER_NAME \
--launch-type FARGATE \
--region $AWS_DEFAULT_REGION
```

Hasil yang diharapkan dari perintah ini adalah sebagai berikut:

```
INFO[0000] Created cluster   cluster=ecs-fargate-redis-tutorial region=ca-central-1
INFO[0001] Waiting for your cluster resources to be created...
INFO[0001] Cloudformation stack status   stackStatus=CREATE_IN_PROGRESS
VPC created: vpc-xxxxxxxxxxxxxxxx
Subnet created: subnet-xxxxxxxxxxxxxxxx
Subnet created: subnet-xxxxxxxxxxxxxxxx
Cluster creation succeeded.
```

Menetapkan variabel lingkungan jaringan untuk klaster Amazon ECS Fargate

Untuk menetapkan variabel lingkungan jaringan untuk klaster Amazon ECS Fargate

1. Atur VPC dan ID subnet Anda dari klaster Amazon ECS. Jika Anda membuat sebuah klaster baru dalam prosedur sebelumnya, maka Anda akan melihat nilai-nilai ini dalam hasil perintah akhir. Jika tidak, Anda harus menggunakan ID klaster yang ada yang akan Anda gunakan dengan Redis.

   ```
   ECS_CLUSTER_VPC=vpc-xxxxxxxxxxxxxxxx
   ECS_CLUSTER_SUBNET_1=subnet-xxxxxxxxxxxxxxxx
   ECS_CLUSTER_SUBNET_2=subnet-xxxxxxxxxxxxxxxx
   ```

2. Dalam tutorial ini, kita akan menginstal aplikasi Redis dan CloudWatch agen di grup keamanan default dari VPC cluster Amazon ECS. Grup keamanan default memungkinkan semua koneksi jaringan dalam grup keamanan yang sama sehingga CloudWatch agen dapat mengikis metrik Prometheus yang terpapar pada wadah Redis. Dalam lingkungan produksi nyata, Anda mungkin ingin membuat grup keamanan khusus untuk aplikasi dan CloudWatch agen Redis dan menetapkan izin khusus untuk mereka.

   Masukkan perintah berikut untuk mendapatkan ID grup keamanan bawaan.

   ```
   aws ec2 describe-security-groups \
   --filters Name=vpc-id,Values=$ECS_CLUSTER_VPC  \
   ```

```
--region $AWS_DEFAULT_REGION
```

Kemudian atur variabel grup keamanan deafult cluster Fargate dengan memasukkan perintah berikut, ganti *my-default-security-group*dengan nilai yang Anda temukan dari perintah sebelumnya.

```
ECS_CLUSTER_SECURITY_GROUP=my-default-security-group
```

Instal beban kerja Redis sampel

Untuk melakukan instalasi beban kerja Redis sampel yang membuka metrik Prometheus

1. Unduh AWS CloudFormation template Redis dengan memasukkan perintah berikut.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/
cwagent-prometheus/sample_traffic/redis/redis-traffic-sample.yaml
```

2. Tetapkan nama-nama peran IAM yang akan dibuat untuk Redis dengan memasukkan perintah-perintah berikut.

```
REDIS_ECS_TASK_ROLE_NAME=redis-prometheus-demo-ecs-task-role-name
REDIS_ECS_EXECUTION_ROLE_NAME=redis-prometheus-demo-ecs-execution-role-name
```

3. Instal beban kerja Redis sampel dengan memasukkan perintah berikut.

```
aws cloudformation create-stack --stack-name Redis-Prometheus-Demo-ECS-
$ECS_CLUSTER_NAME-fargate-awsvpc \
    --template-body file://redis-traffic-sample.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=SecurityGroupID,ParameterValue=
$ECS_CLUSTER_SECURITY_GROUP \
                 ParameterKey=SubnetID,ParameterValue=$ECS_CLUSTER_SUBNET_1 \
                 ParameterKey=TaskRoleName,ParameterValue=$REDIS_ECS_TASK_ROLE_NAME
 \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$REDIS_ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_DEFAULT_REGION
```

AWS CloudFormation Tumpukan menciptakan empat sumber daya:

- Satu peran tugas ECS
- Satu peran eksekusi tugas ECS
- Satu penetapan tugas Redis
- Satu layanan Redis

Dalam penetapan tugas Redis tersebut, ada dua kontainer yang ditentukan:

- Kontainer utama menjalankan sebuah aplikasi Redis sederhana dan membuka port 6379 agar bisa diakses.
- Kontainer lainnya yang menjalankan proses pengekspor Redis untuk membuka metrik Prometheus di port 9121. Ini adalah wadah yang akan ditemukan dan dikikis oleh agen. CloudWatch Label docker berikut didefinisikan sehingga CloudWatch agen dapat menemukan wadah ini berdasarkan itu.

```
ECS_PROMETHEUS_EXPORTER_PORT: 9121
```

Konfigurasikan CloudWatch agen untuk mengikis metrik Redis Prometheus

Untuk mengonfigurasi CloudWatch agen untuk mengikis metrik Redis Prometheus

1. Unduh `cwagent-ecs-prometheus-metric-for-awsvpc.yaml` versi terbaru dengan memasukkan perintah berikut.

   ```
   curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-awsvpc.yaml
   ```

2. Buka file dengan editor teks, dan temukan konfigurasi CloudWatch agen lengkap di belakang `value` kunci di `resource:CWAgentConfigSSMParameter` bagian.

   Kemudian, di bagian `ecs_service_discovery` yang ditampilkan di sini, penemuan layanan berbasis `docker_label` diaktifkan dengan pengaturan bawaan yang didasarkan pada `ECS_PROMETHEUS_EXPORTER_PORT`, yang sesuai dengan label docker yang kami tentukan

dalam penetapan tugas Redis ECS. Jadi, kita tidak perlu melakukan perubahan apa pun dalam bagian ini:

```
ecs_service_discovery": {
  "sd_frequency": "1m",
  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
*  "docker_label": {
  },*
  ...
```

Untuk bagian `metric_declaration`, pengaturan bawaan tidak mengizinkan metrik Redis. Tambahkan bagian berikut untuk mengizinkan metrik Redis. Pastikan untuk mengikuti pola indentasi yang ada saat itu.

```
{
  "source_labels": ["container_name"],
  "label_matcher": "^redis-exporter-.*$",
  "dimensions": [["ClusterName","TaskDefinitionFamily"]],
  "metric_selectors": [
    "^redis_net_(in|out)put_bytes_total$",
    "^redis_(expired|evicted)_keys_total$",
    "^redis_keyspace_(hits|misses)_total$",
    "^redis_memory_used_bytes$",
    "^redis_connected_clients$"
  ]
},
{
  "source_labels": ["container_name"],
  "label_matcher": "^redis-exporter-.*$",
  "dimensions": [["ClusterName","TaskDefinitionFamily","cmd"]],
  "metric_selectors": [
    "^redis_commands_total$"
  ]
},
{
  "source_labels": ["container_name"],
  "label_matcher": "^redis-exporter-.*$",
  "dimensions": [["ClusterName","TaskDefinitionFamily","db"]],
  "metric_selectors": [
    "^redis_db_keys$"
  ]
},
```

3. Jika Anda sudah memiliki CloudWatch agen yang digunakan di cluster Amazon ECS oleh AWS CloudFormation, Anda dapat membuat set perubahan dengan memasukkan perintah berikut.

```
ECS_LAUNCH_TYPE=FARGATE
CREATE_IAM_ROLES=True
ECS_CLUSTER_SUBNET=$ECS_CLUSTER_SUBNET_1
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-change-set --stack-name CWAgent-Prometheus-ECS-
$ECS_CLUSTER_NAME-$ECS_LAUNCH_TYPE-awsvpc \
    --template-body file://cwagent-ecs-prometheus-metric-for-awsvpc.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSLaunchType,ParameterValue=$ECS_LAUNCH_TYPE \
                 ParameterKey=SecurityGroupID,ParameterValue=
$ECS_CLUSTER_SECURITY_GROUP \
                 ParameterKey=SubnetID,ParameterValue=$ECS_CLUSTER_SUBNET \
                 ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${AWS_DEFAULT_REGION} \
    --change-set-name redis-scraping-support
```

4. Buka AWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation.

5. Meninjau changeset `redis-scraping-support` yang baru saja Anda buat. Anda seharusnya melihat satu perubahan diterapkan ke sumber daya `CWAgentConfigSSMParameter`. Jalankan changeset dan restart tugas CloudWatch agen dengan memasukkan perintah berikut.

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 0 \
--service cwagent-prometheus-replica-service-$ECS_LAUNCH_TYPE-awsvpc \
--region ${AWS_DEFAULT_REGION}
```

6. Tunggu sekitar 10 detik, kemudian masukkan perintah berikut.

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 1 \
--service cwagent-prometheus-replica-service-$ECS_LAUNCH_TYPE-awsvpc \
--region ${AWS_DEFAULT_REGION}
```

7. Jika Anda menginstal CloudWatch agen dengan pengumpulan metrik Prometheus untuk cluster untuk pertama kalinya, masukkan perintah berikut:

```
ECS_LAUNCH_TYPE=FARGATE
CREATE_IAM_ROLES=True
ECS_CLUSTER_SUBNET=$ECS_CLUSTER_SUBNET_1
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
$ECS_CLUSTER_NAME-$ECS_LAUNCH_TYPE-awsvpc \
    --template-body file://cwagent-ecs-prometheus-metric-for-awsvpc.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSLaunchType,ParameterValue=$ECS_LAUNCH_TYPE \
                 ParameterKey=SecurityGroupID,ParameterValue=
$ECS_CLUSTER_SECURITY_GROUP \
                 ParameterKey=SubnetID,ParameterValue=$ECS_CLUSTER_SUBNET \
                 ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${AWS_DEFAULT_REGION}
```

Menampilkan metrik Redis

Tutorial ini mengirimkan metrik berikut ke namespace ContainerInsightsECS//Prometheus di. CloudWatch Anda dapat menggunakan CloudWatch konsol untuk melihat metrik di namespace tersebut.

Nama Metrik	Dimensi	
`redis_net_input_bytes_total`	ClusterName, `TaskDefinitionFamily`	
`redis_net_output_bytes_total`	ClusterName, `TaskDefinitionFamily`	

Nama Metrik	Dimensi	
redis_expired_keys_total	ClusterName, TaskDefinitionFamily	
redis_evicted_keys_total	ClusterName, TaskDefinitionFamily	
redis_keyspace_hits_total	ClusterName, TaskDefinitionFamily	
redis_keyspace_misses_total	ClusterName, TaskDefinitionFamily	
redis_memory_used_bytes	ClusterName, TaskDefinitionFamily	
redis_connected_clients	ClusterName, TaskDefinitionFamily	
redis_commands_total	ClusterName , TaskDefinitionFamily , cmd	
redis_db_keys	ClusterName , TaskDefinitionFamily , db	

Note

Nilai dari dimensi cmd bisa berupa: append, client, command, config, dbsize, flushall, get, incr, info, latency, atau slowlog.
Nilai dari dimensi db bisa db0 hingga db15.

Anda juga dapat membuat CloudWatch dasbor untuk metrik Redis Prometheus Anda.

Cara membuat sebuah dasbor untuk metrik-metrik Redis Prometheus

1. Buat variabel lingkungan, yang menggantikan nilai di bawah ini untuk menyesuaikan dengan deployment Anda.

```
DASHBOARD_NAME=your_cw_dashboard_name
ECS_TASK_DEF_FAMILY=redis-prometheus-demo-$ECS_CLUSTER_NAME-fargate-awsvpc
```

2. Masukkan perintah berikut untuk membuat dasbor tersebut.

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-
prometheus/sample_cloudwatch_dashboards/redis/cw_dashboard_redis.json \
| sed "s/{{YOUR_AWS_REGION}}/${REGION_NAME}/g" \
| sed "s/{{YOUR_CLUSTER_NAME}}/${CLUSTER_NAME}/g" \
| sed "s/{{YOUR_NAMESPACE}}/${NAMESPACE}/g" \
```

Menyiapkan dan mengonfigurasikan koleksi metrik-metrik Prometheus di klaster Amazon ECS dan Kubernetes

Untuk mengumpulkan metrik Prometheus dari cluster yang menjalankan Amazon EKS atau Kubernetes, Anda dapat menggunakan agen sebagai kolektor atau menggunakan CloudWatch Distro untuk kolektor. AWS OpenTelemetry Untuk informasi tentang menggunakan AWS Distro untuk OpenTelemetry kolektor, lihat https://aws-otel.github.io/docs/getting-started/container-insights/eks-prometheus.

Bagian berikut menjelaskan cara mengumpulkan metrik Prometheus menggunakan agen. CloudWatch Mereka menjelaskan cara menginstal CloudWatch agen dengan pemantauan Prometheus pada cluster yang menjalankan Amazon EKS atau Kubernetes, dan cara mengonfigurasi agen untuk mengikis target tambahan. Bagian-bagian ini juga akan menyediakan tutorial opsional untuk menyiapkan beban kerja sampel untuk digunakan dalam pengujian dengan pemantauan Prometheus.

Topik

- Instal CloudWatch agen dengan koleksi metrik Prometheus di klaster Amazon EKS dan Kubernetes

Instal CloudWatch agen dengan koleksi metrik Prometheus di klaster Amazon EKS dan Kubernetes

Bagian ini menjelaskan cara mengatur CloudWatch agen dengan pemantauan Prometheus di cluster yang menjalankan Amazon EKS atau Kubernetes. Setelah melakukan hal tersebut, agen secara otomatis melakukan scraping terhadap dan mengimpor metrik-metrik untuk beban kerja berikut yang berjalan di klaster tersebut.

- AWS App Mesh
- NGINX
- Memcached
- Java/JMX
- HAProxy
- Fluent Bit

Anda juga dapat mengonfigurasi agen tersebut untuk melakukan scraping atas dan mengimpor beban kerja dan sumber Prometheus tambahan.

Sebelum mengikuti langkah-langkah ini untuk menginstal CloudWatch agen untuk koleksi metrik Prometheus, Anda harus memiliki klaster yang berjalan di Amazon EKS atau cluster Kubernetes yang berjalan pada instance Amazon EC2.

Persyaratan grup keamanan VPC

Aturan masuknya kelompok keamanan untuk beban kerja Prometheus harus membuka port CloudWatch Prometheus ke agen untuk mengikis metrik Prometheus oleh IP pribadi.

Aturan keluar dari grup keamanan untuk CloudWatch agen harus memungkinkan agen untuk terhubung ke port CloudWatch beban kerja Prometheus dengan IP pribadi.

Topik

Instal CloudWatch agen dengan koleksi metrik Prometheus di klaster Amazon EKS dan Kubernetes

Bagian ini menjelaskan cara mengatur CloudWatch agen dengan pemantauan Prometheus di cluster yang menjalankan Amazon EKS atau Kubernetes. Setelah melakukan hal tersebut, agen secara otomatis melakukan scraping terhadap dan mengimpor metrik-metrik untuk beban kerja berikut yang berjalan di klaster tersebut.

- AWS App Mesh
- NGINX
- Memcached
- Java/JMX
- HAProxy
- Fluent Bit

Anda juga dapat mengonfigurasi agen tersebut untuk melakukan scraping atas dan mengimpor beban kerja dan sumber Prometheus tambahan.

Sebelum mengikuti langkah-langkah ini untuk menginstal CloudWatch agen untuk koleksi metrik Prometheus, Anda harus memiliki klaster yang berjalan di Amazon EKS atau cluster Kubernetes yang berjalan pada instance Amazon EC2.

Persyaratan grup keamanan VPC

Aturan masuknya kelompok keamanan untuk beban kerja Prometheus harus membuka port CloudWatch Prometheus ke agen untuk mengikis metrik Prometheus oleh IP pribadi.

Aturan keluar dari grup keamanan untuk CloudWatch agen harus memungkinkan agen untuk terhubung ke port CloudWatch beban kerja Prometheus dengan IP pribadi.

Topik

Menyiapkan peran IAM

Langkah pertama adalah menyiapkan peran IAM yang diperlukan di klaster tersebut. Ada dua metode:

- Menyiapkan peran IAM untuk sebuah akun layanan, juga dikenal sebagai sebuah peran layanan. Metode ini bisa digunakan untuk tipe peluncuran EC2 dan tipe peluncuran Fargate.
- Menambahkan kebijakan IAM ke peran IAM yang digunakan untuk klaster. Ini hanya bisa dilakukan untuk tipe peluncuran EC2.

Menyiapkan sebuah peran layanan (tipe peluncuran EC2 dan tipe peluncuran Fargate)

Untuk menyiapkan sebuah peran layanan, masukkan perintah berikut. Ganti *MyCluster*dengan nama cluster.

```
eksctl create iamserviceaccount \
 --name cwagent-prometheus \
--namespace amazon-cloudwatch \
 --cluster MyCluster \
--attach-policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy \
--approve \
--override-existing-serviceaccounts
```

Menambahkan sebuah kebijakan ke peran IAM yang dimiliki klaster (khusus tipe peluncuran EC2)

Untuk menyiapkan kebijakan IAM tersebut dalam sebuah klaster untuk dukungan Prometheus

1. Buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.
2. Pada panel navigasi, silakan pilih Instances (Instans).
3. Anda harus menemukan awalan nama peran IAM untuk klaster. Untuk melakukan hal itu, pilih kotak centang yang ada di samping nama instans yang ada di klaster, kemudian pilih Tindakan, Pengaturan Instans, Lampirkan/Ganti Peran IAM. Kemudian salin awalan peran IAM, seperti `eksctl-dev303-workshop-nodegroup`.
4. Buka konsol IAM di https://console.aws.amazon.com/iam/.
5. Pada panel navigasi, silakan pilih Peran.
6. Gunakan kotak pencarian untuk menemukan awalan yang Anda salin sebelumnya dalam prosedur ini, dan pilih peran tersebut.
7. Pilih Lampirkan kebijakan.
8. Gunakan kotak pencarian untuk menemukan CloudWatchAgentServerPolicy. Pilih kotak centang di samping CloudWatchAgentServerPolicy, dan pilih Lampirkan kebijakan.

Menginstal CloudWatch agen untuk mengumpulkan metrik Prometheus

Anda harus menginstal CloudWatch agen di cluster untuk mengumpulkan metrik. Cara melakukan instalasi agen yang berbeda untuk klaster Amazon EKS dan klaster Kubernetes.

Hapus versi CloudWatch agen sebelumnya dengan dukungan Prometheus

Jika Anda telah menginstal versi CloudWatch agen dengan dukungan Prometheus di cluster Anda, Anda harus menghapus versi itu dengan memasukkan perintah berikut. Hal perlu dilakukan hanya untuk versi agen sebelumnya yang memiliki dukungan Prometheus. Anda tidak perlu menghapus CloudWatch agen yang mengaktifkan Wawasan Kontainer tanpa dukungan Prometheus.

```
kubectl delete deployment cwagent-prometheus -n amazon-cloudwatch
```

Menginstal CloudWatch agen di kluster Amazon EKS dengan tipe peluncuran EC2

Untuk menginstal CloudWatch agen dengan dukungan Prometheus di kluster Amazon EKS, ikuti langkah-langkah ini.

Untuk menginstal CloudWatch agen dengan dukungan Prometheus di cluster Amazon EKS

1. Masukkan perintah berikut untuk memeriksa apakah namespace `amazon-cloudwatch` telah dibuat:

   ```
   kubectl get namespace
   ```

2. Jika `amazon-cloudwatch` tidak ditampilkan di hasil, buat ia dengan memasukkan perintah berikut:

   ```
   kubectl create namespace amazon-cloudwatch
   ```

3. Untuk menyebarkan agen dengan konfigurasi default dan mengirimkannya data ke AWS Wilayah tempat ia diinstal, masukkan perintah berikut:

   ```
   kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
   container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/
   service/cwagent-prometheus/prometheus-eks.yaml
   ```

   Agar agen tersebut mengirimkan data ke Wilayah yang berbeda, ikuti langkah-langkah berikut:

   a. Unduh file YAML untuk agen tersebut dengan memasukkan perintah berikut:

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/
service/cwagent-prometheus/prometheus-eks.yaml
```

b. Buka file dengan editor teks, dan cari blok `cwagentconfig.json` dari file tersebut.

c. Tambahkan baris-baris yang disorot, dengan menyebutkan Wilayah yang Anda kehendaki:

```
cwagentconfig.json: |
    {
      "agent": {
        "region": "us-east-2"
      },
      "logs": { ...
```

d. Simpan file tersebut dan terapkan agen dengan menggunakan file yang telah diperbarui.

```
kubectl apply -f prometheus-eks.yaml
```

Menginstal CloudWatch agen di cluster Amazon EKS dengan tipe peluncuran Fargate

Untuk menginstal CloudWatch agen dengan dukungan Prometheus di cluster Amazon EKS dengan jenis peluncuran Fargate, ikuti langkah-langkah ini.

Untuk menginstal CloudWatch agen dengan dukungan Prometheus di cluster Amazon EKS dengan tipe peluncuran Fargate

1. Masukkan perintah berikut untuk membuat profil Fargate untuk CloudWatch agen sehingga dapat berjalan di dalam cluster. Ganti *MyCluster*dengan nama cluster.

```
eksctl create fargateprofile --cluster MyCluster \
--name amazon-cloudwatch \
--namespace amazon-cloudwatch
```

2. Untuk menginstal CloudWatch agen, masukkan perintah berikut. Ganti *MyCluster*dengan nama cluster. Nama ini digunakan dalam nama grup log yang menyimpan peristiwa log yang dikumpulkan oleh agen, dan juga digunakan sebagai dimensi untuk metrik-metrik yang dikumpulkan oleh agen itu.

Ganti *wilayah* dengan nama Wilayah di mana metrik akan dikirimkan. Sebagai contoh, `us-west-1`.

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-eks-fargate.yaml |
sed "s/{{cluster_name}}/MyCluster/;s/{{region_name}}/region/" |
kubectl apply -f -
```

Menginstal CloudWatch agen pada cluster Kubernetes

Untuk menginstal CloudWatch agen dengan dukungan Prometheus pada cluster yang menjalankan Kubernetes, masukkan perintah berikut:

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-k8s.yaml |
sed "s/{{cluster_name}}/MyCluster/;s/{{region_name}}/region/" |
kubectl apply -f -
```

Ganti *MyCluster*dengan nama cluster. Nama ini digunakan dalam nama grup log yang menyimpan peristiwa log yang dikumpulkan oleh agen, dan juga digunakan sebagai dimensi untuk metrik-metrik yang dikumpulkan oleh agen itu.

Ganti *wilayah* dengan nama AWS Wilayah tempat Anda ingin metrik dikirim. Misalnya, **us-west-1**.

Verifikasi bahwa agen sedang berjalan

Pada keduanya, baik di klaster Amazon EKS maupun Kubernetes, Anda dapat memasukkan perintah berikut untuk mengonfirmasi bahwa agen sedang berjalan.

```
kubectl get pod -l "app=cwagent-prometheus" -n amazon-cloudwatch
```

Jika hasilnya menyertakan satu pod CloudWatch agen dalam `Running` status, agen menjalankan dan mengumpulkan metrik Prometheus. Secara default, CloudWatch agen mengumpulkan metrik untuk App Mesh, NGINX, Memcached, Java/JMX, dan HAProxy setiap menit. Untuk informasi selengkapnya tentang metrik-metrik ini, silakan lihat Metrik Prometheus dikumpulkan oleh agen CloudWatch . Untuk petunjuk tentang cara melihat metrik Prometheus Anda di, lihat CloudWatch Menampilkan metrik-metrik Prometheus Anda

Anda juga dapat mengonfigurasi CloudWatch agen untuk mengumpulkan metrik dari eksportir Prometheus lainnya. Untuk informasi selengkapnya, lihat Melakukan scraping atas sumber Prometheus tambahan dan mengimpor metrik-metrik tersebut.

Melakukan scraping atas sumber Prometheus tambahan dan mengimpor metrik-metrik tersebut

CloudWatch Agen dengan pemantauan Prometheus membutuhkan dua konfigurasi untuk mengikis metrik Prometheus. Salah satunya adalah konfigurasi standar Prometheus sebagaimana yang didokumentasikan dalam <scrape_config> dalam dokumentasi Prometheus. Yang lainnya adalah untuk konfigurasi CloudWatch agen.

Untuk klaster Amazon EKS, konfigurasi ditentukan dalam `prometheus-eks.yaml` (untuk tipe peluncuran EC2) atau `prometheus-eks-fargate.yaml` (untuk tipe peluncuran Fargate) sebagai dua peta konfigurasi:

- Bagian `name: prometheus-config` berisi pengaturan untuk scraping Prometheus.
- `name: prometheus-cwagentconfig`Bagian ini berisi konfigurasi untuk CloudWatch agen. Anda dapat menggunakan bagian ini untuk mengonfigurasi bagaimana metrik Prometheus dikumpulkan oleh. CloudWatch Misalnya, Anda menentukan metrik mana yang akan diimpor CloudWatch, dan menentukan dimensinya.

Untuk klaster Kubernet yang berjalan di instans Amazon EC2, konfigurasinya ditentukan di file YAML `prometheus-k8s.yaml` sebagai dua peta konfigurasi:

- Bagian `name: prometheus-config` berisi pengaturan untuk scraping Prometheus.
- `name: prometheus-cwagentconfig`Bagian ini berisi konfigurasi untuk CloudWatch agen.

Untuk mengikis sumber metrik Prometheus tambahan dan mengimpor metrik tersebut, Anda memodifikasi CloudWatch konfigurasi scrape Prometheus dan konfigurasi agen, lalu menerapkan kembali agen dengan konfigurasi yang diperbarui. CloudWatch

Persyaratan grup keamanan VPC

Aturan masuknya kelompok keamanan untuk beban kerja Prometheus harus membuka port CloudWatch Prometheus ke agen untuk mengikis metrik Prometheus oleh IP pribadi.

Aturan keluar dari grup keamanan untuk CloudWatch agen harus memungkinkan agen untuk terhubung ke port CloudWatch beban kerja Prometheus dengan IP pribadi.

Konfigurasi scraping Prometheus

CloudWatch <scrape_config>Agen mendukung konfigurasi scrape Prometheus standar seperti yang didokumentasikan dalam dokumentasi Prometheus.https://prometheus.io/docs/prometheus/latest/configuration/configuration/#scrape_config Anda dapat mengedit bagian ini untuk memperbarui konfigurasi yang sudah ada dalam file ini, dan menambahkan target-target scraping Prometheus tambahan. Secara bawaan, file konfigurasi sampel berisi baris-baris konfigurasi global berikut ini:

```
global:
  scrape_interval: 1m
  scrape_timeout: 10s
```

- scrape_interval— Mendefinisikan seberapa sering scraping akan dilakukan ke target.
- scrape_timeout— Mendefinisikan berapa lama waktu tunggu sebelum permintaan scraping habis.

Anda juga dapat menentukan nilai-nilai yang berbeda untuk pengaturan ini di level pekerjaan, untuk mengganti konfigurasi global.

Pekerjaan scraping Prometheus

File CloudWatch agen YAMAL sudah memiliki beberapa pekerjaan pengikisan default yang dikonfigurasi. Sebagai contoh, di `prometheus-eks.yaml`, pekerjaan scraping bawaan dikonfigurasi di baris `job_name` pada bagian `scrape_configs`. Dalam file ini, bagian `kubernetes-pod-jmx` bawaan berikut melakukan scraping terhadap metrik-metrik pengekspor JMX.

```
   - job_name: 'kubernetes-pod-jmx'
     sample_limit: 10000
     metrics_path: /metrics
     kubernetes_sd_configs:
     - role: pod
     relabel_configs:
     - source_labels: [__address__]
       action: keep
       regex: '.*:9404$'
     - action: labelmap
       regex: __meta_kubernetes_pod_label_(.+)
     - action: replace
       source_labels:
       - __meta_kubernetes_namespace
       target_label: Namespace
```

```
      - source_labels: [__meta_kubernetes_pod_name]
        action: replace
        target_label: pod_name
      - action: replace
        source_labels:
        - __meta_kubernetes_pod_container_name
        target_label: container_name
      - action: replace
        source_labels:
        - __meta_kubernetes_pod_controller_name
        target_label: pod_controller_name
      - action: replace
        source_labels:
        - __meta_kubernetes_pod_controller_kind
        target_label: pod_controller_kind
      - action: replace
        source_labels:
        - __meta_kubernetes_pod_phase
        target_label: pod_phase
```

Masing-masing target default ini dikikis, dan metrik dikirim ke CloudWatch peristiwa log menggunakan format metrik yang disematkan. Untuk informasi selengkapnya, lihat Menyematkan metrik dalam log.

Peristiwa log dari kluster Amazon EKS dan Kubernetes disimpan di grup log ***/aws/containerinsights/ cluster_name /prometheus*** *di Logs.* CloudWatch Peristiwa log dari klaster Amazon ECS disimpan di grup log /aws/ecs/containerinsights/***cluster_name***/prometheus.

Setiap pekerjaan scraping dimuat dalam log stream yang berbeda di grup log ini. Sebagai contoh, pekerjaan scraping Prometheus `kubernetes-pod-appmesh-envoy` ditetapkan untuk App Mesh. ***Semua metrik App Mesh Prometheus dari kluster Amazon EKS dan Kubernetes dikirim ke aliran log bernama /aws/containerinsights/ cluster_name >prometheus//.*** kubernetes-pod-appmesh-envoy

Untuk menambahkan sebuah target scraping baru, Anda harus menambahkan sebuah bagian `job_name` baru pada bagian `scrape_configs` file YAML, dan kemudian mulai ulang agen tersebut. Untuk contoh proses ini, silakan lihat Tutorial untuk menambahkan sebuah target scraping Prometheus baru: metrik Server API Prometheus.

CloudWatch konfigurasi agen untuk Prometheus

File konfigurasi CloudWatch agen memiliki `prometheus` bagian di bawah `metrics_collected` untuk konfigurasi pengikisan Prometheus. File konfigurasi tersebut mencakup opsi-opsi konfigurasi berikut:

- cluster_name— menentukan nama klaster yang akan ditambahkan sebagai label pada peristiwa log. Bidang ini bersifat opsional. Jika Anda menghilangkannya, maka agen tersebut akan dapat mendeteksi nama klaster Amazon EKS dan Kubernetes.
- log_group_name— menentukan nama grup log untuk metrik-metrik Prometheus yang di-scraping. Bidang ini bersifat opsional. Jika Anda menghilangkannya, CloudWatch gunakan / aws/containerinsights/ ***`cluster_name /prometheus untuk log dari Amazon EKS dan Kubernetes cluster`***.
- prometheus_config_path— menentukan jalur file konfigurasi scraping Prometheus. Jika nilai bidang ini dimulai dengan `env:`, maka konten file konfigurasi Prometheus scraping akan diambil dari variabel lingkungan kontainer. Jangan ubah bidang ini.
- ecs_service_discovery— adalah bagian untuk menentukan konfigurasi untuk penemuan layanan Amazon ECS Prometheus. Untuk informasi selengkapnya, lihat Panduan mendetail untuk penemuan otomatis di klaster-klaster Amazon ECS.

  Bagian `ecs_service_discovery` dapat berisi bidang-bidang berikut:

  - `sd_frequency` adalah frekuensi untuk menemukan pengekspor Prometheus. Tentukan sebuah angka dan sebuah akhiran unit. Sebagai contoh, `1m` satu kali per menit atau `30s` satu kali per 30 detik. Akhiran unit yang valid adalah `ns`, `us`, `ms`, `s`, `m`, dan `h`.

    Bidang ini bersifat opsional. Bawaannya adalah 60 detik (1 menit).
  - `sd_target_cluster` adalah nama klaster Amazon ECS target untuk penemuan otomatis. Bidang ini bersifat opsional. Defaultnya adalah nama cluster Amazon ECS tempat CloudWatch agen diinstal.
  - `sd_cluster_region` adalah Wilayah klaster Amazon ECS target. Bidang ini bersifat opsional. Defaultnya adalah Wilayah cluster Amazon ECS tempat CloudWatch agen diinstal.
  - `sd_result_file` adalah jalur file YAML untuk hasil-hasil target Prometheus. Konfigurasi scraping Prometheus akan merujuk pada file ini.
  - `docker_label` adalah sebuah bagian opsional yang dapat Anda gunakan untuk menentukan konfigurasi untuk penemuan layanan berbasis label docker. Jika Anda menghilangkan bagian

ini, maka penemuan berbasis label docker tidak akan digunakan. Bagian ini dapat berisi bidang-bidang berikut:

- `sd_port_label` adalah nama label docker dari kontainer yang menentukan port kontainer untuk metrik-metrik Prometheus. Nilai bawaannya adalah `ECS_PROMETHEUS_EXPORTER_PORT`. Jika wadah tidak memiliki label docker ini, CloudWatch agen akan melewatkannya.
- `sd_metrics_path_label` adalah nama label docker dari kontainer yang menentukan jalur metrik-metrik Prometheus. Nilai bawaannya adalah `ECS_PROMETHEUS_METRICS_PATH`. Jika kontainer tidak memiliki label docker ini, maka agen akan mengasumsikan jalur default `/metrics`.
- `sd_job_name_label` adalah nama label docker dari kontainer yang menentukan nama pekerjaan scraping Prometheus. Nilai bawaannya adalah `job`. Jika wadah tidak memiliki label docker ini, CloudWatch agen menggunakan nama pekerjaan dalam konfigurasi scrape Prometheus.

- `task_definition_list` adalah sebuah bagian opsional yang dapat Anda gunakan untuk menentukan konfigurasi penemuan layanan berbasis penetapan tugas. Jika Anda menghilangkan bagian ini, maka penemuan berbasis penetapan tugas tidak akan digunakan. Bagian ini dapat berisi bidang-bidang berikut:
  - `sd_task_definition_arn_pattern` adalah pola yang akan digunakan untuk menentukan penetapan tugas Amazon ECS yang harus ditemukan. Ini adalah ekspresi biasa.
  - `sd_metrics_ports` mencantumkan containerPort untuk metrik-metrik Prometheus. Pisahkan containerPorts dengan titik koma.
  - `sd_container_name_pattern` menentukan nama-nama kontainer tugas Amazon ECS. Ini adalah ekspresi biasa.
  - `sd_metrics_path` menentukan jalur metrik Prometheus. Jika Anda menghilangkannya, maka agen tersebut akan mengasumsikan jalur bawaan `/metrics`
  - `sd_job_name` menyebutkan nama pekerjaan scraping Prometheus. Jika Anda menghilangkan bidang ini, CloudWatch agen menggunakan nama pekerjaan dalam konfigurasi scrape Prometheus.

- metric_declaration— adalah bagian-bagian yang menentukan larik log dengan format metrik tersemat yang akan dihasilkan. Ada `metric_declaration` bagian untuk setiap sumber Prometheus yang diimpor agen secara default CloudWatch . Masing-masing bagian ini mencakup bidang-bidang berikut:

- `label_matcher` adalah ekspresi reguler yang memeriksa nilai dari label-label yang tercantum dalam `source_labels`. Metrik yang cocok diaktifkan untuk dimasukkan dalam format metrik tertanam yang dikirim ke CloudWatch.

  Jika Anda memiliki beberapa label yang ditentukan dalam `source_labels`, maka kami menyarankan Anda untuk tidak menggunakan karakter ^ atau $ dalam ekspresi reguler untuk `label_matcher`.
- `source_labels` menentukan nilai dari label-label yang diperiksa oleh baris `label_matcher`.
- `label_separator` menentukan pemisah yang akan digunakan dalam baris `label_matcher` jika ada beberapa `source_labels` yang ditetapkan. Bawaannya adalah `;`. Anda dapat melihat nilai-nilai bawaan ini digunakan di `label_matcher` dalam contoh berikut.
- `metric_selectors`adalah ekspresi reguler yang menentukan metrik yang akan dikumpulkan dan dikirim ke. CloudWatch
- `dimensions`adalah daftar label yang akan digunakan sebagai CloudWatch dimensi untuk setiap metrik yang dipilih.

Lihat contoh `metric_declaration` berikut ini.

```
"metric_declaration": [
  {
     "source_labels":[ "Service", "Namespace"],
     "label_matcher":"(.*node-exporter.*|.*kube-dns.*);kube-system",
     "dimensions":[
        ["Service", "Namespace"]
     ],
     "metric_selectors":[
        "^coredns_dns_request_type_count_total$"
     ]
  }
]
```

Contoh ini mengonfigurasikan sebuah bagian format metrik tersemat yang akan dikirim sebagai sebuah peristiwa log jika kondisi-kondisi berikut dipenuhi:

- Nilai dari `Service` berisi `node-exporter` atau `kube-dns`.
- Nilai dari `Namespace` adalah `kube-system`.

- Metrik Prometheus `coredns_dns_request_type_count_total` memuat label `Service` dan `Namespace`.

Peristiwa log yang dikirim mencakup bagian yang disorot berikut ini:

```
{
   "CloudWatchMetrics":[
      {
         "Metrics":[
            {
               "Name":"coredns_dns_request_type_count_total"
            }
         ],
         "Dimensions":[
            [
               "Namespace",
               "Service"
            ]
         ],
         "Namespace":"ContainerInsights/Prometheus"
      }
   ],
   "Namespace":"kube-system",
   "Service":"kube-dns",
   "coredns_dns_request_type_count_total":2562,
   "eks_amazonaws_com_component":"kube-dns",
   "instance":"192.168.61.254:9153",
   "job":"kubernetes-service-endpoints",
   ...
}
```

Tutorial untuk menambahkan sebuah target scraping Prometheus baru: metrik Server API Prometheus

Server API Kubernetes membuka metrik-metrik Prometheus pada titik akhir secara bawaan. Contoh resmi untuk konfigurasi scraping Server API Kubernetes tersedia di Github.

Tutorial berikut menunjukkan cara melakukan langkah-langkah berikut untuk mulai mengimpor metrik Kubernetes API Server ke: CloudWatch

- Menambahkan konfigurasi scraping Prometheus untuk Kubernetes API Server ke file YAMM agen. CloudWatch

- Mengonfigurasi definisi metrik format metrik yang disematkan dalam file CloudWatch YAMAL agen.
- (Opsional) Membuat CloudWatch dasbor untuk metrik Kubernetes API Server.

> **Note**
>
> Server API Kubernetes membuka metrik-metrik pengukuran, penghitung, histogram, dan ringkasan. Dalam rilis dukungan metrik Prometheus ini, hanya mengimpor metrik dengan jenis CloudWatch pengukur, penghitung, dan ringkasan.

Untuk mulai mengumpulkan metrik Prometheus Server API Kubernetes di CloudWatch

1. Unduh versi terbaru dari file `prometheus-eks.yaml`, `prometheus-eks-fargate.yaml`, atau `prometheus-k8s.yaml` dengan memasukkan salah satu dari perintah-perintah berikut.

   Untuk sebuah klaster Amazon EKS dengan tipe peluncuran EC2, masukkan perintah berikut:

   ```
   curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-eks.yaml
   ```

   Untuk sebuah klaster Amazon EKS dengan tipe peluncuran Fargate, masukkan perintah berikut:

   ```
   curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-eks-fargate.yaml
   ```

   Untuk sebuah klaster Kubernetes yang berjalan di sebuah instans Amazon EC2, masukkan perintah berikut:

   ```
   curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-k8s.yaml
   ```

2. Buka file dengan editor teks, temukan bagian `prometheus-config` dan tambahkan bagian berikut di dalam bagian tersebut. Kemudian simpan perubahannya:

   ```
       # Scrape config for API servers
       - job_name: 'kubernetes-apiservers'
   ```

```
      kubernetes_sd_configs:
        - role: endpoints
          namespaces:
            names:
              - default
      scheme: https
      tls_config:
        ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        insecure_skip_verify: true
      bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
      relabel_configs:
      - source_labels: [__meta_kubernetes_service_name,
 __meta_kubernetes_endpoint_port_name]
        action: keep
        regex: kubernetes;https
      - action: replace
        source_labels:
        - __meta_kubernetes_namespace
        target_label: Namespace
      - action: replace
        source_labels:
        - __meta_kubernetes_service_name
        target_label: Service
```

3. Meskipun file YAML masih terbuka di editor teks, temukan bagian `cwagentconfig.json`. Tambahkan subbagian berikut dan simpan perubahannya. Bagian ini menempatkan metrik server API ke daftar CloudWatch izin agen. Tiga jenis metrik Server API ditambahkan ke daftar yang diizinkan:

   - penghitungan objek etcd
   - Metrik pengendali registrasi Server API
   - Metrik permintaan Server API

```
{"source_labels": ["job", "resource"],
  "label_matcher": "^kubernetes-apiservers;(services|daemonsets.apps|
deployments.apps|configmaps|endpoints|secrets|serviceaccounts|replicasets.apps)",
  "dimensions": [["ClusterName","Service","resource"]],
  "metric_selectors": [
  "^etcd_object_counts$"
  ]
},
```

```
{"source_labels": ["job", "name"],
   "label_matcher": "^kubernetes-apiservers;APIServiceRegistrationController$",
   "dimensions": [["ClusterName","Service","name"]],
   "metric_selectors": [
   "^workqueue_depth$",
   "^workqueue_adds_total$",
   "^workqueue_retries_total$"
  ]
},
{"source_labels": ["job","code"],
  "label_matcher": "^kubernetes-apiservers;2[0-9]{2}$",
  "dimensions": [["ClusterName","Service","code"]],
  "metric_selectors": [
   "^apiserver_request_total$"
  ]
},
{"source_labels": ["job"],
  "label_matcher": "^kubernetes-apiservers",
  "dimensions": [["ClusterName","Service"]],
  "metric_selectors": [
  "^apiserver_request_total$"
  ]
},
```

4. Jika Anda sudah memiliki CloudWatch agen dengan dukungan Prometheus yang diterapkan di cluster, Anda harus menghapusnya dengan memasukkan perintah berikut:

```
kubectl delete deployment cwagent-prometheus -n amazon-cloudwatch
```

5. Terapkan CloudWatch agen dengan konfigurasi Anda yang diperbarui dengan memasukkan salah satu perintah berikut. Untuk sebuah klaster Amazon EKS dengan tipe peluncuran EC2, masukkan:

```
kubectl apply -f prometheus-eks.yaml
```

   Untuk sebuah klaster Amazon EKS dengan tipe peluncuran Fargate, Anda perlu memasukkan perintah berikut. Ganti *MyCluster*dan *wilayah* dengan nilai agar sesuai dengan penerapan Anda.

```
cat prometheus-eks-fargate.yaml \
| sed "s/{{cluster_name}}/MyCluster/;s/{{region_name}}/region/" \
```

```
| kubectl apply -f -
```

Untuk sebuah klaster Kubernetes, masukkan perintah berikut. Ganti *MyCluster*dan *wilayah* dengan nilai agar sesuai dengan penerapan Anda.

```
cat prometheus-k8s.yaml \
| sed "s/{{cluster_name}}/MyCluster/;s/{{region_name}}/region/" \
| kubectl apply -f -
```

Setelah Anda melakukannya, Anda akan melihat sebuah log stream baru bernama kubernetes-apiservers di grup log /aws/containerinsights/***cluster_name***/prometheus. Log stream ini harus mencakup peristiwa log dengan penetapan format metrik tersemat seperti berikut:

```
{
   "CloudWatchMetrics":[
      {
         "Metrics":[
            {
               "Name":"apiserver_request_total"
            }
         ],
         "Dimensions":[
            [
               "ClusterName",
               "Service"
            ]
         ],
         "Namespace":"ContainerInsights/Prometheus"
      }
   ],
   "ClusterName":"my-cluster-name",
   "Namespace":"default",
   "Service":"kubernetes",
   "Timestamp":"1592267020339",
   "Version":"0",
   "apiserver_request_count":0,
   "apiserver_request_total":0,
   "code":"0",
   "component":"apiserver",
   "contentType":"application/json",
   "instance":"192.0.2.0:443",
```

```
    "job":"kubernetes-apiservers",
    "prom_metric_type":"counter",
    "resource":"pods",
    "scope":"namespace",
    "verb":"WATCH",
    "version":"v1"
}
```

Anda dapat melihat metrik Anda di CloudWatch konsol di namespace ContainerInsights/Prometheus. Anda juga dapat secara opsional membuat CloudWatch dasbor untuk metrik Prometheus Kubernetes API Server Anda.

(Opsional) Membuat sebuah dasbor untuk metrik-metrik Server API Kubernetes.

Untuk melihat metrik Kubernetes API Server di dasbor Anda, Anda harus terlebih dahulu menyelesaikan langkah-langkah di bagian sebelumnya untuk mulai mengumpulkan metrik ini. CloudWatch

Cara membuat sebuah dasbor untuk metrik-metrik Server API Kubernetes

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pastikan Anda telah memilih AWS Wilayah yang benar.
3. Pada panel navigasi, silakan pilih Dasbor.
4. Pilih Buat Dasbor. Masukkan nama untuk dasbor baru tersebut, dan pilih Buat dasbor.
5. Di Tambahkan ke dasbor ini, pilih Batalkan.
6. Pilih Tindakan, Tampilkan/sunting sumber.
7. Unduh file JSON berikut: Sumber Dasbor API Kubernetes.
8. Buka file JSON yang Anda unduh dengan editor teks, dan lakukan perubahan-perubahan berikut:

   - Ganti semua string `{{YOUR_CLUSTER_NAME}}` dengan nama persis seperti nama klaster Anda. Anda harus memastikan untuk tidak menambahkan spasi putih sebelum atau setelah teks.
   - Ganti semua string `{{YOUR_AWS_REGION}}` dengan nama Wilayah tempat metrik-metrik dikumpulkan. Sebagai contoh, `us-west-2`. Pastikan untuk tidak menambahkan spasi putih sebelum atau setelah teks.
9. Salin seluruh gumpalan JSON dan tempel ke kotak teks di CloudWatch konsol, ganti apa yang sudah ada di dalam kotak.

10. Pilih Perbarui, Simpan dasbor.

(Opsional) Menyiapkan sampel beban kerja Amazon EKS terkontainer untuk pengujian metrik Prometheus

Untuk menguji dukungan metrik Prometheus CloudWatch di Container Insights, Anda dapat menyiapkan satu atau beberapa beban kerja kontainer berikut. CloudWatch Agen dengan dukungan Prometheus secara otomatis mengumpulkan metrik dari masing-masing beban kerja ini. Untuk melihat metrik-metrik yang dikumpulkan secara bawaan, silakan lihat Metrik Prometheus dikumpulkan oleh agen CloudWatch .

Sebelum Anda dapat melakukan instalasi salah satu beban kerja ini, Anda harus melakukan instalasi Helm 3.x dengan memasukkan perintah-perintah berikut:

```
brew install helm
```

Untuk informasi selengkapnya, silakan lihat Helm.

Topik

- Menyiapkan beban kerja AWS App Mesh sampel untuk Amazon EKS dan Kubernetes
- Siapkan NGINX dengan lalu lintas sampel di Amazon EKS dan Kubernetes
- Siapkan memcached dengan sebuah pengekspor metrik di Amazon EKS dan Kubernetes
- Menyiapkan beban kerja sampel Java/JMX pada Amazon EKS dan Kubernetes
- Siapkan HAProxy dengan sebuah pengekspor metrik di Amazon EKS dan Kubernetes
- Tutorial untuk menambahkan sebuah target scraping Prometheus baru: Redis di klaster Amazon EKS dan Kubernetes

Menyiapkan beban kerja AWS App Mesh sampel untuk Amazon EKS dan Kubernetes

Dukungan Prometheus dalam dukungan Container Insights. CloudWatch AWS App Mesh Bagian-bagian berikut akan menjelaskan cara menyiapkan App Mesh.

CloudWatch Wawasan Kontainer juga dapat mengumpulkan Log Akses Utusan App Mesh. Untuk informasi selengkapnya, lihat (Opsional) Mengaktifkan log akses App Mesh Envoy.

Topik

- Menyiapkan beban kerja AWS App Mesh sampel pada sebuah klaster Amazon EKS dengan tipe peluncuran EC2 atau sebuah klaster Kubernetes
- Siapkan AWS App Mesh contoh beban kerja di klaster Amazon EKS dengan tipe peluncuran Fargate

Menyiapkan beban kerja AWS App Mesh sampel pada sebuah klaster Amazon EKS dengan tipe peluncuran EC2 atau sebuah klaster Kubernetes

Gunakan petunjuk ini jika Anda menyiapkan App Mesh pada sebuah klaster yang menjalankan Amazon EKS dengan tipe peluncuran EC2, atau sebuah klaster Kubernetes.

Mengonfigurasi izin IAM

Anda harus menambahkan AWSAppMeshFullAccesskebijakan ke peran IAM untuk Amazon EKS atau grup node Kubernetes Anda. Pada Amazon EKS, nama grup simpul ini terlihat mirip dengan `eksctl-integ-test-eks-prometheus-NodeInstanceRole-ABCDEFHIJKL`. Di Kubernetes, mungkin terlihat mirip dengan `nodes.integ-test-kops-prometheus.k8s.local`.

Melakukan instalasi App Mesh

Untuk melakukan instalasi pengendali App Mesh Kubernetes, ikuti petunjuk-petunjuk yang diuraikan di Kontroler App Mesh.

Melakukan instalasi aplikasi sampel

aws-app-mesh-examplesberisi beberapa penelusuran App Mesh Kubernetes. Untuk tutorial ini, Anda melakukan instalasi untuk sebuah aplikasi warna sampel yang menunjukkan bagaimana rute http dapat menggunakan header untuk mencocokkan permintaan masuk.

Cara menggunakan sebuah aplikasi App Mesh sampel untuk menguji Wawasan Kontainer

1. Instal aplikasi dengan menggunakan petunjuk ini: https://github.com/aws/aws-app-mesh-examples/tree/main/walkthroughs/howto-k8s-http-headers.
2. Luncurkan sebuah pod curler untuk menghasilkan lalu lintas:

   ```
   kubectl -n default run -it curler --image=tutum/curl /bin/bash
   ```

3. Lakukan curling pada titik akhir yang berbeda dengan mengubah header HTTP. Jalankan perintah curl beberapa kali, sebagaimana yang ditunjukkan:

```
curl -H "color_header: blue" front.howto-k8s-http-headers.svc.cluster.local:8080/;
 echo;

curl -H "color_header: red" front.howto-k8s-http-headers.svc.cluster.local:8080/;
 echo;

curl -H "color_header: yellow" front.howto-k8s-http-
headers.svc.cluster.local:8080/; echo;
```

4. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
5. Di AWS Wilayah tempat klaster Anda berjalan, pilih Metrik di panel navigasi. Metrik berada di namespace ContainerInsights/Prometheus.
6. Untuk melihat peristiwa CloudWatch Log, pilih Grup log di panel navigasi. Peristiwa-peristiwa berada dalam grup log `/aws/containerinsights/`*`your_cluster_name`*`/prometheus` di dalam log stream `kubernetes-pod-appmesh-envoy`.

Menghapus lingkungan pengujian App Mesh

Setelah selesai menggunakan App Mesh dan aplikasi sampel, gunakan perintah berikut untuk menghapus sumber daya yang tidak diperlukan. Menghapus aplikasi sampel dengan memasukkan perintah berikut:

```
cd aws-app-mesh-examples/walkthroughs/howto-k8s-http-headers/
kubectl delete -f _output/manifest.yaml
```

Menghapus pengendali App Mesh dengan memasukkan perintah berikut:

```
helm delete appmesh-controller -n appmesh-system
```

Siapkan AWS App Mesh contoh beban kerja di klaster Amazon EKS dengan tipe peluncuran Fargate

Gunakan petunjuk-petunjuk ini jika Anda hendak menyiapkan App Mesh pada sebuah klaster yang menjalankan Amazon EKS dengan tipe peluncuran Fargate.

Mengonfigurasi izin IAM

Untuk menyiapkan izin IAM, masukkan perintah berikut. Ganti *`MyCluster`*dengan nama cluster Anda.

```
eksctl create iamserviceaccount --cluster MyCluster \
 --namespace howto-k8s-fargate \
 --name appmesh-pod \
 --attach-policy-arn arn:aws:iam::aws:policy/AWSAppMeshEnvoyAccess \
 --attach-policy-arn arn:aws:iam::aws:policy/AWSCloudMapDiscoverInstanceAccess \
 --attach-policy-arn arn:aws:iam::aws:policy/AWSXRayDaemonWriteAccess \
 --attach-policy-arn arn:aws:iam::aws:policy/CloudWatchLogsFullAccess \
 --attach-policy-arn arn:aws:iam::aws:policy/AWSAppMeshFullAccess \
 --attach-policy-arn arn:aws:iam::aws:policy/AWSCloudMapFullAccess \
 --override-existing-serviceaccounts \
 --approve
```

Melakukan instalasi App Mesh

Untuk melakukan instalasi pengendali App Mesh Kubernetes, ikuti petunjuk-petunjuk yang diuraikan di Kontroler App Mesh. Pastikan untuk mengikuti petunjuk-petunjuk untuk Amazon EKS dengan tipe peluncuran Fargate.

Melakukan instalasi aplikasi sampel

aws-app-mesh-examplesberisi beberapa penelusuran App Mesh Kubernetes. Untuk tutorial ini, Anda melakukan instalasi aplikasi warna sampel yang bekerja untuk klaster Amazon EKS dengan tipe peluncuran Fargate.

Cara menggunakan sebuah aplikasi App Mesh sampel untuk menguji Wawasan Kontainer

1. Instal aplikasi dengan menggunakan petunjuk ini: https://github.com/aws/aws-app-mesh-examples/tree/main/walkthroughs/howto-k8s-fargate.

   Instruksi-instruksi tersebut mengasumsikan bahwa Anda membuat sebuah klaster baru dengan profil Fargate yang benar. Jika Anda ingin menggunakan sebuah klaster Amazon EKS yang telah Anda siapkan, maka Anda dapat menggunakan perintah-perintah berikut untuk menyiapkan klaster tersebut untuk demonstrasi ini. Ganti *MyCluster*dengan nama cluster Anda.

   ```
   eksctl create iamserviceaccount --cluster MyCluster \
    --namespace howto-k8s-fargate \
    --name appmesh-pod \
    --attach-policy-arn arn:aws:iam::aws:policy/AWSAppMeshEnvoyAccess \
    --attach-policy-arn arn:aws:iam::aws:policy/AWSCloudMapDiscoverInstanceAccess \
    --attach-policy-arn arn:aws:iam::aws:policy/AWSXRayDaemonWriteAccess \
    --attach-policy-arn arn:aws:iam::aws:policy/CloudWatchLogsFullAccess \
   ```

```
 --attach-policy-arn arn:aws:iam::aws:policy/AWSAppMeshFullAccess \
 --attach-policy-arn arn:aws:iam::aws:policy/AWSCloudMapFullAccess \
 --override-existing-serviceaccounts \
 --approve
```

```
eksctl create fargateprofile --cluster MyCluster \
--namespace howto-k8s-fargate --name howto-k8s-fargate
```

2. Port meneruskan deployment aplikasi depan:

```
kubectl -n howto-k8s-fargate port-forward deployment/front 8080:8080
```

3. Lakukan curling pada aplikasi depan:

```
while true; do  curl -s http://localhost:8080/color; sleep 0.1; echo ; done
```

4. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
5. Di AWS Wilayah tempat klaster Anda berjalan, pilih Metrik di panel navigasi. Metrik berada di namespace ContainerInsights/Prometheus.
6. Untuk melihat peristiwa CloudWatch Log, pilih Grup log di panel navigasi. Peristiwa-peristiwa berada dalam grup log `/aws/containerinsights/`*`your_cluster_name`*`/prometheus` di dalam log stream `kubernetes-pod-appmesh-envoy`.

Menghapus lingkungan pengujian App Mesh

Setelah selesai menggunakan App Mesh dan aplikasi sampel, gunakan perintah berikut untuk menghapus sumber daya yang tidak diperlukan. Menghapus aplikasi sampel dengan memasukkan perintah berikut:

```
cd aws-app-mesh-examples/walkthroughs/howto-k8s-fargate/
kubectl delete -f _output/manifest.yaml
```

Menghapus pengendali App Mesh dengan memasukkan perintah berikut:

```
helm delete appmesh-controller -n appmesh-system
```

Siapkan NGINX dengan lalu lintas sampel di Amazon EKS dan Kubernetes

NGINX adalah sebuah server web yang juga dapat digunakan sebagai penyeimbang beban dan proksi balik. Untuk informasi selengkapnya tentang bagaimana Kubernetes menggunakan NGINX untuk ingress, silakan lihat kubernetes/ingress-nginx.

Untuk melakukan instalasi Ingress-NGINX dengan layanan lalu lintas sampel untuk menguji dukungan Prometheus Wawasan Kontainer

1. Masukkan perintah berikut untuk menambahkan repo ingress-nginx Helm:

```
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
```

2. Masukkan perintah-perintah berikut.

```
kubectl create namespace nginx-ingress-sample

helm install my-nginx ingress-nginx/ingress-nginx \
--namespace nginx-ingress-sample \
--set controller.metrics.enabled=true \
--set-string controller.metrics.service.annotations."prometheus\.io/port"="10254" \
--set-string controller.metrics.service.annotations."prometheus\.io/scrape"="true"
```

3. Periksa apakah layanan-layanan dimulai dengan benar dengan memasukkan perintah berikut:

```
kubectl get service -n nginx-ingress-sample
```

   Output dari perintah ini akan menampilkan beberapa kolom, termasuk sebuah kolom `EXTERNAL-IP`.

4. Mengatur variabel `EXTERNAL-IP` terhadap nilai kolom `EXTERNAL-IP` dalam baris pengendali ingress NGINX.

```
EXTERNAL_IP=your-nginx-controller-external-ip
```

5. Mulai beberapa lalu lintas NGINX sampel dengan memasukkan perintah berikut.

```
SAMPLE_TRAFFIC_NAMESPACE=nginx-sample-traffic
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-
prometheus/sample_traffic/nginx-traffic/nginx-traffic-sample.yaml |
sed "s/{{external_ip}}/$EXTERNAL_IP/g" |
```

```
sed "s/{{namespace}}/$SAMPLE_TRAFFIC_NAMESPACE/g" |
kubectl apply -f -
```

6. Masukkan perintah berikut untuk mengonfirmasi bahwa ketiga pod berada dalam status `Running`.

   ```
   kubectl get pod -n $SAMPLE_TRAFFIC_NAMESPACE
   ```

   Jika mereka berjalan, Anda akan segera melihat metrik di namespace ContainerInsights/Prometheus.

Cara menghapus NGINX dan aplikasi lalu lintas sampel

1. Hapus layanan lalu lintas sampel dengan memasukkan perintah berikut:

   ```
   kubectl delete namespace $SAMPLE_TRAFFIC_NAMESPACE
   ```

2. Hapus egress NGINX dengan nama rilis Helm.

   ```
   helm uninstall my-nginx --namespace nginx-ingress-sample
   kubectl delete namespace nginx-ingress-sample
   ```

Siapkan memcached dengan sebuah pengekspor metrik di Amazon EKS dan Kubernetes

Memcached adalah sebuah sistem caching objek memori sumber terbuka. Untuk informasi selengkapnya, silakan lihat Apa itu Memcached?

Jika Anda menjalankan memcached pada sebuah klaster dengan tipe peluncuran Fargate, maka Anda perlu meyiapkan profil Fargate sebelum melakukan langkah-langkah dalam prosedur ini. Untuk menyiapkan profilnya, masukkan perintah berikut. Ganti *MyCluster*dengan nama cluster Anda.

```
eksctl create fargateprofile --cluster MyCluster \
--namespace memcached-sample --name memcached-sample
```

Cara melakukan instalasi dengan pengekspor metrik untuk menguji dukungan Wawasan Kontainer Prometheus

1. Masukkan perintah berikut untuk menambahkan repo:

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

2. Masukkan perintah berikut untuk membuat sebuah namespace baru:

```
kubectl create namespace memcached-sample
```

3. Masukkan perintah berikut untuk melakukan instalasi Memcached

```
helm install my-memcached bitnami/memcached --namespace memcached-sample \
--set metrics.enabled=true \
--set-string serviceAnnotations.prometheus\\.io/port="9150" \
--set-string serviceAnnotations.prometheus\\.io/scrape="true"
```

4. Masukkan perintah berikut untuk mengonfirmasi keterangan dari layanan yang berjalan:

```
kubectl describe service my-memcached-metrics -n memcached-sample
```

   Anda seharusnya melihat dua keterangan berikut:

```
Annotations:   prometheus.io/port: 9150
               prometheus.io/scrape: true
```

Cara menghapus instalasi memcached

- Masukkan perintah-perintah berikut.

```
helm uninstall my-memcached --namespace memcached-sample
kubectl delete namespace memcached-sample
```

Menyiapkan beban kerja sampel Java/JMX pada Amazon EKS dan Kubernetes

JMX Exporter adalah sebuah pengekspor Prometheus resmi yang dapat melakukan scraping dan mengekspos JMX mBeans sebagai metrik-metrik Prometheus. Untuk informasi selengkapnya, silakan lihat prometheus/jmx_exporter.

Wawasan Kontainer dapat mengumpulkan metrik-metrik Prometheus yang telah ditentukan sebelumnya dari Java Virtual Machine (JVM), Java, dan Tomcat (Catalina) dengan menggunakan JMX Exporter.

Konfigurasi scraping Prometheus bawaan

Secara default, CloudWatch agen dengan dukungan Prometheus akan mengikis metrik Java/JMX Prometheus dari setiap pod di kluster Amazon EKS atau Kubernetes. `http://CLUSTER_IP:9404/metrics` Hal ini dilakukan oleh penemuan `role: pod` Prometheus `kubernetes_sd_config`. 9404 adalah port bawaan yang dialokasikan untuk JMX Exporter oleh Prometheus. Untuk informasi selengkapnya tentang penemuan `role: pod`, silakan lihat pod. Anda dapat mengonfigurasi JMX Exporter untuk mengekspos metrik-metrik sebuah pada port atau metrik_path yang berbeda. Jika Anda mengubah port atau jalur, perbarui jmx scrape_config default di peta konfigurasi agen. CloudWatch Jalankan perintah berikut untuk mendapatkan konfigurasi Prometheus CloudWatch agen saat ini:

```
kubectl describe cm prometheus-config -n amazon-cloudwatch
```

Bidang yang harus diubah adalah bidang `/metrics` dan `regex: '.*:9404$'`, seperti yang disorot dalam contoh berikut.

```
job_name: 'kubernetes-jmx-pod'
sample_limit: 10000
metrics_path: /metrics
kubernetes_sd_configs:
- role: pod
relabel_configs:
- source_labels: [__address__]
  action: keep
  regex: '.*:9404$'
- action: replace
  regex: (.+)
  source_labels:
```

Konfigurasi scraping Prometheus lainnya

Jika Anda mengekspos aplikasi yang berjalan pada sekumpulan pod dengan pengekspor Java/JMX Prometheus oleh Layanan Kubernetes, maka Anda juga dapat beralih untuk menggunakan penemuan `role: service` atau penemuan `role: endpoint` dari Prometheus

`kubernetes_sd_config`. Untuk informasi selengkapnya tentang metode penemuan ini, silakan lihat _layanan, titik akhir, dan _<kubernetes_sd_config>..

Lebih banyak label meta disediakan oleh dua mode penemuan layanan ini yang dapat berguna bagi Anda untuk membangun dimensi CloudWatch metrik. Sebagai contoh, Anda dapat mengubah label `__meta_kubernetes_service_name` menjadi `Service` dan memasukkannya ke dalam dimensi metrik Anda. Untuk informasi selengkapnya tentang menyesuaikan CloudWatch metrik dan dimensinya, lihat. _CloudWatch konfigurasi agen untuk Prometheus

Citra Docker dengan JMX Exporter

Berikutnya, buat sebuah citra Docker. Bagian-bagian berikut memberikan dua contoh Dockerfile.

Setelah membangun citra tersebut, muatkan citra itu ke Amazon EKS atau Kubernetes, dan kemudian jalankan perintah berikut untuk memverifikasi bahwa metrik-metrik Prometheus diekspos oleh `JMX_EXPORTER` di port 9404. Ganti *`$JAR_SAMPLE_TRAFFIC_POD`* dengan nama pod yang berjalan dan ganti *`$JAR_SAMPLE_TRAFFIC_NAMESPACE`* dengan namespace aplikasi Anda.

Jika Anda menjalankan JMX Exporter pada sebuah klaster dengan tipe peluncuran Fargate, maka Anda juga perlu meyiapkan sebuah profil Fargate sebelum melakukan langkah-langkah dalam prosedur ini. Untuk menyiapkan profilnya, masukkan perintah berikut. Ganti *`MyCluster`*dengan nama cluster Anda.

```
eksctl create fargateprofile --cluster MyCluster \
--namespace $JAR_SAMPLE_TRAFFIC_NAMESPACE\
 --name $JAR_SAMPLE_TRAFFIC_NAMESPACE
```

```
kubectl exec $JAR_SAMPLE_TRAFFIC_POD -n $JARCAT_SAMPLE_TRAFFIC_NAMESPACE -- curl
 http://localhost:9404
```

Contoh: Citra Docker Apache Tomcat dengan metrik-metrik Prometheus

Server Apache Tomcat membuka JMX mBeans secara bawaan. Anda dapat mengintegrasikan JMX Exporter dengan Tomcat untuk mengekspos JMX mBeans sebagai metrik-metrik Prometheus. Contoh Dockerfile berikut menunjukkan langkah-langkah untuk membangun sebuah citra pengujian:

```
# From Tomcat 9.0 JDK8 OpenJDK
FROM tomcat:9.0-jdk8-openjdk

RUN mkdir -p /opt/jmx_exporter
```

```
COPY ./jmx_prometheus_javaagent-0.12.0.jar /opt/jmx_exporter
COPY ./config.yaml /opt/jmx_exporter
COPY ./setenv.sh /usr/local/tomcat/bin
COPY your web application.war /usr/local/tomcat/webapps/

RUN chmod  o+x /usr/local/tomcat/bin/setenv.sh

ENTRYPOINT ["catalina.sh", "run"]
```

Daftar berikut menjelaskan empat baris COPY dalam Dockerfile ini.

- Unduh file jar JMX Exporter terbaru dari https://github.com/prometheus/jmx_exporter.
- `config.yaml` adalah file konfigurasi JMX Exporter. Untuk informasi selengkapnya, silakan lihat https://github.com/prometheus/jmx_exporter#Configuration.

  Berikut ini adalah sebuah file konfigurasi sampel untuk Java dan Tomcat:

```
lowercaseOutputName: true
lowercaseOutputLabelNames: true

rules:
- pattern: 'java.lang<type=OperatingSystem><>(FreePhysicalMemorySize|
TotalPhysicalMemorySize|FreeSwapSpaceSize|TotalSwapSpaceSize|SystemCpuLoad|
ProcessCpuLoad|OpenFileDescriptorCount|AvailableProcessors)'
  name: java_lang_OperatingSystem_$1
  type: GAUGE

- pattern: 'java.lang<type=Threading><>(TotalStartedThreadCount|ThreadCount)'
  name: java_lang_threading_$1
  type: GAUGE

- pattern: 'Catalina<type=GlobalRequestProcessor, name=\"(\w+-\w+)-(\d+)\"><>(\w+)'
  name: catalina_globalrequestprocessor_$3_total
  labels:
    port: "$2"
    protocol: "$1"
  help: Catalina global $3
  type: COUNTER

- pattern: 'Catalina<j2eeType=Servlet, WebModule=//([-a-zA-Z0-9+&@#/%?=~_|!:.,;]*[-
a-zA-Z0-9+&@#/%=~_|]), name=([-a-zA-Z0-9+/$%~_-|!.]*), J2EEApplication=none,
 J2EEServer=none><>(requestCount|maxTime|processingTime|errorCount)'
  name: catalina_servlet_$3_total
```

```
  labels:
    module: "$1"
    servlet: "$2"
  help: Catalina servlet $3 total
  type: COUNTER

- pattern: 'Catalina<type=ThreadPool, name="(\w+-\w+)-(\d+)"><>(currentThreadCount|
currentThreadsBusy|keepAliveCount|pollerThreadCount|connectionCount)'
  name: catalina_threadpool_$3
  labels:
    port: "$2"
    protocol: "$1"
  help: Catalina threadpool $3
  type: GAUGE

- pattern: 'Catalina<type=Manager, host=([-a-zA-Z0-9+&@#/%?=~_|!:.,;]*[-a-zA-
Z0-9+&@#/%=~_|]), context=([-a-zA-Z0-9+/$%~_-|!.]*)><>(processingTime|sessionCounter|
rejectedSessions|expiredSessions)'
  name: catalina_session_$3_total
  labels:
    context: "$2"
    host: "$1"
  help: Catalina session $3 total
  type: COUNTER

- pattern: ".*"
```

- `setenv.sh` adalah sebuah skrip perusahaan rintisan Tomcat untuk memulai JMX exporter bersama dengan Tomcat dan mengekspos metrik-metrik Prometheus di port 9404 dari host lokal. Ia juga menyediakan JMX Exporter dengan jalur file `config.yaml`.

```
$ cat setenv.sh
export JAVA_OPTS="-javaagent:/opt/jmx_exporter/
jmx_prometheus_javaagent-0.12.0.jar=9404:/opt/jmx_exporter/config.yaml $JAVA_OPTS"
```

- Web application.war Anda adalah file `war` aplikasi web Anda yang akan dimuat oleh Tomcat.

Bangun sebuah citra docker dengan konfigurasi ini dan unggah citra tersebut ke sebuah repositori citra.

Contoh: Citra Docker Aplikasi Jar Java dengan metrik-metrik Prometheus

Contoh Dockerfile berikut menunjukkan langkah-langkah untuk membangun sebuah citra pengujian:

```
# Alpine Linux with OpenJDK JRE
FROM openjdk:8-jre-alpine

RUN mkdir -p /opt/jmx_exporter

COPY ./jmx_prometheus_javaagent-0.12.0.jar /opt/jmx_exporter
COPY ./SampleJavaApplication-1.0-SNAPSHOT.jar /opt/jmx_exporter
COPY ./start_exporter_example.sh /opt/jmx_exporter
COPY ./config.yaml /opt/jmx_exporter

RUN chmod -R o+x /opt/jmx_exporter
RUN apk add curl

ENTRYPOINT exec /opt/jmx_exporter/start_exporter_example.sh
```

Daftar berikut menjelaskan empat baris `COPY` dalam Dockerfile ini.

- Unduh file jar JMX Exporter terbaru dari https://github.com/prometheus/jmx_exporter.
- `config.yaml` adalah file konfigurasi JMX Exporter. Untuk informasi selengkapnya, silakan lihat https://github.com/prometheus/jmx_exporter#Configuration.

  Berikut ini adalah sebuah file konfigurasi sampel untuk Java dan Tomcat:

```
lowercaseOutputName: true
lowercaseOutputLabelNames: true

rules:
- pattern: 'java.lang<type=OperatingSystem><>(FreePhysicalMemorySize|
TotalPhysicalMemorySize|FreeSwapSpaceSize|TotalSwapSpaceSize|SystemCpuLoad|
ProcessCpuLoad|OpenFileDescriptorCount|AvailableProcessors)'
  name: java_lang_OperatingSystem_$1
  type: GAUGE

- pattern: 'java.lang<type=Threading><>(TotalStartedThreadCount|ThreadCount)'
  name: java_lang_threading_$1
  type: GAUGE

- pattern: 'Catalina<type=GlobalRequestProcessor, name=\"(\w+-\w+)-(\d+)\"><>(\w+)'
  name: catalina_globalrequestprocessor_$3_total
  labels:
    port: "$2"
    protocol: "$1"
```

```
  help: Catalina global $3
  type: COUNTER

- pattern: 'Catalina<j2eeType=Servlet, WebModule=//([-a-zA-Z0-9+&@#/%?=~_|!:.,;]*[-
a-zA-Z0-9+&@#/%=~_|]), name=([-a-zA-Z0-9+/$%~_-|!.]*), J2EEApplication=none,
 J2EEServer=none><>(requestCount|maxTime|processingTime|errorCount)'
  name: catalina_servlet_$3_total
  labels:
    module: "$1"
    servlet: "$2"
  help: Catalina servlet $3 total
  type: COUNTER

- pattern: 'Catalina<type=ThreadPool, name="(\w+-\w+)-(\d+)"><>(currentThreadCount|
currentThreadsBusy|keepAliveCount|pollerThreadCount|connectionCount)'
  name: catalina_threadpool_$3
  labels:
    port: "$2"
    protocol: "$1"
  help: Catalina threadpool $3
  type: GAUGE

- pattern: 'Catalina<type=Manager, host=([-a-zA-Z0-9+&@#/%?=~_|!:.,;]*[-a-zA-
Z0-9+&@#/%=~_|]), context=([-a-zA-Z0-9+/$%~_-|!.]*)><>(processingTime|sessionCounter|
rejectedSessions|expiredSessions)'
  name: catalina_session_$3_total
  labels:
    context: "$2"
    host: "$1"
  help: Catalina session $3 total
  type: COUNTER

- pattern: ".*"
```

- `start_exporter_example.sh` adalah skrip untuk memulai aplikasi JAR dengan metrik-metrik Prometheus yang diekspor. Ia juga menyediakan JMX Exporter dengan jalur file `config.yaml`.

```
$ cat start_exporter_example.sh
java -javaagent:/opt/jmx_exporter/jmx_prometheus_javaagent-0.12.0.jar=9404:/
opt/jmx_exporter/config.yaml -cp  /opt/jmx_exporter/SampleJavaApplication-1.0-
SNAPSHOT.jar com.gubupt.sample.app.App
```

- SampleJavaApplication-1.0-snapshot.jar adalah contoh file jar aplikasi Java. Ganti file tersebut dengan aplikasi Java yang ingin Anda pantau.

Bangun sebuah citra docker dengan konfigurasi ini dan unggah citra tersebut ke sebuah repositori citra.

Siapkan HAProxy dengan sebuah pengekspor metrik di Amazon EKS dan Kubernetes

HAProxy adalah sebuah aplikasi proksi sumber terbuka. Untuk informasi selengkapnya, silakan lihat HAProxy.

Jika Anda menjalankan HAProxy pada sebuah klaster dengan tipe peluncuran Fargate, maka Anda perlu meyiapkan profil Fargate sebelum melakukan langkah-langkah dalam prosedur ini. Untuk menyiapkan profilnya, masukkan perintah berikut. Ganti *MyCluster*dengan nama cluster Anda.

```
eksctl create fargateprofile --cluster MyCluster \
--namespace haproxy-ingress-sample --name haproxy-ingress-sample
```

Untuk melakukan instalasi HAProxy dengan sebuah pengekspor metrik untuk menguji dukungan Wawasan Kontainer Prometheus

1. Masukkan perintah berikut untuk menambahkan repo inkubator Helm:

   ```
   helm repo add haproxy-ingress https://haproxy-ingress.github.io/charts
   ```

2. Masukkan perintah berikut untuk membuat sebuah namespace baru:

   ```
   kubectl create namespace haproxy-ingress-sample
   ```

3. Masukkan perintah-perintah berikut untuk melakukan instalasi HAProxy:

   ```
   helm install haproxy haproxy-ingress/haproxy-ingress \
   --namespace haproxy-ingress-sample \
   --set defaultBackend.enabled=true \
   --set controller.stats.enabled=true \
   --set controller.metrics.enabled=true \
   --set-string controller.metrics.service.annotations."prometheus\.io/port"="9101" \
   --set-string controller.metrics.service.annotations."prometheus\.io/scrape"="true"
   ```

4. Masukkan perintah berikut untuk mengonfirmasi keterangan dari layanan:

```
kubectl describe service haproxy-haproxy-ingress-metrics -n haproxy-ingress-sample
```

Anda seharusnya melihat dua keterangan berikut.

```
Annotations:   prometheus.io/port: 9101
               prometheus.io/scrape: true
```

Cara menghapus instalasi HAProxy

- Masukkan perintah-perintah berikut.

```
helm uninstall haproxy --namespace haproxy-ingress-sample
kubectl delete namespace haproxy-ingress-sample
```

Tutorial untuk menambahkan sebuah target scraping Prometheus baru: Redis di klaster Amazon EKS dan Kubernetes

Tutorial ini menyediakan Anda dengan pengenalan langsung untuk melakukan scraping pada metrik-metrik Prometheus dari sebuah aplikasi Redis sampel di Amazon EKS dan Kubernetes. Redis (https://redis.io/) adalah sebuah penyimpanan struktur data di dalam memori yang bersifat sumber terbuka (dilisensikan BSD), yang digunakan sebagai basis data, cache, dan broker pesan. Untuk informasi selengkapnya, silakan lihat redis.

redis_exporter (dilisensikan di bawah Lisensi MIT) digunakan untuk mengekspos metrik-metrik Prometheus Redis pada port yang ditentukan (bawaan: 0.0.0.0:9121) Untuk informasi selengkapnya, silakan lihat redis_exporter.

Citra Docker dalam dua repositori Docker Hub berikut digunakan dalam tutorial ini:

- redis
- redis_exporter

Untuk melakukan instalasi sebuah beban kerja Redis sampel yang mengekspos metrik-metrik Prometheus

1. Atur namespace untuk beban kerja Redis sampel.

```
REDIS_NAMESPACE=redis-sample
```

2. Jika Anda menjalankan Redis pada sebuah klaster dengan tipe peluncuran Fargate, maka Anda perlu menyiapkan profil Fargate. Untuk menyiapkan profilnya, masukkan perintah berikut. Ganti *MyCluster*dengan nama cluster Anda.

```
eksctl create fargateprofile --cluster MyCluster \
--namespace $REDIS_NAMESPACE --name $REDIS_NAMESPACE
```

3. Masukkan perintah berikut untuk melakukan instalasi beban kerja Redis sampel.

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-
prometheus/sample_traffic/redis/redis-traffic-sample.yaml \
| sed "s/{{namespace}}/$REDIS_NAMESPACE/g" \
| kubectl apply -f -
```

4. Instalasi tersebut mencakup sebuah layanan bernama `my-redis-metrics` yang mengekspos metrik-metrik Prometheus Redis pada port 9121 Masukkan perintah berikut untuk mendapatkan detail layanan tersebut:

```
kubectl describe service/my-redis-metrics  -n $REDIS_NAMESPACE
```

Di `Annotations` bagian hasil, Anda akan melihat dua anotasi yang cocok dengan konfigurasi scrape Prometheus agen, sehingga dapat menemukan beban kerja CloudWatch secara otomatis:

```
prometheus.io/port: 9121
prometheus.io/scrape: true
```

Konfigurasi scraping Prometheus terkait dapat ditemukan di bagian `- job_name: kubernetes-service-endpoints` dari `kubernetes-eks.yaml` atau `kubernetes-k8s.yaml`.

Untuk mulai mengumpulkan metrik Redis Prometheus di CloudWatch

1. Unduh versi terbaru dari file `kubernetes-eks.yaml` atau `kubernetes-k8s.yaml` dengan memasukkan salah satu perintah dari perintah-perintah berikut. Untuk sebuah klaster Amazon EKS dengan tipe peluncuran EC2, masukkan perintah ini:

   ```
   curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-eks.yaml
   ```

   Untuk sebuah klaster Amazon EKS dengan tipe peluncuran Fargate, masukkan perintah ini:

   ```
   curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-eks-fargate.yaml
   ```

   Untuk sebuah klaster Kubernetes yang berjalan di sebuah instans Amazon EC2, masukkan perintah ini:

   ```
   curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-k8s.yaml
   ```

2. Buka file dengan editor teks, dan temukan bagian `cwagentconfig.json`. Tambahkan subbagian berikut dan simpan perubahannya. Pastikan bahwa indentasi mengikuti pola yang ada saat itu.

   ```
   {
     "source_labels": ["pod_name"],
     "label_matcher": "^redis-instance$",
     "dimensions": [["Namespace","ClusterName"]],
     "metric_selectors": [
       "^redis_net_(in|out)put_bytes_total$",
       "^redis_(expired|evicted)_keys_total$",
       "^redis_keyspace_(hits|misses)_total$",
       "^redis_memory_used_bytes$",
       "^redis_connected_clients$"
     ]
   },
   {
     "source_labels": ["pod_name"],
   ```

```
  "label_matcher": "^redis-instance$",
  "dimensions": [["Namespace","ClusterName","cmd"]],
  "metric_selectors": [
    "^redis_commands_total$"
  ]
},
{
  "source_labels": ["pod_name"],
  "label_matcher": "^redis-instance$",
  "dimensions": [["Namespace","ClusterName","db"]],
  "metric_selectors": [
    "^redis_db_keys$"
  ]
},
```

Bagian yang Anda tambahkan menempatkan metrik Redis ke daftar izin CloudWatch agen. Untuk daftar metrik ini, silakan lihat bagian berikut.

3. Jika Anda sudah memiliki CloudWatch agen dengan dukungan Prometheus yang diterapkan di cluster ini, Anda harus menghapusnya dengan memasukkan perintah berikut.

```
kubectl delete deployment cwagent-prometheus -n amazon-cloudwatch
```

4. Terapkan CloudWatch agen dengan konfigurasi Anda yang diperbarui dengan memasukkan salah satu perintah berikut. Ganti *MyCluster*dan *wilayah* agar sesuai dengan pengaturan Anda.

   Untuk sebuah klaster Amazon EKS dengan tipe peluncuran EC2, masukkan perintah ini:

```
kubectl apply -f prometheus-eks.yaml
```

   Untuk sebuah klaster Amazon EKS dengan tipe peluncuran Fargate, masukkan perintah ini:

```
cat prometheus-eks-fargate.yaml \
| sed "s/{{cluster_name}}/MyCluster/;s/{{region_name}}/region/" \
| kubectl apply -f -
```

   Untuk sebuah klaster Kubernetes, masukkan perintah ini.

```
cat prometheus-k8s.yaml \
| sed "s/{{cluster_name}}/MyCluster/;s/{{region_name}}/region/" \
```

```
| kubectl apply -f -
```

Menampilkan metrik Prometheus Redis Anda

Tutorial ini mengirimkan metrik berikut ke namespace ContainerInsights/Prometheus di. CloudWatch Anda dapat menggunakan CloudWatch konsol untuk melihat metrik di namespace tersebut.

Nama metrik	Dimensi	
redis_net_input_bytes_total	ClusterName, Namespace	
redis_net_output_bytes_total	ClusterName, Namespace	
redis_expired_keys_total	ClusterName, Namespace	
redis_evicted_keys_total	ClusterName, Namespace	
redis_keyspace_hits_total	ClusterName, Namespace	
redis_keyspace_misses_total	ClusterName, Namespace	
redis_memory_used_bytes	ClusterName, Namespace	
redis_connected_clients	ClusterName, Namespace	

Nama metrik	Dimensi	
`redis_commands_total`	ClusterName, `Namespace` , cmd	
`redis_db_keys`	ClusterName, `Namespace` , db	

> **Note**
>
> Nilai dari dimensi cmd bisa berupa: `append`, `client`, `command`, `config`, `dbsize`, `flushall`, `get`, `incr`, `info`, `latency`, atau `slowlog`.
> Nilai dari dimensi db bisa `db0` hingga `db15`.

Anda juga dapat membuat CloudWatch dasbor untuk metrik Redis Prometheus Anda.

Cara membuat sebuah dasbor untuk metrik-metrik Redis Prometheus

1. Buat variabel lingkungan, yang menggantikan nilai di bawah ini untuk menyesuaikan dengan deployment Anda.

```
DASHBOARD_NAME=your_cw_dashboard_name
REGION_NAME=your_metric_region_such_as_us-east-1
CLUSTER_NAME=your_k8s_cluster_name_here
NAMESPACE=your_redis_service_namespace_here
```

2. Masukkan perintah berikut untuk membuat dasbor tersebut.

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-
prometheus/sample_cloudwatch_dashboards/redis/cw_dashboard_redis.json \
| sed "s/{{YOUR_AWS_REGION}}/${REGION_NAME}/g" \
| sed "s/{{YOUR_CLUSTER_NAME}}/${CLUSTER_NAME}/g" \
| sed "s/{{YOUR_NAMESPACE}}/${NAMESPACE}/g" \
```

Konversi tipe metrik Prometheus oleh Agen CloudWatch

Pustaka klien Prometheus menawarkan empat tipe metrik inti:

- Penghitung
- Pengukur
- Ringkasan
- Histogram

CloudWatch Agen mendukung jenis metrik penghitung, pengukur, dan ringkasan. Dukungan untuk metrik-metrik histogram direncanakan untuk rilis mendatang.

Metrik Prometheus dengan tipe metrik histogram yang tidak didukung dijatuhkan oleh agen. CloudWatch Untuk informasi selengkapnya, lihat Mencatat log metrik-metrik Prometheus yang dibuang.

Metrik pengukuran

Sebuah metrik pengukur Prometheus adalah sebuah metrik yang merepresentasikan nilai numerik tunggal yang dapat naik dan turun tanpa alasan. CloudWatch Agen menggores metrik pengukur dan mengirimkan nilai-nilai ini secara langsung.

Metrik penghitung

Metrik penghitung Prometheus adalah sebuah metrik kumulatif yang merepresentasikan sebuah penghitung tunggal yang terus-menerus meningkat secara monoton, di mana nilainya hanya dapat bertambah atau direset menjadi nol CloudWatchAgen menghitung delta dari scrape sebelumnya dan mengirimkan nilai delta sebagai nilai metrik dalam peristiwa log. Jadi CloudWatch agen akan mulai menghasilkan satu peristiwa log dari goresan kedua dan melanjutkan dengan goresan berikutnya, jika ada.

Metrik ringkasan

Sebuah metrik ringkasan Prometheus adalah jenis metrik kompleks yang direpresentasikan oleh beberapa titik data. Metrik ini menyediakan jumlah observasi total dan jumlah dari semua nilai-nilai yang diamati. Metrik ini menghitung kuantil yang dapat dikonfigurasi selama jendela waktu yang bergerak

Jumlah dan hitungan dari sebuah metrik ringkasan bersifat kumulatif, namun kuantil-kuantilnya tidak demikian Contoh berikut menunjukkan varian kuantil.

```
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 7.123e-06
```

```
go_gc_duration_seconds{quantile="0.25"} 9.204e-06
go_gc_duration_seconds{quantile="0.5"} 1.1065e-05
go_gc_duration_seconds{quantile="0.75"} 2.8731e-05
go_gc_duration_seconds{quantile="1"} 0.003841496
go_gc_duration_seconds_sum 0.37630427
go_gc_duration_seconds_count 9774
```

CloudWatch Agen menangani jumlah dan hitungan metrik ringkasan dengan cara yang sama seperti menangani metrik penghitung, seperti yang dijelaskan di bagian sebelumnya. CloudWatch Agen mempertahankan nilai kuantil seperti yang dilaporkan semula.

Metrik Prometheus dikumpulkan oleh agen CloudWatch

CloudWatch Agen dengan dukungan Prometheus secara otomatis mengumpulkan metrik dari beberapa layanan dan beban kerja. Metrik-metrik yang dikumpulkan secara bawaan dicantumkan dalam bagian-bagian berikut. Anda juga dapat mengonfigurasi agen untuk mengumpulkan lebih banyak metrik dari layanan-layanan ini, dan untuk mengumpulkan metrik-metrik Prometheus dari aplikasi dan layanan lain. Untuk informasi selengkapnya tentang pengumpulan metrik tambahan, silakan lihat CloudWatch konfigurasi agen untuk Prometheus.

Metrik Prometheus yang dikumpulkan dari kluster Amazon EKS dan Kubernetes ada di namespace / Prometheus. ContainerInsights Metrik Prometheus yang dikumpulkan dari cluster Amazon ECS ada di ruang nama ECS//Prometheus. ContainerInsights

Topik

- Metrik-metrik Prometheus untuk App Mesh
- Metrik-metrik Prometheus untuk NGINX
- Metrik-metrik Prometheus untuk Memcached
- Metrik-metrik Prometheus untuk Java/JMX
- Metrik-metrik Prometheus untuk HAProxy

Metrik-metrik Prometheus untuk App Mesh

Metrik-metrik berikut secara otomatis dikumpulkan dari App Mesh .

CloudWatch Wawasan Kontainer juga dapat mengumpulkan Log Akses Utusan App Mesh. Untuk informasi selengkapnya, lihat (Opsional) Mengaktifkan log akses App Mesh Envoy.

Metrik-metrik Prometheus untuk App Mesh di klaster Amazon EKS dan Kubernetes

Nama metrik	Dimensi
envoy_http_downstream_rq_total	ClusterName, Namespace
envoy_http_downstream_rq_xx	ClusterName, Namespace ClusterName, Namespace , envoy_http_conn_manager_prefix, envoy_response_code_class
envoy_cluster_upstream_cx_rx_bytes_total	ClusterName, Namespace
envoy_cluster_upstream_cx_tx_bytes_total	ClusterName, Namespace
envoy_cluster_membership_healthy	ClusterName, Namespace
envoy_cluster_membership_total	ClusterName, Namespace
envoy_server_memory_heap_size	ClusterName, Namespace
envoy_server_memory_allocated	ClusterName, Namespace

Nama metrik	Dimensi	
envoy_cluster_upstream_cx_connect_timeout	ClusterName, Namespace	
envoy_cluster_upstream_rq_pending_failure_eject	ClusterName, Namespace	
envoy_cluster_upstream_rq_pending_overflow	ClusterName, Namespace	
envoy_cluster_upstream_rq_timeout	ClusterName, Namespace	
envoy_cluster_upstream_rq_try_per_timeout	ClusterName, Namespace	
envoy_cluster_upstream_rq_rx_reset	ClusterName, Namespace	

Nama metrik	Dimensi	
envoy_cluster_upstream_cx_destroy_local_with_active_rq	ClusterName, Namespace	
envoy_cluster_upstream_cx_destroy_remote_active_rq	ClusterName, Namespace	
envoy_cluster_upstream_rq_maintenance_mode	ClusterName, Namespace	
envoy_cluster_upstream_flow_control_paused_reading_total	ClusterName, Namespace	
envoy_cluster_upstream_flow_control_resumed_reading_total	ClusterName, Namespace	

Nama metrik	Dimensi	
envoy_cluster_upstream_flow_control_backed_up_total	ClusterName, Namespace	
envoy_cluster_upstream_flow_control_drained_total	ClusterName, Namespace	
envoy_cluster_upstream_rq_retry	ClusterName, Namespace	
envoy_cluster_upstream_rq_retry_success	ClusterName, Namespace	
envoy_cluster_upstream_rq_retry_overflow	ClusterName, Namespace	
envoy_server_live	ClusterName, Namespace	
envoy_server_uptime	ClusterName, Namespace	

Metrik-metrik Prometheus untuk App Mesh di klaster Amazon ECS

Nama metrik	Dimensi	
envoy_http_downstream_rq_total	ClusterName, TaskDefinitionFamily	
envoy_http_downstream_rq_xx	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_cx_rx_bytes_total	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_cx_tx_bytes_total	ClusterName, TaskDefinitionFamily	
envoy_cluster_membership_healthy	ClusterName, TaskDefinitionFamily	
envoy_cluster_membership_total	ClusterName, TaskDefinitionFamily	
envoy_server_memory_heap_size	ClusterName, TaskDefinitionFamily	
envoy_server_memory_allocated	ClusterName, TaskDefinitionFamily	
envoy_cluster_upst	ClusterName, TaskDefinitionFamily	

Nama metrik	Dimensi	
ream_cx_connect_timeout		
envoy_cluster_upstream_rq_pending_failure_eject	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_rq_pending_overflow	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_rq_timeout	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_rq_try_per_timeout	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_rq_rx_reset	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_cx_destroy_local_with_active_rq	ClusterName, TaskDefinitionFamily	

Nama metrik	Dimensi	
envoy_cluster_upstream_cx_destroy_remote_active_rq	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_rq_maintenance_mode	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_flow_control_paused_reading_total	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_flow_control_resumed_reading_total	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_flow_control_backed_up_total	ClusterName, TaskDefinitionFamily	

Nama metrik	Dimensi
envoy_cluster_upstream_flow_control_drained_total	ClusterName, TaskDefinitionFamily
envoy_cluster_upstream_rq_retry	ClusterName, TaskDefinitionFamily
envoy_cluster_upstream_rq_retry_success	ClusterName, TaskDefinitionFamily
envoy_cluster_upstream_rq_retry_overflow	ClusterName, TaskDefinitionFamily
envoy_server_live	ClusterName, TaskDefinitionFamily
envoy_server_uptime	ClusterName, TaskDefinitionFamily
envoy_http_downstream_rq_xx	ClusterName,, TaskDefinitionFamily envoy_http_conn_manager_awalan, envoy_response_code_class ClusterName,, TaskDefinitionFamily envoy_response_code_class

> **Note**
>
> TaskDefinitionFamily adalah namespace Kubernetes dari mesh.

Nilai dari `envoy_http_conn_manager_prefix` dapat berupa `ingress`, `egress`, atau `admin`.
Nilai dari `envoy_response_code_class` dapat berupa 1 (singkatan untuk `1xx`), 2 singkatan untuk `2xx`), 3 singkatan untuk `3xx`), 4 singkatan untuk `4xx`), atau 5 singkatan untuk `5xx`).

Metrik-metrik Prometheus untuk NGINX

Metrik-metrik berikut secara otomatis akan dikumpulkan dari NGINX pada klaster Amazon EKS dan Kubernetes.

Nama metrik	Dimensi	
`nginx_ingress_controller_nginx_process_cpu_seconds_total`	ClusterName, `Namespace` , Layanan	
`nginx_ingress_controller_success`	ClusterName, `Namespace` , Layanan	
`nginx_ingress_controller_requests`	ClusterName, `Namespace` , Layanan	
`nginx_ingress_controller_nginx_process_connections`	ClusterName, `Namespace` , Layanan	
`nginx_ingress_cont`	ClusterName, `Namespace` , Layanan	

Nama metrik	Dimensi	
roller_nginx_process_connections_total		
nginx_ingress_controller_nginx_process_resident_memory_bytes	ClusterName, Namespace , Layanan	
nginx_ingress_controller_config_last_reload_successful	ClusterName, Namespace , Layanan	
nginx_ingress_controller_requests	ClusterName,Namespace , Layanan, status	

Metrik-metrik Prometheus untuk Memcached

Metrik-metrik berikut secara otomatis akan dikumpulkan dari Memcached pada klaster Amazon EKS dan Kubernetes.

Nama metrik	Dimensi	
memcached_current_items	ClusterName, Namespace , Layanan	

Nama metrik	Dimensi
memcached_current_connections	ClusterName, Namespace , Layanan
memcached_limit_bytes	ClusterName, Namespace , Layanan
memcached_current_bytes	ClusterName, Namespace , Layanan
memcached_written_bytes_total	ClusterName, Namespace , Layanan
memcached_read_bytes_total	ClusterName, Namespace , Layanan
memcached_items_evicted_total	ClusterName, Namespace , Layanan
memcached_items_reclaimed_total	ClusterName, Namespace , Layanan
memcached_commands_total	ClusterName, Namespace , Layanan ClusterName,Namespace , Layanan, perintah ClusterName,Namespace , Layanan, status, perintah

Metrik-metrik Prometheus untuk Java/JMX

Metrik-metrik yang dikumpulkan di klaster Amazon EKS dan Kubernetes

Di klaster Amazon EKS dan Kubernetes, Wawasan Kontainer dapat mengumpulkan metrik-metrik Prometheus yang telah ditentukan sebelumnya untuk yang berikutnya dari Java Virtual Machine (JVM), Java, dan Tomcat (Catalina) dengan menggunakan JMX Exporter. Untuk informasi selengkapnya, silakan lihat prometheus/jmx_exporter di Github.

Java/JMX di klaster Amazon EKS dan Kubernetes

Nama metrik	Dimensi	
jvm_classes_loaded	ClusterName , Namespace	
jvm_threads_current	ClusterName , Namespace	
jvm_threads_daemon	ClusterName , Namespace	
java_lang_operatingsystem_totalswapspacesize	ClusterName , Namespace	
java_lang_operatingsystem_systemcpuload	ClusterName , Namespace	
java_lang_operatingsystem_processcpuload	ClusterName , Namespace	
java_lang_operatingsystem_f	ClusterName , Namespace	

Nama metrik	Dimensi	
reeswapsp acesize		
java_lang _operatin gsystem_t otalphysi calmemorysize	ClusterName , Namespace	
java_lang _operatin gsystem_f reephysic almemorysize	ClusterName , Namespace	
java_lang _operatin gsystem_o penfilede scriptorcount	ClusterName , Namespace	
java_lang _operatin gsystem_a vailablep rocessors	ClusterName , Namespace	
jvm_memor y_bytes_used	ClusterName , Namespace , bidang	
jvm_memor y_pool_by tes_used	ClusterName , Namespace , kolam	

> **Note**
>
> Nilai dari dimensi `area` bisa `heap` atau `nonheap`.
> Nilai dari `pool` dimensi bisa `Tenured Gen`, `Compress Class Space`, `Survivor Space`, `Eden Space`, `Code Cache`, atau `Metaspace`.

Tomcat/JMX di klaster Amazon EKS dan Kubernetes

Selain metrik-metrik Java/JMX di tabel sebelumnya, metrik-metrik berikut juga dikumpulkan untuk beban kerja Tomcat.

Nama metrik	Dimensi	
`catalina_manager_activesessions`	`ClusterName` , `Namespace`	
`catalina_manager_rejectedsessions`	`ClusterName` , `Namespace`	
`catalina_globalrequestprocessor_bytesreceived`	`ClusterName` , `Namespace`	
`catalina_globalrequestprocessor_bytessent`	`ClusterName` , `Namespace`	
`catalina_globalrequestproce`	`ClusterName` , `Namespace`	

Nama metrik	Dimensi	
ssor_requestcount		
catalina_globalrequestprocessor_errorcount	ClusterName , Namespace	
catalina_globalrequestprocessor_processingtime	ClusterName , Namespace	

Java/JMX di klaster Amazon ECS

Nama metrik	Dimensi	
jvm_classes_loaded	ClusterName , TaskDefinitionFamily	
jvm_threads_current	ClusterName , TaskDefinitionFamily	
jvm_threads_daemon	ClusterName , TaskDefinitionFamily	
java_lang_operatingsystem_totalswapspacesize	ClusterName , TaskDefinitionFamily	
java_lang_operatin	ClusterName , TaskDefinitionFamily	

Nama metrik	Dimensi	
gsystem_systemcpuload		
java_lang_operatingsystem_processcpuload	ClusterName , TaskDefinitionFamily	
java_lang_operatingsystem_freeswapspacesize	ClusterName , TaskDefinitionFamily	
java_lang_operatingsystem_totalphysicalmemorysize	ClusterName , TaskDefinitionFamily	
java_lang_operatingsystem_freephysicalmemorysize	ClusterName , TaskDefinitionFamily	
java_lang_operatingsystem_openfiledescriptorcount	ClusterName , TaskDefinitionFamily	

Nama metrik	Dimensi	
java_lang_operatingsystem_availableprocessors	ClusterName , TaskDefinitionFamily	
jvm_memory_bytes_used	ClusterName , TaskDefinitionFamily, daerah	
jvm_memory_pool_bytes_used	ClusterName , TaskDefinitionFamily, kolam	

> **Note**
>
> Nilai dari dimensi `area` bisa `heap` atau `nonheap`.
> Nilai dari dimensi `pool` bisa `Tenured Gen`, `Compress Class Space`, `Survivor Space`, `Eden Space`, `Code Cache`, atau `Metaspace`.

Tomcat/JMX di klaster Amazon ECS

Selain metrik-metrik Java/JMX di tabel sebelumnya, metrik-metrik berikut juga dikumpulkan untuk beban kerja Tomcat di klaster Amazon ECS.

Nama metrik	Dimensi	
catalina_manager_activesessions	ClusterName , TaskDefinitionFamily	
catalina_manager_rejectedsessions	ClusterName , TaskDefinitionFamily	

Nama metrik	Dimensi	
catalina_globalrequestprocessor_bytesreceived	ClusterName , TaskDefinitionFamily	
catalina_globalrequestprocessor_bytessent	ClusterName , TaskDefinitionFamily	
catalina_globalrequestprocessor_requestcount	ClusterName , TaskDefinitionFamily	
catalina_globalrequestprocessor_errorcount	ClusterName , TaskDefinitionFamily	
catalina_globalrequestprocessor_processingtime	ClusterName , TaskDefinitionFamily	

Metrik-metrik Prometheus untuk HAProxy

Metrik-metrik berikut secara otomatis akan dikumpulkan dari HAProxy pada klaster Amazon EKS dan Kubernetes.

Metrik-metrik yang dikumpulkan tergantung pada versi HAProxy Ingress yang sedang Anda gunakan. Untuk informasi selengkapnya tentang HAProxy Ingress dan versinya, silakan lihat haproxy-ingress.

Nama metrik	Dimensi	Ketersediaan
haproxy_backend_bytes_in_total	ClusterName , Namespace , Layanan	Semua versi HAProxy Ingress
haproxy_backend_bytes_out_total	ClusterName , Namespace , Layanan	Semua versi HAProxy Ingress
haproxy_backend_connection_errors_total	ClusterName , Namespace , Layanan	Semua versi HAProxy Ingress
haproxy_backend_connections_total	ClusterName , Namespace , Layanan	Semua versi HAProxy Ingress
haproxy_backend_current_sessions	ClusterName , Namespace , Layanan	Semua versi HAProxy Ingress
haproxy_backend_http_responses_total	ClusterName , Namespace , Layanan, kode, backend	Semua versi HAProxy Ingress
haproxy_backend_status	ClusterName , Namespace , Layanan	Hanya di HAProxy Ingress versi 0.10 atau yang lebih baru
haproxy_backend_up	ClusterName , Namespace , Layanan	Hanya di HAProxy Ingress versi 0.10 atau yang lebih lama

Nama metrik	Dimensi	Ketersediaan
haproxy_frontend_bytes_in_total	ClusterName , Namespace , Layanan	Semua versi HAProxy Ingress
haproxy_frontend_bytes_out_total	ClusterName , Namespace , Layanan	Semua versi HAProxy Ingress
haproxy_frontend_connections_total	ClusterName , Namespace , Layanan	Semua versi HAProxy Ingress
haproxy_frontend_current_sessions	ClusterName , Namespace , Layanan	Semua versi HAProxy Ingress
haproxy_frontend_http_requests_total	ClusterName , Namespace , Layanan	Semua versi HAProxy Ingress
haproxy_frontend_http_responses_total	ClusterName , Namespace , Layanan, kode, frontend	Semua versi HAProxy Ingress
haproxy_frontend_request_errors_total	ClusterName , Namespace , Layanan	Semua versi HAProxy Ingress

Nama metrik	Dimensi	Ketersediaan
`haproxy_frontend_requests_denied_total`	`ClusterName` , `Namespace` , Layanan	Semua versi HAProxy Ingress

> **Note**
>
> Nilai dari dimensi `code` bisa `1xx`, `2xx`, `3xx`, `4xx`, `5xx`, atau `other`.
> Nilai dari dimensi `backend` bisa:
>
> - `http-default-backend`, `http-shared-backend`, atau `httpsback-shared-backend` untuk HAProxy Ingress versi 0.0.27 atau versi yang lebih lama.
> - `_default_backend` untuk HAProxy Ingress versi yang lebih baru dari 0.0.27.
>
> Nilai dari dimensi `frontend` bisa:
>
> - `httpfront-default-backend`, `httpfront-shared-frontend`, atau `httpfronts` untuk HAProxy Ingress versi 0.0.27 atau versi yang lebih lama.
> - `_front_http` atau `_front_https` untuk HAProxy Ingress dengan versi yang lebih baru dari 0.0.27.

Menampilkan metrik-metrik Prometheus Anda

Anda dapat memantau dan mengatur alarm untuk semua metrik Prometheus Anda termasuk metrik-metrik yang telah dikurasi dan diagregasi sebelumnya dari App Mesh, NGINX, Java/JMX, Memcached, dan HAProxy, serta setiap exporter Prometheus lain yang mungkin telah Anda tambahkan secara manual. Untuk informasi selengkapnya tentang cara mengumpulkan metrik dari pengekspor Prometheus lainnya, silakan lihat Tutorial untuk menambahkan sebuah target scraping Prometheus baru: metrik Server API Prometheus.

Di CloudWatch konsol, Container Insights menyediakan laporan pra-bangun berikut:

- Untuk klaster Amazon EKS dan Kubernetes, ada laporan-laporan yang telah dibuat sebelumnya untuk App Mesh, NGINX, HAPROXY, Memcached, dan Java/JMX.

- Untuk klaster Amazon ECS, ada laporan-laporan yang telah dibuat sebelumnya untuk App Mesh dan Java/JMX.

Wawasan Kontainer juga menyediakan dasbor yang disesuaikan untuk masing-masing beban kerja yang digunakan oleh Wawasan Kontainer untuk mengumpulkan metrik-metrik terkurasi. Anda dapat mengunduh dasbor ini dari GitHub

Cara melihat semua metrik Prometheus Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik.
3. Dalam daftar ruang nama, pilih ContainerInsights/Prometheus atau ECS//Prometheus. ContainerInsights
4. Pilih salah satu set dimensi dalam daftar berikut. Kemudian pilih kotak centang yang ada di samping metrik yang ingin Anda lihat.

Cara melihat laporan yang telah dibuat sebelumnya tentang metrik-metrik Prometheus Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Pemantauan Performa.
3. Di kotak menu geser-turun yang ada di sekitar bagian atas halaman, pilih opsi Prometheus mana pun.

   Di kotak menu geser-turun lainnya, pilih sebuah klaster yang hendak dilihat

Kami juga telah menyediakan dasbor kustom untuk NGINX, App Mesh, Memcached, HAProxy, dan Java/JMX.

Cara menggunakan dasbor kustom yang disediakan Amazon

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Dasbor.
3. Pilih Buat Dasbor. Masukkan nama untuk dasbor baru tersebut, dan pilih Buat dasbor.
4. Di Tambahkan ke dasbor ini, pilih Batalkan.
5. Pilih Tindakan, Tampilkan/sunting sumber.

6. Unduh salah satu file JSON berikut:

   - Sumber dasbor kustom NGINX di Github.
   - Sumber dasbor kustom App Mesh di Github.
   - Sumber dasbor kustom Memcached di Github
   - Sumber dasbor kustom HAProxy-Ingress di Github
   - Sumber dasbor kustom Java/JMX di Github.

7. Buka file JSON yang Anda unduh dengan editor teks, dan lakukan perubahan-perubahan berikut:

   - Ganti semua string `{{YOUR_CLUSTER_NAME}}` dengan nama persis seperti nama klaster Anda. Anda harus memastikan untuk tidak menambahkan spasi putih sebelum atau setelah teks.
   - Ganti semua `{{YOUR_REGION}}` string dengan AWS Region tempat cluster Anda berjalan. Sebagai contoh, **`us-west-1`** Anda harus memastikan untuk tidak menambahkan spasi putih sebelum atau setelah teks.
   - Ganti semua string `{{YOUR_NAMESPACE}}` dengan nama persis seperti nama beban kerja Anda.
   - Ganti semua string `{{YOUR_SERVICE_NAME}}` dengan nama persis seperti nama layanan beban kerja Anda. Sebagai contoh, **`haproxy-haproxy-ingress-controller-metrics`**.

8. Salin seluruh gumpalan JSON dan tempel ke kotak teks di CloudWatch konsol, ganti apa yang sudah ada di dalam kotak.

9. Pilih Perbarui, Simpan dasbor.

Pemecahan masalah metrik-metrik Prometheus

Bagian ini akan memberikan Anda bantuan untuk melakukan pemecahan masalah konfigurasi untuk penyiapan metrik-metrik Prometheus Anda.

Topik

- Pemecahan masalah untuk metri-metrik Prometheus di Amazon ECS
- Pemecahan masalah terkait metrik-metrik Prometheus di klaster Amazon EKS dan Kubernetes

Pemecahan masalah untuk metri-metrik Prometheus di Amazon ECS

Bagian ini akan memberikan Anda bantuan untuk melakukan pemecahan masalah konfigurasi untuk penyiapan metrik-metrik Prometheus Anda di klaster Amazon ECS.

Saya tidak melihat metrik Prometheus dikirim ke Log CloudWatch

Metrik-metrik Prometheus harus diserap sebagai peristiwa log dalam grup log /aws/ecs/containerinsights/cluster-name/Prometheus. Jika grup log tidak dibuat atau metrik Prometheus tidak dikirim ke grup log, Anda harus terlebih dahulu memeriksa apakah target Prometheus telah berhasil ditemukan oleh agen. CloudWatch Dan selanjutnya periksa grup keamanan dan pengaturan izin CloudWatch agen. Langkah-langkah berikut akan memandu Anda melakukan debug.

Langkah 1: Aktifkan CloudWatch mode debugging agen

Pertama, ubah CloudWatch agen ke mode debug dengan menambahkan baris tebal berikut ke file AWS CloudFormation template Anda, `cwagent-ecs-prometheus-metric-for-bridge-host.yaml` atau`cwagent-ecs-prometheus-metric-for-awsvpc.yaml`. Kemudian simpan filenya.

```
cwagentconfig.json: |
    {
      "agent": {
        "debug": true
      },
      "logs": {
        "metrics_collected": {
```

Buat AWS CloudFormation set perubahan baru terhadap tumpukan yang ada. Setel parameter lain di changeset ke nilai yang sama seperti di tumpukan yang ada AWS CloudFormation . Contoh berikut adalah untuk CloudWatch agen yang dipasang di cluster Amazon ECS menggunakan tipe peluncuran EC2 dan mode jaringan jembatan.

```
ECS_NETWORK_MODE=bridge
 CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name
NEW_CHANGESET_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-change-set --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
```

```
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSNetworkMode,ParameterValue=$ECS_NETWORK_MODE \
                 ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=$ECS_EXECUTION_ROLE_NAME
 \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION \
    --change-set-name $NEW_CHANGESET_NAME
```

Buka AWS CloudFormation konsol untuk meninjau set perubahan baru,. `$NEW_CHANGESET_NAME` Harus ada satu perubahan yang diterapkan pada sumber daya CW AgentConfig SSMParameter. Jalankan changeset dan restart tugas CloudWatch agen dengan memasukkan perintah berikut.

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 0 \
--service your_service_name_here \
--region $AWS_REGION
```

Tunggu sekitar 10 detik dan kemudian masukkan perintah berikut.

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 1 \
--service your_service_name_here \
--region $AWS_REGION
```

Langkah 2: Memeriksa log penemuan layanan ECS

Definisi tugas ECS dari CloudWatch agen memungkinkan log secara default di bagian di bawah ini. Log dikirim ke CloudWatch Log di grup log ecs-cwagent-prometheus/ecs/.

```
LogConfiguration:
  LogDriver: awslogs
    Options:
      awslogs-create-group: 'True'
      awslogs-group: "/ecs/ecs-cwagent-prometheus"
      awslogs-region: !Ref AWS::Region
      awslogs-stream-prefix: !Sub 'ecs-${ECSLaunchType}-awsvpc'
```

Lakukan penyaringan log berdasarkan string `ECS_SD_Stats` untuk mendapatkan metrik-metrik terkait penemuan layanan ECS, seperti yang ditunjukkan dalam contoh berikut.

```
2020-09-1T01:53:14Z D! ECS_SD_Stats: AWSCLI_DescribeContainerInstances: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: AWSCLI_DescribeInstancesRequest: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: AWSCLI_DescribeTaskDefinition: 2
2020-09-1T01:53:14Z D! ECS_SD_Stats: AWSCLI_DescribeTasks: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: AWSCLI_ListTasks: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: Exporter_DiscoveredTargetCount: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: LRUCache_Get_EC2MetaData: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: LRUCache_Get_TaskDefinition: 2
2020-09-1T01:53:14Z D! ECS_SD_Stats: LRUCache_Size_ContainerInstance: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: LRUCache_Size_TaskDefinition: 2
2020-09-1T01:53:14Z D! ECS_SD_Stats: Latency: 43.399783ms
```

Arti dari masing-masing metrik untuk siklus penemuan layanan ECS tertentu adalah sebagai berikut:

- AWSCLI_DescribeContainerInstances— jumlah panggilan `ECS::DescribeContainerInstances` API yang dilakukan.
- AWSCLI_DescribeInstancesRequest— jumlah panggilan `ECS::DescribeInstancesRequest` API yang dilakukan.
- AWSCLI_DescribeTaskDefinition— jumlah panggilan `ECS::DescribeTaskDefinition` API yang dilakukan.
- AWSCLI_DescribeTasks— jumlah panggilan `ECS::DescribeTasks` API yang dilakukan.
- AWSCLI_ListTasks— jumlah panggilan `ECS::ListTasks` API yang dilakukan.
- ExporterDiscoveredTargetCount— jumlah target Prometheus yang ditemukan dan berhasil diekspor ke file hasil target di dalam wadah.
- Lrucache_get_EC2 MetaData — berapa kali metadata instance kontainer diambil dari cache.
- Lrucache_get_ TaskDefinition — berapa kali metadata definisi tugas ECS diambil dari cache.
- Lrucache_size_ ContainerInstance — jumlah metadata instance kontainer unik yang di-cache dalam memori.
- Lrucache_size_ TaskDefinition — jumlah definisi tugas ECS unik yang di-cache dalam memori.
- Latensi – berapa lama siklus penemuan layanan berlangsung.

Periksa nilai `ExporterDiscoveredTargetCount` untuk melihat apakah target-target Prometheus yang ditemukan sesuai dengan harapan Anda. Jika tidak, kemungkinan penyebabnya adalah sebagai berikut:

- Konfigurasi penemuan layanan ECS mungkin tidak sesuai dengan pengaturan aplikasi Anda. Untuk penemuan layanan berbasis label docker, kontainer target Anda mungkin tidak memiliki label docker yang diperlukan yang dikonfigurasi di agen CloudWatch untuk menemukannya secara otomatis. Untuk penemuan layanan berbasis ekspresi reguler ARN definisi tugas ECS, pengaturan regex di agen mungkin tidak cocok CloudWatch dengan definisi tugas aplikasi Anda.
- Peran tugas ECS CloudWatch agen mungkin tidak memiliki izin untuk mengambil metadata tugas ECS. Periksa apakah CloudWatch agen telah diberikan izin hanya-baca berikut:
  - `ec2:DescribeInstances`
  - `ecs:ListTasks`
  - `ecs:DescribeContainerInstances`
  - `ecs:DescribeTasks`
  - `ecs:DescribeTaskDefinition`

Langkah 3: Memeriksa koneksi jaringan dan kebijakan peran tugas ECS

Jika masih belum ada peristiwa log yang dikirim ke grup CloudWatch log Log target meskipun nilai `Exporter_DiscoveredTargetCount` menunjukkan bahwa ada target Prometheus yang ditemukan, ini dapat disebabkan oleh salah satu hal berikut:

- CloudWatch Agen mungkin tidak dapat terhubung ke port target Prometheus. Periksa pengaturan grup keamanan di belakang CloudWatch agen. IP pribadi harus memungkinkan CloudWatch agen untuk terhubung ke port eksportir Prometheus.
- Peran tugas ECS CloudWatch agen mungkin tidak memiliki kebijakan yang CloudWatchAgentServerPolicydikelola. Peran tugas ECS CloudWatch agen harus memiliki kebijakan ini untuk dapat mengirim metrik Prometheus sebagai peristiwa log. Jika Anda menggunakan AWS CloudFormation template sampel untuk membuat peran IAM secara otomatis, peran tugas ECS dan peran eksekusi ECS diberikan dengan hak istimewa paling sedikit untuk melakukan pemantauan Prometheus.

Pemecahan masalah terkait metrik-metrik Prometheus di klaster Amazon EKS dan Kubernetes

Bagian ini akan menyediakan Anda bantuan untuk melakukan pemecahan masalah konfigurasi metrik-metrik Prometheus Anda di klaster Amazon EKS dan Kubernetes.

Langkah-langkah pemecahan masalah umum di Amazon EKS

Untuk mengonfirmasi bahwa CloudWatch agen sedang berjalan, masukkan perintah berikut.

```
kubectl get pod -n amazon-cloudwatch
```

Outputnya harus menyertakan sebuah baris dengan `cwagent-prometheus-id` pada kolom `NAME` dan `Running` dalam `STATUS column.`

Untuk menampilkan detail tentang pod yang sedang berjalan, Anda harus memasukkan perintah berikut. Ganti *pod-name* dengan nama lengkap pod Anda yang namanya diawali dengan `cw-agent-prometheus`.

```
kubectl describe pod pod-name -n amazon-cloudwatch
```

Jika Anda telah menginstal CloudWatch Wawasan Kontainer, Anda dapat menggunakan Wawasan CloudWatch Log untuk menanyakan log dari CloudWatch agen yang mengumpulkan metrik Prometheus.

Cara menjalankan kueri terhadap log aplikasi

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Wawasan CloudWatch Log.
3. Pilih grup log untuk log aplikasi tersebut, /aws/containerinsights/***cluster-name***/application
4. Ganti ekspresi kueri pencarian dengan kueri berikut, dan kemudian pilih Jalankan kueri

```
fields ispresent(kubernetes.pod_name) as haskubernetes_pod_name, stream,
 kubernetes.pod_name, log |
filter haskubernetes_pod_name and kubernetes.pod_name like /cwagent-prometheus
```

Anda juga dapat mengonfirmasi bahwa metrik dan metadata Prometheus sedang dicerna sebagai peristiwa Log. CloudWatch

Cara mengonfirmasi bahwa data Prometheus sedang diserap

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Wawasan CloudWatch Log.
3. Kemudian pilih /aws/containerinsights/***cluster-name***/prometheus
4. Ganti ekspresi kueri pencarian dengan kueri berikut, dan kemudian pilih Jalankan kueri

```
fields @timestamp, @message | sort @timestamp desc | limit 20
```

Mencatat log metrik-metrik Prometheus yang dibuang

Rilis ini tidak akan mengumpulkan metrik-metrik Prometheus yang bertipe histogram. Anda dapat menggunakan CloudWatch agen untuk memeriksa apakah ada metrik Prometheus yang dijatuhkan karena merupakan metrik histogram. Anda juga dapat mencatat daftar 500 metrik Prometheus pertama yang dijatuhkan dan tidak dikirim CloudWatch karena merupakan metrik histogram.

Untuk melihat apakah ada metrik yang dibuang, masukkan perintah berikut:

```
kubectl logs -l "app=cwagent-prometheus" -n amazon-cloudwatch --tail=-1
```

Jika ada metrik yang dibuang, maka Anda akan melihat garis-garis berikut di file `/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log`.

```
I! Drop Prometheus metrics with unsupported types. Only Gauge, Counter and Summary are
 supported.
I! Please enable CWAgent debug mode to view the first 500 dropped metrics
```

Jika Anda melihat garis-garis tersebut dan ingin mengetahui metrik apa yang dibuang, lakukan langkah-langkah berikut ini.

Cara mencatat log daftar metrik Prometheus yang dibuang

1. Ubah CloudWatch agen ke mode debug dengan menambahkan baris tebal berikut ke `prometheus-k8s.yaml` file `prometheus-eks.yaml` atau Anda, dan simpan file.

   ```
   {
         "agent": {
           "debug": true
         },
   ```

   Bagian file ini akan terlihat seperti berikut ini:

   ```
   cwagentconfig.json: |
       {
         "agent": {
   ```

```
        "debug": true
      },
      "logs": {
        "metrics_collected": {
```

2. Instal ulang CloudWatch agen untuk mengaktifkan mode debug dengan memasukkan perintah berikut:

```
kubectl delete deployment cwagent-prometheus -n amazon-cloudwatch
kubectl apply -f prometheus.yaml
```

Metrik yang dijatuhkan dicatat di pod CloudWatch agen.

3. Untuk mengambil log dari pod CloudWatch agen, masukkan perintah berikut:

```
kubectl logs -l "app=cwagent-prometheus" -n amazon-cloudwatch --tail=-1
```

*Atau, jika Anda menginstal logging Container Insights Fluentd, log juga disimpan di grup CloudWatch log Logs /aws/containerinsights/ cluster_name **/application**.*

Untuk menjalankan kueri pada log ini, Anda dapat mengikuti langkah-langkah untuk menjalankan kueri terhadap log aplikasi di Langkah-langkah pemecahan masalah umum di Amazon EKS.

Di mana metrik Prometheus dicerna sebagai peristiwa log Log? CloudWatch

CloudWatch Agen membuat aliran log untuk setiap konfigurasi pekerjaan scrape Prometheus. Sebagai contoh, di dalam file `prometheus-eks.yaml` dan file `prometheus-k8s.yaml`, garis `job_name: 'kubernetes-pod-appmesh-envoy'` melakukan scraping terhadap metrik-metrik App Mesh. Target Prometheus ditentukan sebagai `kubernetes-pod-appmesh-envoy`. Jadi semua metrik App Mesh Prometheus dicerna CloudWatch sebagai peristiwa Log di aliran log di bawah grup log bernama /AWS/ContainerInsights/Cluster-Name/Prometheus. kubernetes-pod-appmesh-envoy

Saya tidak melihat metrik Amazon EKS atau Kubernetes Prometheus dalam metrik CloudWatch

Pertama, Anda harus memastikan bahwa metrik-metrik Prometheus diserap sebagai peristiwa log dalam grup log /aws/containerinsights/cluster-name/Prometheus. Gunakan informasi dalam Di mana metrik Prometheus dicerna sebagai peristiwa log Log? CloudWatch untuk membantu Anda

memeriksa log stream target. Jika log stream tidak dibuat atau tidak ada peristiwa log baru di log stream, periksa hal berikut:

- Periksa apakah titik akhir pengekspor metrik Prometheus sudah diatur dengan benar
- Periksa apakah konfigurasi pengikisan Prometheus di `config map: cwagent-prometheus` bagian file YAMAL agen sudah benar. CloudWatch Konfigurasinya harus sama dengan konfigurasi yang akan digunakan dalam file konfigurasi Prometheus. Untuk informasi selengkapnya, silakan lihat <scrape_config> dalam dokumentasi Prometheus.

Jika metrik Prometheus dicerna sebagai peristiwa log dengan benar, periksa apakah setelan format metrik yang disematkan ditambahkan ke dalam peristiwa log untuk menghasilkan metrik. CloudWatch

```
"CloudWatchMetrics":[
   {
      "Metrics":[
         {
            "Name":"envoy_http_downstream_cx_destroy_remote_active_rq"
         }
      ],
      "Dimensions":[
         [
            "ClusterName",
            "Namespace"
         ]
      ],
      "Namespace":"ContainerInsights/Prometheus"
   }
],
```

Untuk informasi selengkapnya tentang format metrik tersemat, silakan lihat Spesifikasi: Format metrik tersemat .

Jika tidak ada format metrik yang disematkan dalam peristiwa log, periksa apakah `metric_declaration` bagian tersebut dikonfigurasi dengan benar di `config map: prometheus-cwagentconfig` bagian file YAMAL instalasi CloudWatch agen. Untuk informasi selengkapnya, lihat Tutorial untuk menambahkan sebuah target scraping Prometheus baru: metrik Server API Prometheus.

Integrasi dengan Wawasan Aplikasi

Amazon CloudWatch Application Insights membantu Anda memantau aplikasi dan mengidentifikasi serta menyiapkan metrik utama, log, dan alarm di seluruh sumber daya aplikasi dan tumpukan teknologi Anda. Untuk informasi selengkapnya, lihat Wawasan CloudWatch Aplikasi Amazon.

Anda dapat mengaktifkan Wawasan Aplikasi untuk mengumpulkan data tambahan dari aplikasi-aplikasi dan layanan mikro terkontainer Anda. Jika Anda belum melakukannya, maka Anda dapat mengaktifkannya dengan memilih Konfigurasi Otomatis Wawasan Aplikasi yang ada di bawah tampilan performa di dasbor Wawasan Kontainer.

Jika Anda telah menyiapkan CloudWatch Application Insights untuk memantau aplikasi kontainer Anda, dasbor Application Insights akan muncul di bawah dasbor Container Insights.

Untuk informasi selengkapnya tentang Wawasan Aplikasi dan aplikasi terkontainer, silakan lihat Aktifkan Wawasan Aplikasi untuk melakukan pemantauan sumber daya Amazon ECS dan Amazon EKS.

Melihat peristiwa siklus hidup Amazon ECS dalam Wawasan Kontainer

Anda dapat melihat peristiwa siklus hidup Amazon ECS dalam konsol Wawasan Kontainer. Hal ini akan membantu Anda dalam mengkorelasikan metrik-metrik kontainer, log, dan peristiwa dalam satu tampilan untuk memberikan Anda visibilitas operasional yang lebih lengkap.

Peristiwa-peristiwa itu termasuk peristiwa perubahan status instans kontainer, peristiwa perubahan status tugas, dan peristiwa-peristiwa tindakan layanan. Mereka secara otomatis dikirim oleh Amazon ECS ke Amazon EventBridge dan juga dikumpulkan CloudWatch dalam format log peristiwa. Untuk informasi selengkapnya tentang peristiwa-peristiwa ini, silakan lihat peristiwa-peristiwa Amazon ECS.

Harga Standard Container Insights berlaku untuk peristiwa Siklus Hidup Amazon ECS. Untuk informasi lebih lanjut, lihat Amazon CloudWatch Harga.

Untuk mengonfigurasi tabel peristiwa siklus hidup dan membuat aturan untuk sebuah klaster, Anda harus memiliki izin `events:PutRule`, `events:PutTargets`, dan `logs:CreateLogGroup`. Anda juga harus memastikan bahwa ada kebijakan sumber daya yang memungkinkan EventBridge untuk membuat aliran log dan mengirim CloudWatch log ke Log. Jika kebijakan sumber daya ini tidak ada, maka Anda dapat memasukkan perintah berikut untuk membuatnya:

```
aws --region region logs put-resource-policy --policy-name 'EventBridgeCloudWatchLogs'
 --policy-document '{
```

```
    "Statement": [
        {
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Effect": "Allow",
            "Principal": {
                "Service": ["events.amazonaws.com", "delivery.logs.amazonaws.com"]
            },
            "Resource": "arn:aws:logs:region:account-id:log-group:/aws/events/ecs/
containerinsights/*:*",
            "Sid": "TrustEventBridgeToStoreECSLifecycleLogEvents"
        }
    ],
    "Version": "2012-10-17"
}'
```

Anda juga dapat menggunakan perintah berikut untuk memeriksa apakah Anda sudah memiliki kebijakan ini, dan untuk mengonfirmasi apakah sudah dilampirkan dan berfungsi dengan benar.

```
aws logs describe-resource-policies --region region --output json
```

Untuk melihat tabel peristiwa siklus hidup, Anda harus memiliki izin `events:DescribeRule`, `events:ListTargetsByRule`, dan `logs:DescribeLogGroups`.

Untuk melihat peristiwa siklus hidup Amazon ECS di konsol Container Insights CloudWatch

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pilih Wawasan, Wawasan Kontainer.
3. Pilih Lihat dasbor kinerja.
4. Pada pilihan geser-turun berikutnya, pilih Klaster ECS, Layanan ECS, atau Tugas ECS.
5. Jika Anda memilih Layanan ECS atau Tugas ECS di langkah sebelumnya, silakan pilih tab Peristiwa siklus hidup.
6. Di bagian bawah halaman, jika Anda melihat Konfigurasi peristiwa siklus hidup, pilih untuk membuat EventBridge aturan untuk klaster Anda.

   Peristiwa-peristiwa itu akan ditampilkan di bawah panel wawasan kontainer dan di atas bagian Wawasan Aplikasi. Untuk menjalankan analitik tambahan dan membuat visualisasi tambahan

mengenai peristiwa-peristiwa ini, silakan pilih Tampilkan di Wawasan Log di tabel Peristiwa Siklus Hidup.

Pemecahan Masalah Wawasan Kontainer

Bagian-bagian berikut dapat membantu Anda jika Anda mengalami masalah yang terjadi dengan Wawasan Kontainer.

Deployment mengalami kegagalan di Amazon EKS atau Kubernetes

Jika agen tidak menerapkan deployment dengan benar di klaster Kubernetes, coba lakukan hal berikut:

- Jalankan perintah berikut untuk mendapatkan daftar pod.

```
kubectl get pods -n amazon-cloudwatch
```

- Jalankan perintah berikut dan periksa peristiwa di bagian bawah output.

```
kubectl describe pod pod-name -n amazon-cloudwatch
```

- Jalankan perintah berikut untuk memeriksa log.

```
kubectl logs pod-name -n amazon-cloudwatch
```

Kepanikan yang tidak sah: Tidak dapat mengambil data cadvisor dari kubelet

Jika deployment Anda gagal dan mengalami kesalahan `Unauthorized panic: Cannot retrieve cadvisor data from kubelet`, maka kubelet Anda mungkin tidak mengaktifkan mode otorisasi Webhook. Mode ini wajib digunakan untuk Wawasan Kontainer. Untuk informasi selengkapnya, lihat Memverifikasi prasyarat.

Menerapkan Wawasan Kontainer di sebuah klaster yang dihapus dan dibuat kembali di Amazon ECS

Jika Anda menghapus sebuah klaster Amazon ECS yang sudah ada yang tidak mengaktifkan Wawasan Kontainer, dan Anda kemudian membuat ulang klaster itu dengan nama yang sama, maka Anda tidak akan dapat mengaktifkan Wawasan Kontainer di klaster baru ini pada saat Anda membuat

ulang klaster tersebut. Anda dapat mengaktifkan Wawasan Kontainer dengan membuat ulang, dan kemudian memasukkan perintah berikut:

```
aws ecs update-cluster-settings --cluster myCICluster --settings
 name=containerInsights,value=enabled
```

Kesalahan titik akhir yang tidak valid

Jika Anda melihat sebuah pesan kesalahan yang mirip dengan pesan berikut ini, Anda harus memeriksa dan memastikan bahwa Anda mengganti semua placeholder seperti *cluster-name* dan *region-name* dalam perintah yang Anda gunakan dengan informasi yang benar untuk deployment Anda.

```
"log": "2020-04-02T08:36:16Z E! cloudwatchlogs: code: InvalidEndpointURL, message:
 invalid endpoint uri, original error: &url.Error{Op:\"parse\", URL:\"https://
logs.{{region_name}}.amazonaws.com/\", Err:\"{\"}, &awserr.baseError{code:
\"InvalidEndpointURL\", message:\"invalid endpoint uri\", errs:[]error{(*url.Error)
(0xc0008723c0)}}\n",
```

Metrik-metrik yang tidak ditampilkan di konsol

Jika Anda tidak melihat metrik Container Insights di dalamnya AWS Management Console, pastikan Anda telah menyelesaikan penyiapan Container Insights. Metrik-metrik tidak ditampilkan sebelum Wawasan Kontainer telah disiapkan sepenuhnya. Untuk informasi selengkapnya, lihat Menyiapkan Wawasan Kontainer.

Pod metrik hilang di Amazon EKS atau Kubernetes setelah melakukan upgrade pada klaster

Bagian ini mungkin berguna jika semua atau beberapa metrik pod hilang setelah Anda menerapkan CloudWatch agen sebagai daemonset pada klaster baru atau yang ditingkatkan, atau Anda melihat log kesalahan dengan pesan tersebut. `W! No pod metric collected`

Kesalahan-kesalahan ini dapat disebabkan oleh perubahan dalam runtime kontainer, seperti containerd atau docker systemd cgroup driver. Anda biasanya dapat memecahkan masalah ini dengan memperbarui manifes deployment Anda sehingga soket containerd dari host dipasang ke kontainer. Lihat contoh berikut ini:

```
# For full example see https://github.com/aws-samples/amazon-cloudwatch-container-
insights/blob/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/
container-insights-monitoring/cwagent/cwagent-daemonset.yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: cloudwatch-agent
  namespace: amazon-cloudwatch
spec:
  template:
    spec:
      containers:
        - name: cloudwatch-agent
# ...
          # Don't change the mountPath
          volumeMounts:
# ...
            - name: dockersock
              mountPath: /var/run/docker.sock
              readOnly: true
            - name: varlibdocker
              mountPath: /var/lib/docker
              readOnly: true
            - name: containerdsock # NEW mount
              mountPath: /run/containerd/containerd.sock
              readOnly: true
# ...
      volumes:
# ...
        - name: dockersock
          hostPath:
            path: /var/run/docker.sock
        - name: varlibdocker
          hostPath:
            path: /var/lib/docker
        - name: containerdsock # NEW volume
          hostPath:
            path: /run/containerd/containerd.sock
```

Tidak ada metrik pod saat menggunakan Bottlerocket untuk Amazon EKS

Bottlerocket adalah sebuah sistem operasi sumber terbuka berbasis Linux yang dibangun dengan tujuan oleh AWS untuk menjalankan kontainer.

Bottlerocket menggunakan jalur `containerd` yang berbeda pada host, sehingga Anda perlu mengubah volumenya ke lokasinya. Jika tidak, maka Anda akan melihat sebuah kesalahan di log yang mencakup `W! No pod metric collected`. Lihat contoh berikut ini.

```
volumes:
  # ...
    - name: containerdsock
      hostPath:
        # path: /run/containerd/containerd.sock
        # bottlerocket does not mount containerd sock at normal place
        # https://github.com/bottlerocket-os/bottlerocket/
commit/91810c85b83ff4c3660b496e243ef8b55df0973b
        path: /run/dockershim.sock
```

Tidak ada metrik filesystem kontainer saat menggunakan runtime containerd untuk Amazon EKS atau Kubernetes

Masalah ini adalah masalah yang diketahui dan sedang dikerjakan oleh kontributor komunitas. Untuk informasi selengkapnya, lihat Metrik penggunaan disk untuk containerd dan metrik sistem file container tidak didukung oleh cadvisor untuk containerd on. GitHub

Volume log tak terduga meningkat dari CloudWatch agen saat mengumpulkan metrik Prometheus

Ini adalah regresi yang diperkenalkan dalam versi 1.247347.6b250880 dari agen. CloudWatch Regresi ini telah diperbaiki dalam versi agen yang lebih baru. Dampaknya terbatas pada skenario di mana pelanggan mengumpulkan log CloudWatch agen itu sendiri dan juga menggunakan Prometheus. Untuk informasi lebih lanjut, lihat agen [prometheus] mencetak semua metrik yang tergores saat masuk. GitHub

Citra docker terbaru yang disebutkan dalam catatan rilis tidak ditemukan dari Dockerhub

Kami memperbarui catatan rilis dan tanda pada Github sebelum kami memulai rilis aktual secara internal. Biasanya dibutuhkan waktu 1-2 minggu untuk melihat citra docker terbaru pada registri setelah kita bertemu nomor versinya di Github. Tidak ada rilis malam untuk gambar wadah CloudWatch agen. Anda dapat membangun gambar langsung dari sumber di lokasi berikut: https://github.com/aws/ amazon-cloudwatch-agent /tree/main//amazon-cloudwatch-container-insightscloudwatch-agent-dockerfile

CrashLoopBackoff kesalahan pada CloudWatch agen

Jika Anda melihat `CrashLoopBackOff` kesalahan untuk CloudWatch agen, pastikan bahwa izin IAM Anda disetel dengan benar. Untuk informasi selengkapnya, lihat Memverifikasi prasyarat.

CloudWatch agen atau pod Fluentd terjebak dalam keadaan tertunda

Jika Anda memiliki CloudWatch agen atau pod Fluentd yang terjebak `Pending` atau mengalami `FailedScheduling` kesalahan, tentukan apakah node Anda memiliki sumber daya komputasi yang cukup berdasarkan jumlah inti dan jumlah RAM yang dibutuhkan oleh agen. Masukkan perintah berikut untuk mendeskripsikan pod:

```
kubectl describe pod cloudwatch-agent-85ppg -n amazon-cloudwatch
```

Membangun image CloudWatch agen Docker Anda sendiri

Anda dapat membuat image Docker CloudWatch agen Anda sendiri dengan merujuk ke Dockerfile yang terletak di https://github.com/aws-samples/ amazon-cloudwatch-container-insights /blob/latest/ / Dockerfile. cloudwatch-agent-dockerfile

Dockerfile tersebut mendukung pembuatan citra multi-arsitektur secara langsung dengan menggunakan `docker buildx`.

Menerapkan fitur CloudWatch agen lain di kontainer Anda

Anda dapat menerapkan fitur pemantauan tambahan dalam kontainer Anda menggunakan CloudWatch agen. Fitur-fitur ini mencakup hal-hal sebagai berikut:

- Format Metrik Tersemat— Untuk informasi selengkapnya, silakan lihat Menyematkan metrik dalam log.
- StatsD— Untuk informasi selengkapnya, silakan lihat Ambil metrik kustom dengan StatSD .

Instruksi dan file yang diperlukan terletak GitHub di lokasi berikut:

- Untuk kontainer-kontainer Amazon ECS, silakan lihat Contoh penetapan tugas Amazon ECS yang didasarkan pada mode deployment.
- Untuk kontainer Amazon EKS dan Kubernetes, silakan lihat Contoh file YAML Kubernetes yang didasarkan pada mode deployment.

Wawasan Lambda

CloudWatch Lambda Insights adalah solusi pemantauan dan pemecahan masalah untuk aplikasi tanpa server yang berjalan. AWS Lambda Solusi ini mengumpulkan, menggabungkan, dan merangkum metrik tingkat sistem termasuk waktu CPU, memori, cakram, dan jaringan. Aplikasi ini juga mengumpulkan, menggabungkan, dan merangkum informasi diagnostik seperti proses mulai yang dingin dan penonaktifan pekerja Lambda untuk membantu Anda mengisolasi masalah dengan fungsi Lambda Anda dan menyelesaikan masalahnya segera.

Lambda Insights menggunakan ekstensi CloudWatch Lambda baru, yang disediakan sebagai lapisan Lambda. Saat Anda menginstal ekstensi ini pada fungsi Lambda, ekstensi ini mengumpulkan metrik tingkat sistem dan memancarkan peristiwa log kinerja tunggal untuk setiap pemanggilan fungsi Lambda tersebut. CloudWatch menggunakan format metrik tertanam untuk mengekstrak metrik dari peristiwa log.

Untuk informasi selengkapnya tentang ekstensi Lambda, lihat Menggunakan AWS Lambda ekstensi. Untuk informasi selengkapnya tentang format metrik tersemat, silakan lihat Menyematkan metrik dalam log.

Anda dapat menggunakan Wawasan Lambda dengan fungsi Lambda yang menggunakan runtime Lanmba yang mendukung ekstensi Lambda. Untuk melihat daftar runtime, silakan lihat API Ekstensi Lambda.

Penetapan Harga

Untuk setiap fungsi Lambda yang diaktifkan untuk Wawasan Lambda, Anda hanya perlu membayar yang Anda gunakan untuk metrik dan log. Untuk contoh harga, lihat CloudWatch Harga Amazon.

Anda dikenakan biaya atas waktu pelaksanaan yang dihabiskan oleh ekstensi Lambda, dengan penambahan 1 ms. Untuk informasi selengkapnya tentang penetapan harga Lambda, silakan lihat Penetapan Harga AWS Lambda.

Memulai Wawasan Lambda

Untuk mengaktifkan Wawasan Lambda di fungsi Lambda, Anda dapat menggunakan tombol sekali-klik di konsol Lambda. Atau, Anda dapat menggunakan AWS CLI, AWS CloudFormation, AWS Serverless Application Model CLI, atau. AWS Cloud Development Kit (AWS CDK)

Bagian berikut memberikan instruksi yang terperinci untuk menyelesaikan langkah-langkah ini.

Topik

Versi yang tersedia untuk ekstensi Wawasan lambda.

Bagian ini mencantumkan versi ekstensi Lambda Insights, dan ARN yang akan digunakan untuk ekstensi ini di setiap Wilayah. AWS

Topik

platform x86-64

Bagian ini mencantumkan versi ekstensi Lambda Insights untuk platform x86-64, dan ARN yang akan digunakan untuk ekstensi ini di setiap Wilayah. AWS

1.0.295.0

Versi 1.0.295.0 mencakup pembaruan ketergantungan untuk semua runtime yang kompatibel.

ARN untuk versi 1.0.295.0

Tabel berikut mencantumkan ARN yang akan digunakan untuk versi ekstensi ini di setiap AWS Wilayah yang tersedia.

Wilayah	ARN
AS Timur (Virginia Utara)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:51
AS Timur (Ohio)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:51
AS Barat (California Utara)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:51
AS Barat (Oregon)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:51
Afrika (Cape Town)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:42
Asia Pasifik (Hong Kong)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:42
Asia Pasifik (Hyderabad)	arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension:24
Asia Pasifik (Jakarta)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension:28
Asia Pasifik (Melbourne)	arn:aws:lambda:ap-southeast-4:158895979263:layer:LambdaInsightsExtension:19
Asia Pasifik (Mumbai)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:49
Asia Pasifik (Osaka)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension:32
Asia Pasifik (Seoul)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:50

Wilayah	ARN
Asia Pasifik (Singapura)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:51
Asia Pasifik (Sydney)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:51
Asia Pasifik (Tokyo)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:78
(Canada (Central)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:50
Kanada Barat (Calgary)	arn:aws:lambda:ca-west-1:946466191631:layer:LambdaInsightsExtension:11
Tiongkok (Beijing)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:41
Tiongkok (Ningxia);	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:41
Eropa (Frankfurt)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:51
Eropa (Irlandia)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:51
Eropa (London)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:51
Eropa (Milan)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:42
Eropa (Paris)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:50
Eropa (Spanyol)	arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension:26

Wilayah	ARN
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:48`
Eropa (Zürich)	`arn:aws:lambda:eu-central-2:033019950311:layer:LambdaInsightsExtension:25`
Israel (Tel Aviv)	`arn:aws:lambda:il-central-1:459530977127:layer:LambdaInsightsExtension:19`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:42`
Timur Tengah (UEA)	`arn:aws:lambda:me-central-1:732604637566:layer:LambdaInsightsExtension:25`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:50`

1.0.275.0

Versi 1.0.275.0 mencakup pembaruan ketergantungan penting untuk semua runtime yang kompatibel.

ARN untuk versi 1.0.275.0

Tabel berikut mencantumkan ARN yang akan digunakan untuk versi ekstensi ini di setiap AWS Wilayah yang tersedia.

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:49`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:49`

Wilayah	ARN
AS Barat (California Utara)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:49
AS Barat (Oregon)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:49
Afrika (Cape Town)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:40
Asia Pasifik (Hong Kong)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:40
Asia Pasifik (Hyderabad)	arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension:22
Asia Pasifik (Jakarta)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension:26
Asia Pasifik (Melbourne)	arn:aws:lambda:ap-southeast-4:158895979263:layer:LambdaInsightsExtension:17
Asia Pasifik (Mumbai)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:47
Asia Pasifik (Osaka)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension:30
Asia Pasifik (Seoul)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:48
Asia Pasifik (Singapura)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:49
Asia Pasifik (Sydney)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:49
Asia Pasifik (Tokyo)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:76

Wilayah	ARN
(Canada (Central)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:48
Kanada Barat (Calgary)	arn:aws:lambda:ca-west-1:946466191631:layer:LambdaInsightsExtension:9
Tiongkok (Beijing)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:39
Tiongkok (Ningxia);	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:39
Eropa (Frankfurt)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:49
Eropa (Irlandia)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:49
Eropa (London)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:49
Eropa (Milan)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:40
Eropa (Paris)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:48
Eropa (Spanyol)	arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension:24
Eropa (Stockholm)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:46
Eropa (Zürich)	arn:aws:lambda:eu-central-2:033019950311:layer:LambdaInsightsExtension:23
Israel (Tel Aviv)	arn:aws:lambda:il-central-1:459530977127:layer:LambdaInsightsExtension:17

Wilayah	ARN
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:40`
Timur Tengah (UEA)	`arn:aws:lambda:me-central-1:732604637566:layer:LambdaInsightsExtension:23`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:48`

1.0.273.0

Versi 1.0.273.0 mencakup perbaikan bug penting untuk semua runtime yang kompatibel, dan menambahkan dukungan untuk Canada West (Calgary).

ARN untuk versi 1.0.273.0

Tabel berikut mencantumkan ARN yang akan digunakan untuk versi ekstensi ini di setiap AWS Wilayah yang tersedia.

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:45`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:45`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:45`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:45`
Afrika (Cape Town)	`arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:35`

Wilayah	ARN
Asia Pasifik (Hong Kong)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:35
Asia Pasifik (Hyderabad)	arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension:17
Asia Pasifik (Jakarta)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension:21
Asia Pasifik (Melbourne)	arn:aws:lambda:ap-southeast-4:158895979263:layer:LambdaInsightsExtension:12
Asia Pasifik (Mumbai)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:43
Asia Pasifik (Osaka)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension:26
Asia Pasifik (Seoul)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:44
Asia Pasifik (Singapura)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:45
Asia Pasifik (Sydney)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:45
Asia Pasifik (Tokyo)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:72
(Canada (Central)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:44
Kanada Barat (Calgary)	arn:aws:lambda:ca-west-1:946466191631:layer:LambdaInsightsExtension:4
Tiongkok (Beijing)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:36

Wilayah	ARN
Tiongkok (Ningxia);	`arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:36`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:45`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:45`
Eropa (London)	`arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:45`
Eropa (Milan)	`arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:35`
Eropa (Paris)	`arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:44`
Eropa (Spanyol)	`arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension:19`
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:42`
Eropa (Zürich)	`arn:aws:lambda:eu-central-2:033019950311:layer:LambdaInsightsExtension:17`
Israel (Tel Aviv)	`arn:aws:lambda:il-central-1:459530977127:layer:LambdaInsightsExtension:12`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:35`
Timur Tengah (UEA)	`arn:aws:lambda:me-central-1:732604637566:layer:LambdaInsightsExtension:18`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:44`

1.0.229.0

Versi 1.0.229.0 mencakup perbaikan bug yang penting untuk semua runtime yang kompatibel, dan menambahkan dukungan untuk Wilayah Israel (Tel Aviv).

ARN untuk versi 1.0.229.0

Tabel berikut mencantumkan ARN yang akan digunakan untuk versi ekstensi ini di setiap AWS Wilayah yang tersedia.

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:38`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:38`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:38`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:38`
Afrika (Cape Town)	`arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:28`
Asia Pasifik (Hong Kong)	`arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:28`
Asia Pasifik (Hyderabad)	`arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension:10`
Asia Pasifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension:14`
Asia Pasifik (Melbourne)	`arn:aws:lambda:ap-southeast-4:158895979263:layer:LambdaInsightsExtension:5`

Wilayah	ARN
Asia Pasifik (Mumbai)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:36
Asia Pasifik (Osaka)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension:19
Asia Pasifik (Seoul)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:37
Asia Pasifik (Singapura)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:38
Asia Pasifik (Sydney)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:38
Asia Pasifik (Tokyo)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:60
Kanada (Pusat)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:37
Tiongkok (Beijing)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:29
Tiongkok (Ningxia);	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:29
Eropa (Frankfurt)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:38
Eropa (Irlandia)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:38
Eropa (London)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:38
Eropa (Milan)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:28

Wilayah	ARN
Eropa (Paris)	`arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:37`
Eropa (Spanyol)	`arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension:12`
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:35`
Eropa (Zürich)	`arn:aws:lambda:eu-central-2:033019950311:layer:LambdaInsightsExtension:11`
Israel (Tel Aviv)	`arn:aws:lambda:il-central-1:459530977127:layer:LambdaInsightsExtension:5`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:28`
Timur Tengah (UEA)	`arn:aws:lambda:me-central-1:732604637566:layer:LambdaInsightsExtension:11`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:37`

1.0.178.0

Versi 1.0.178.0 menambahkan dukungan untuk Wilayah berikut. AWS

- Asia Pasifik (Hyderabad)
- Asia Pasifik (Jakarta)
- Eropa (Spanyol)
- Eropa (Zürich)
- Timur Tengah (UEA)

ARN untuk versi 1.0.178.0

Tabel berikut mencantumkan ARN yang akan digunakan untuk versi ekstensi ini di setiap AWS Wilayah yang tersedia.

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:35`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:33`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:33`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:33`
Afrika (Cape Town)	`arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:25`
Asia Pasifik (Hong Kong)	`arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:25`
Asia Pasifik (Hyderabad)	`arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension:8`
Asia Pasifik (Jakarta)	`arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension:11`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:31`
Asia Pasifik (Osaka)	`arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension:2`
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:32`

Wilayah	ARN
Asia Pasifik (Singapura)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:33
Asia Pasifik (Sydney)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:33
Asia Pasifik (Tokyo)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:50
Kanada (Pusat)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:32
Tiongkok (Beijing)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:26
Tiongkok (Ningxia);	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:26
Eropa (Frankfurt)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:35
Eropa (Irlandia)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:33
Eropa (London)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:33
Eropa (Milan)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:25
Eropa (Paris)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:32
Eropa (Spanyol)	arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension:10
Eropa (Stockholm)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:30

Wilayah	ARN
Eropa (Zürich)	`arn:aws:lambda:eu-central-2:033019950311:layer:LambdaInsightsExtension:7`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:25`
Timur Tengah (UEA)	`arn:aws:lambda:me-central-1:732604637566:layer:LambdaInsightsExtension:9`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:32`

1.0.143.0

Versi 1.0.143.0 mencakup perbaikan bug dalam kompatibilitas Python 3.7 dan Go 1.x. Runtime Lambda Python 3.6 sudah tidak digunakan lagi. Untuk informasi selengkapnya, silakan lihat runtime Lambda.

ARN untuk versi 1.0.143.0

Tabel berikut mencantumkan ARN yang akan digunakan untuk versi ekstensi ini di setiap AWS Wilayah yang tersedia.

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:21`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:21`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:20`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:21`

Wilayah	ARN
Afrika (Cape Town)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:13
Asia Pasifik (Hong Kong)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:13
Asia Pasifik (Mumbai)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:21
Asia Pasifik (Osaka)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension:2
Asia Pasifik (Seoul)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:20
Asia Pasifik (Singapura)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:21
Asia Pasifik (Sydney)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:21
Asia Pasifik (Tokyo)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:32
Kanada (Pusat)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:20
Tiongkok (Beijing)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:14
Tiongkok (Ningxia);	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:14
Eropa (Frankfurt)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:21
Eropa (Irlandia)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:21

Wilayah	ARN
Eropa (London)	`arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:21`
Eropa (Milan)	`arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:13`
Eropa (Paris)	`arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:20`
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:20`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:13`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:20`

1.0.135.0

Versi 1.0.135.0 mencakup perbaikan bug untuk cara kerja Wawasan Lambda mengumpulkan dan melaporkan penggunaan deskriptor cakram dan berkas. Dalam versi ekstensi sebelumnya, metrik `tmp_free` melaporkan ruang kosong maksimum dalam direktori `/tmp` saat sebuah fungsi berjalan. Versi ini mengubah metrik untuk melaporkan nilai minimum sebagai gantinya, membuatnya lebih berguna saat menilai penggunaan disk. Untuk informasi selengkapnya tentang kuota penyimpanan direktori `tmp`, silakan lihat Kuota Lambda.

Versi 1.0.135.0 juga sekarang melaporkan penggunaan deskriptor berkas (`fd_use`dan`fd_max`) sebagai nilai maksimum di seluruh proses daripada melaporkan tingkat sistem operasi.

ARN untuk versi 1.0.135.0

Tabel berikut mencantumkan ARN yang akan digunakan untuk versi ekstensi ini di setiap AWS Wilayah yang tersedia.

Wilayah	ARN
AS Timur (Virginia Utara)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:18
AS Timur (Ohio)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:18
AS Barat (California Utara)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:18
AS Barat (Oregon)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:18
Afrika (Cape Town)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:11
Asia Pasifik (Hong Kong)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:11
Asia Pasifik (Mumbai)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:18
Asia Pasifik (Osaka)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension:1
Asia Pasifik (Seoul)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:18
Asia Pasifik (Singapura)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:18
Asia Pasifik (Sydney)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:18
Asia Pasifik (Tokyo)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:25

Wilayah	ARN
Kanada (Pusat)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:18
Tiongkok (Beijing)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:11
Tiongkok (Ningxia);	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:11
Eropa (Frankfurt)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:18
Eropa (Irlandia)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:18
Eropa (London)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:18
Eropa (Milan)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:11
Eropa (Paris)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:18
Eropa (Stockholm)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:18
Timur Tengah (Bahrain)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:11
Amerika Selatan (Sao Paulo)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:18

1.0.119.0

ARN untuk versi 1.0.119.0

Tabel berikut mencantumkan ARN yang akan digunakan untuk versi ekstensi ini di setiap AWS Wilayah yang tersedia.

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:16`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:16`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:16`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:16`
Afrika (Cape Town)	`arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:9`
Asia Pasifik (Hong Kong)	`arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:9`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:16`
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:16`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:16`
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:16`
Asia Pasifik (Tokyo)	`arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:23`

Wilayah	ARN
Kanada (Pusat)	`arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:16`
Tiongkok (Beijing)	`arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:9`
Tiongkok (Ningxia);	`arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:9`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:16`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:16`
Eropa (London)	`arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:16`
Eropa (Milan)	`arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:9`
Eropa (Paris)	`arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:16`
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:16`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:9`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:16`

1.0.98.0

Versi ini menghapus pencatatan pada log yang tidak perlu dan juga mengatasi masalah dengan CLI AWS Serverless Application Model lokal yang menginvokasi. Untuk informasi selengkapnya tentang

masalah ini, lihat Menambahkan LambdaInsightsExtension hasil dalam batas waktu dengan 'sam local invoke'.

ARN untuk versi 1.0.98.0

Tabel berikut mencantumkan ARN yang akan digunakan untuk versi ekstensi ini di setiap AWS Wilayah yang tersedia.

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:14`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:14`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:14`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:14`
Afrika (Cape Town)	`arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:8`
Asia Pasifik (Hong Kong)	`arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:8`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:14`
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:14`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:14`
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:14`

Wilayah	ARN
Asia Pasifik (Tokyo)	`arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:14`
Kanada (Pusat)	`arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:14`
Tiongkok (Beijing)	`arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:8`
Tiongkok (Ningxia);	`arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:8`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:14`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:14`
Eropa (London)	`arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:14`
Eropa (Milan)	`arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:8`
Eropa (Paris)	`arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:14`
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:14`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:8`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:14`

1.0.89.0

Versi ini mengoreksi stempel waktu event performa untuk selalu mewakili awal invokasi fungsi.

ARN untuk versi 1.0.89.0

Tabel berikut mencantumkan ARN yang akan digunakan untuk versi ekstensi ini di setiap AWS Wilayah yang tersedia.

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:12`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:12`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:12`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:12`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:12`
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:12`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:12`
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:12`
Asia Pasifik (Tokyo)	`arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:12`
Kanada (Pusat)	`arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:12`

Wilayah	ARN
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:12`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:12`
Eropa (London)	`arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:12`
Eropa (Paris)	`arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:12`
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:12`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:12`

1.0.86.0

Dengan ekstensi versi 1.0.54.0, metrik memori terkadang dilaporkan secara tidak benar dan terkadang lebih tinggi dari 100%. Versi 1.0.86.0 memperbaiki masalah pengukuran memori dengan menggunakan data event yang sama dengan metrik platform Lambda. Ini berarti bahwa Anda mungkin melihat perubahan yang dramatis pada nilai metrik memori yang terrekam. Hal ini dicapai dengan menggunakan API Lambda Log yang baru. Hal ini memberikan pengukuran yang lebih akurat dari penggunaan memori sandbox Lambda. Namun demikian, sesuatu yang harus diperhatikan adalah bahwa API Log Lambda tidak dapat mengirimkan event pelaporan platform jika sebuah fungsi sandbox dinyatakan timeout dan kemudian spin. Dalam kasus ini. Wawasan Lambda tidak dapat merekam metrik invokasi. Untuk informasi selengkapnya tentang API Log Lambda, silakan lihat API Log Lambda AWS.

Fitur-fitur baru dalam versi 1.0.86.0

- Menggunakan API Log Lambda untuk memperbaiki metrik memori. Ini memecahkan masalah sebelumnya di mana statistik memori lebih besar dari 100%.
- Memperkenalkan `Init Duration` sebagai CloudWatch metrik baru.

- Menggunakan ARN invokasi untuk menambahkan dimensi versi untuk alias dan versi invoke. Jika Anda menggunakan alias atau versi Lambda untuk mencapai deployment tambahan (seperti deployment biru-hijau), Anda dapat melihat metrik Anda berdasarkan alias invoke. Dimensi versi tidak diterapkan jika fungsi tidak menggunakan alias atau versi. Untuk informasi selengkapnya, silakan lihat alias-alias fungsi Lambda.
- Tambahkan `billed_mb_ms field` ke peristiwa performa untuk menampilkan biaya untuk setiap kali menginvokasi. Penambahan ini tidak mempertimbangkan biaya yang terkait dengan konkurensi yang disediakan.
- Tambahkan bidang `billed_duration` dan `duration` untuk event performa.

ARN untuk versi 1.0.86.0

Tabel berikut mencantumkan ARN yang akan digunakan untuk versi ekstensi ini di setiap AWS Wilayah yang tersedia.

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:11`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:11`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:11`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:11`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:11`
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:11`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:11`

Wilayah	ARN
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:11`
Asia Pasifik (Tokyo)	`arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:11`
Kanada (Pusat)	`arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:11`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:11`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:11`
Eropa (London)	`arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:11`
Eropa (Paris)	`arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:11`
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:11`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:11`

1.0.54.0

Versi 1.0.54.0 adalah rilisan awal ekstensi Wawasan Lambda.

ARN untuk versi 1.0.54.0

Tabel berikut mencantumkan ARN yang akan digunakan untuk versi ekstensi ini di setiap AWS Wilayah yang tersedia.

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:2`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:2`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:2`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:2`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:2`
Asia Pasifik (Seoul)	`arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:2`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:2`
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:2`
Asia Pasifik (Tokyo)	`arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:2`
Kanada (Pusat)	`arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:2`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:2`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:2`

Wilayah	ARN
Eropa (London)	`arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:2`
Eropa (Paris)	`arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:2`
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:2`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:2`

Platform ARM64

Bagian ini mencantumkan versi ekstensi Lambda Insights untuk platform ARM64, dan ARN yang akan digunakan untuk ekstensi ini di setiap Wilayah. AWS

1.0.295.0

Versi 1.0.295.0 mencakup pembaruan ketergantungan untuk semua runtime yang kompatibel.

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:18`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension-Arm64:20`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:16`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:18`
Afrika (Cape Town)	`arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension-Arm64:16`

Wilayah	ARN
Asia Pasifik (Hong Kong)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension-Arm64:16
Asia Pasifik (Hyderabad)	arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension-Arm64:4
Asia Pasifik (Jakarta)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension-Arm64:16
Asia Pasifik (Mumbai)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension-Arm64:20
Asia Pasifik (Osaka)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension-Arm64:15
Asia Pasifik (Seoul)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:17
Asia Pasifik (Singapura)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:18
Asia Pasifik (Sydney)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:18
Asia Pasifik (Tokyo)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:29
Kanada (Pusat)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:16
Eropa (Frankfurt)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:18
Eropa (Irlandia)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:18
Eropa (London)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:18

Wilayah	ARN
Eropa (Milan)	`arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension-Arm64:16`
Eropa (Paris)	`arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension-Arm64:16`
Eropa (Spanyol)	`arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension-Arm64:4`
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension-Arm64:16`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension-Arm64:16`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:16`

1.0.275.0

Versi 1.0.275.0 mencakup perbaikan bug untuk semua runtime yang kompatibel dan dukungan untuk Wilayah Eropa (Spanyol) dan Asia Pasifik (Hyderabad).

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:16`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension-Arm64:18`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:14`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:16`

Wilayah	ARN
Afrika (Cape Town)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension-Arm64:14
Asia Pasifik (Hong Kong)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension-Arm64:14
Asia Pasifik (Hyderabad)	arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension-Arm64:2
Asia Pasifik (Jakarta)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension-Arm64:14
Asia Pasifik (Mumbai)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension-Arm64:18
Asia Pasifik (Osaka)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension-Arm64:13
Asia Pasifik (Seoul)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:15
Asia Pasifik (Singapura)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:16
Asia Pasifik (Sydney)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:16
Asia Pasifik (Tokyo)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:27
Kanada (Pusat)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:14
Eropa (Frankfurt)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:16
Eropa (Irlandia)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:16

Wilayah	ARN
Eropa (London)	`arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:16`
Eropa (Milan)	`arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension-Arm64:14`
Eropa (Paris)	`arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension-Arm64:14`
Eropa (Spanyol)	`arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension-Arm64:2`
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension-Arm64:14`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension-Arm64:14`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:14`

1.0.273.0

Versi 1.0.273.0 mencakup perbaikan bug untuk semua runtime yang kompatibel.

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:12`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension-Arm64:14`
AS Barat (California Utara)	`arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:9`

Wilayah	ARN
AS Barat (Oregon)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:12
Afrika (Cape Town)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension-Arm64:9
Asia Pasifik (Hong Kong)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension-Arm64:9
Asia Pasifik (Jakarta)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension-Arm64:9
Asia Pasifik (Mumbai)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension-Arm64:14
Asia Pasifik (Osaka)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension-Arm64:9
Asia Pasifik (Seoul)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:11
Asia Pasifik (Singapura)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:12
Asia Pasifik (Sydney)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:12
Asia Pasifik (Tokyo)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:23
Kanada (Pusat)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:10
Eropa (Frankfurt)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:12
Eropa (Irlandia)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:12

Wilayah	ARN
Eropa (London)	`arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:12`
Eropa (Milan)	`arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension-Arm64:9`
Eropa (Paris)	`arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension-Arm64:10`
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension-Arm64:10`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension-Arm64:9`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:10`

1.0.229.0

Versi 1.0.229.0 termasuk perbaikan bug untuk semua runtime yang kompatibel. Versi ini juga menambahkan dukungan untuk Wilayah berikut:

- AS Barat (California Utara)
- Afrika (Cape Town)
- Asia Pasifik (Hong Kong)
- Asia Pasifik (Jakarta)
- Asia Pasifik (Osaka)
- Asia Pasifik (Seoul)
- Kanada (Pusat)
- Eropa (Milan)
- Eropa (Paris)
- Eropa (Stockholm)
- Timur Tengah (Bahrain)

- Amerika Selatan (Sao Paulo)

Wilayah	ARN
AS Timur (Virginia Utara)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:5
AS Timur (Ohio)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension-Arm64:7
AS Barat (California Utara)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:3
AS Barat (Oregon)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:5
Afrika (Cape Town)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension-Arm64:2
Asia Pasifik (Hong Kong)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension-Arm64:2
Asia Pasifik (Jakarta)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension-Arm64:2
Asia Pasifik (Mumbai)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension-Arm64:7
Asia Pasifik (Osaka)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension-Arm64:2
Asia Pasifik (Seoul)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:4
Asia Pasifik (Singapura)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:5
Asia Pasifik (Sydney)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:5

Wilayah	ARN
Asia Pasifik (Tokyo)	`arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:11`
Kanada (Pusat)	`arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:3`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:5`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:5`
Eropa (London)	`arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:5`
Eropa (Spanyol)	`arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension-Arm64:2`
Eropa (Paris)	`arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension-Arm64:3`
Eropa (Stockholm)	`arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension-Arm64:3`
Timur Tengah (Bahrain)	`arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension-Arm64:2`
Amerika Selatan (Sao Paulo)	`arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:3`

1.0.135.0

Versi 1.0.135.0 mencakup perbaikan bug untuk cara kerja Wawasan Lambda mengumpulkan dan melaporkan penggunaan deskriptor cakram dan berkas. Dalam versi ekstensi sebelumnya, metrik `tmp_free` melaporkan ruang kosong maksimum dalam direktori `/tmp` saat sebuah fungsi berjalan. Versi ini mengubah metrik untuk melaporkan nilai minimum sebagai gantinya, membuatnya lebih

berguna saat menilai penggunaan disk. Untuk informasi selengkapnya tentang kuota penyimpanan direktori `tmp`, silakan lihat Kuota Lambda.

Versi 1.0.135.0 juga sekarang melaporkan penggunaan deskriptor berkas (`fd_used`dan`fd_max`) sebagai nilai maksimum di seluruh proses daripada melaporkan tingkat sistem operasi.

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:2`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension-Arm64:2`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:2`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension-Arm64:2`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:2`
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:2`
Asia Pasifik (Tokyo)	`arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:2`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:2`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:2`
Eropa (London)	`arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:2`

1.0.119.0

Wilayah	ARN
AS Timur (Virginia Utara)	`arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:1`
AS Timur (Ohio)	`arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension-Arm64:1`
AS Barat (Oregon)	`arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:1`
Asia Pasifik (Mumbai)	`arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension-Arm64:1`
Asia Pasifik (Singapura)	`arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:1`
Asia Pasifik (Sydney)	`arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:1`
Asia Pasifik (Tokyo)	`arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:1`
Eropa (Frankfurt)	`arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:1`
Eropa (Irlandia)	`arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:1`
Eropa (London)	`arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:1`

Menggunakan konsol untuk mengaktifkan Wawasan Lambda pada fungsi Lambda yang sudah ada

Gunakan langkah-langkah berikut pada Konsol Lambda untuk mengaktifkan Wawasan Lambda pada fungsi Lambda yang sudah ada.

Untuk mengaktifkan Wawasan Lambda pada fungsi Lambda

1. Buka AWS Lambda konsol di https://console.aws.amazon.com/lambda/.
2. Pilih nama fungsi, lalu pilih tab Konfigurasi pada layar berikut.
3. Di bawah tab Konfigurasi, pilih Alat pemantauan dan operasi di menu navigasi kiri, lalu pilih Edit.

   Anda akan diarahkan ke layar tempat Anda dapat mengedit alat pemantauan.
4. Dengan pemantauan yang disempurnakan oleh Lambda Insights, pilih Edit.
5. Di bawah CloudWatch Lambda Insights, aktifkan pemantauan yang disempurnakan, lalu pilih Simpan.

Menggunakan AWS CLI untuk mengaktifkan Lambda Insights pada fungsi Lambda yang ada

Ikuti langkah-langkah berikut untuk menggunakan Lambda Insights AWS CLI untuk mengaktifkan Lambda Insights pada fungsi Lambda yang ada.

Langkah 1: Memperbarui izin fungsi

Perbarui izin fungsi

- Lampirkan kebijakan IAM CloudWatchLambdaInsightsExecutionRolePolicyterkelola ke peran eksekusi fungsi dengan memasukkan perintah berikut.

```
aws iam attach-role-policy \
--role-name function-execution-role \
--policy-arn "arn:aws:iam::aws:policy/CloudWatchLambdaInsightsExecutionRolePolicy"
```

Langkah 2: Melakukan instalasi ekstensi Lambda

Pasang ekstensi Lambda dengan memasukkan perintah berikut. Ganti nilai ARN untuk parameter `layers` dengan ARN yang cocok dengan wilayah Anda dan versi ekstensi yang ingin Anda gunakan. Untuk informasi selengkapnya, lihat Versi yang tersedia untuk ekstensi Wawasan lambda..

```
aws lambda update-function-configuration \
   --function-name function-name \
   --layers "arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:14"
```

Langkah 3: Aktifkan titik akhir VPC CloudWatch Log

Langkah ini diperlukan hanya untuk fungsi yang berjalan di subnet pribadi tanpa akses internet, dan jika Anda belum mengkonfigurasi titik akhir CloudWatch Logs virtual private cloud (VPC).

Jika Anda perlu melakukan langkah ini, masukkan perintah berikut, ganti placeholder dengan informasi untuk VPC Anda.

Untuk informasi selengkapnya, lihat Menggunakan CloudWatch Log dengan Titik Akhir VPC Antarmuka.

```
aws ec2 create-vpc-endpoint \
--vpc-id vpcId \
--vpc-endpoint-type Interface \
--service-name com.amazonaws.region.logs \
--subnet-id subnetId
--security-group-id securitygroupId
```

Menggunakan AWS SAM CLI untuk mengaktifkan Lambda Insights pada fungsi Lambda yang ada

Ikuti langkah-langkah berikut untuk menggunakan Lambda Insights AWS SAM AWS CLI untuk mengaktifkan Lambda Insights pada fungsi Lambda yang ada.

Jika Anda belum menginstal AWS SAM CLI versi terbaru, Anda harus menginstal atau memutakhirkannya terlebih dahulu. Untuk informasi selengkapnya, lihat Menginstal AWS SAM CLI.

Langkah 1: Melakukan instalasi lapisan

Untuk membuat ekstensi Wawasan Lambda tersedia pada semua fungsi Lambda Anda, tambahkan properti `Layers` dengan bagian `Globals` dari template SAM Anda dengan ARN dari lapisan Wawasan Lambda. Contoh di bawah menggunakan lapisan untuk rilisan awal Wawasan Lambda. Untuk versi rilisan terbaru dari lapisan ekstensi Wawasan Lambda, silakan lihat Versi yang tersedia untuk ekstensi Wawasan lambda..

```
Globals:
  Function:
    Layers:
      - !Sub "arn:aws:lambda:
${AWS::Region}:580247275435:layer:LambdaInsightsExtension:14"
```

Untuk mengaktifkan lapisan ini pada satu fungsi saja, tambahkan properti `Layers` kepada fungsi sebagaimana diperlihatkan dalam contoh ini.

```
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      Layers:
        - !Sub "arn:aws:lambda:
${AWS::Region}:580247275435:layer:LambdaInsightsExtension:14"
```

Langkah 2: Menambahkan kebijakan yang sudah dikelola

Untuk setiap fungsi, tambahkan kebijakan CloudWatchLambdaInsightsExecutionRolePolicyIAM.

AWS SAM tidak mendukung kebijakan global, jadi Anda harus mengaktifkannya pada setiap fungsi satu per satu, seperti yang ditunjukkan dalam contoh ini. Untuk informasi selengkapnya mengenai global, silakan lihat Bagian Global.

```
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      Policies:
        - CloudWatchLambdaInsightsExecutionRolePolicy
```

Menginvokasi secara lokal

AWS SAM CLI mendukung ekstensi Lambda. Namun demikian, setiap invokasi yang dilaksanakan secara lokal mengatur ulang lingkungan runtime. Data Wawasan Lambda tidak akan tersedia dari invokasi lokal karena runtime dimulai ulang tanpa adanya event shutdown. Untuk informasi selengkapnya, lihat Rilis 1.6.0 - Menambahkan dukungan untuk pengujian ekstensi lokal. AWS Lambda

Pemecahan Masalah

Untuk memecahkan masalah instalasi Wawasan Lambda Anda, tambahkan variabel lingkungan berikut ke fungsi Lambda Anda untuk mengaktifkan pencatatan debug.

```
Resources:
  MyFunction:
```

```
    Type: AWS::Serverless::Function
    Properties:
      Environment:
        Variables:
          LAMBDA_INSIGHTS_LOG_LEVEL: info
```

Menggunakan AWS CloudFormation untuk mengaktifkan Lambda Insights pada fungsi Lambda yang ada

Ikuti langkah-langkah berikut yang akan digunakan AWS CloudFormation untuk mengaktifkan Lambda Insights pada fungsi Lambda yang ada.

Langkah 1: Melakukan instalasi lapisan

Tambahkan lapisan Wawasan Lambda ke properti `Layers` di dalam ARN lapisan Wawasan Lambda. Contoh di bawah menggunakan lapisan untuk rilisan awal Wawasan Lambda. Untuk versi rilisan terbaru dari lapisan ekstensi Wawasan Lambda, silakan lihat Versi yang tersedia untuk ekstensi Wawasan lambda..

```
Resources:
  MyFunction:
    Type: AWS::Lambda::Function
    Properties:
      Layers:
        - !Sub "arn:aws:lambda:
${AWS::Region}:580247275435:layer:LambdaInsightsExtension:14"
```

Langkah 2: Menambahkan kebijakan yang sudah dikelola

Tambahkan kebijakan CloudWatchLambdaInsightsExecutionRolePolicyIAM ke peran eksekusi fungsi Anda.

```
Resources:
  MyFunctionExecutionRole:
    Type: 'AWS::IAM::Role'
    Properties:
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/CloudWatchLambdaInsightsExecutionRolePolicy'
```

Langkah 3: (Opsional) Menambahkan titik akhir VPC

Langkah ini diperlukan hanya untuk fungsi yang berjalan di subnet pribadi tanpa akses internet, dan jika Anda belum mengkonfigurasi titik akhir CloudWatch Logs virtual private cloud (VPC). Untuk informasi selengkapnya, lihat Menggunakan CloudWatch Log dengan Titik Akhir VPC Antarmuka.

```
Resources:
  CloudWatchLogsVpcPrivateEndpoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      PrivateDnsEnabled: 'true'
      VpcEndpointType: Interface
      VpcId: !Ref: VPC
      ServiceName: !Sub com.amazonaws.${AWS::Region}.logs
      SecurityGroupIds:
        - !Ref InterfaceVpcEndpointSecurityGroup
      SubnetIds:
        - !Ref PublicSubnet01
        - !Ref PublicSubnet02
        - !Ref PublicSubnet03
```

Menggunakan AWS CDK untuk mengaktifkan Lambda Insights pada fungsi Lambda yang ada

Ikuti langkah-langkah berikut untuk menggunakan Lambda Insights AWS CDK untuk mengaktifkan Lambda Insights pada fungsi Lambda yang ada. Untuk menggunakan langkah-langkah ini, Anda harus sudah menggunakan AWS CDK untuk mengelola sumber daya Anda.

Perintah di bagian ini ada di TypeScript.

Pertama-tama, perbarui izin fungsi.

```
executionRole.addManagedPolicy(
 ManagedPolicy.fromAwsManagedPolicyName('CloudWatchLambdaInsightsExecutionRolePolicy')
);
```

Berikutnya, pasang ekstensi pada fungsi Lambda. Ganti nilai ARN untuk parameter `layerArn` dengan ARN yang cocok dengan Wilayah Anda dan versi ekstensi yang ingin Anda gunakan. Untuk informasi selengkapnya, lihat Versi yang tersedia untuk ekstensi Wawasan lambda..

```
import lambda = require('@aws-cdk/aws-lambda');
const layerArn = 'arn:aws:lambda:us-
west-1:580247275435:layer:LambdaInsightsExtension:14';
```

```
const layer = lambda.LayerVersion.fromLayerVersionArn(this, 'LayerFromArn', layerArn);
```

Jika perlu, aktifkan titik akhir virtual private cloud (VPC) untuk Log. CloudWatch Langkah ini diperlukan hanya untuk fungsi yang berjalan di subnet pribadi tanpa akses internet, dan jika Anda belum mengonfigurasi titik akhir VPC CloudWatch Log.

```
const cloudWatchLogsEndpoint = vpc.addInterfaceEndpoint('cwl-gateway', {
  service: InterfaceVpcEndpointAwsService.CLOUDWATCH_LOGS,
});

cloudWatchLogsEndpoint.connections.allowDefaultPortFromAnyIpv4();
```

Gunakan Kerangka Kerja Nirserver untuk mengaktifkan Wawasan Lambda pada fungsi Lambda yang sudah ada

Ikuti langkah-langkah ini untuk menggunakan Kerangka Kerja Nirserver guna mengaktifkan Wawasan Lambda pada fungsi Lambda yang sudah ada. Untuk informasi selengkapnya tentang Kerangka Kerja Nirserver, silakan lihat serverless.com.

Hal ini dilakukan melalui plugin Wawasan Lambda untuk Nirserver. Untuk informasi lebih lanjut, lihat serverless-plugin-lambda-insights.

Jika Anda belum melakukan instalasi antarmuka baris-perintah Nirserver versi terbaru, Anda harus melakukan instalasi atau meningkatkannya terlebih dahulu. Untuk informasi selengkapnya, lihat Memulai Kerangka Tanpa Server Open Source &. AWS

Gunakan Kerangka Kerja Nirserver guna mengaktifkan Wawasan Lambda pada fungsi Lambda.

1. Menginstal plugin Nirserver untuk Wawasan Lambda dengan menjalankan perintah berikut di direktori Nirserver Anda:

   ```
   npm install --save-dev serverless-plugin-lambda-insights
   ```

2. Dalam berkas `serverless.yml`, tambahkan plugin di bagian `plugins` seperti yang ditunjukkan:

   ```
   provider:
     name: aws
   plugins:
   ```

```
  - serverless-plugin-lambda-insights
```

3. Aktifkan Wawasan Lambda.

   - Anda dapat mengaktifkan Wawasan Lambda satu-persatu per fungsi dengan menambahkan properti berikut ke berkas serverless.yml

     ```
     functions:
       myLambdaFunction:
         handler: src/app/index.handler
         lambdaInsights: true #enables Lambda Insights for this function
     ```

   - Anda dapat mengaktifkan Wawasan Lambda untuk semua fungsi dalam berkas `serverless.yml` dengan menambahkan bagian kustom berikut:

     ```
     custom:
       lambdaInsights:
         defaultLambdaInsights: true #enables Lambda Insights for all functions
     ```

4. Deploy ulang layanan Nirserver dengan memasukkan perintah berikut:

   ```
   serverless deploy
   ```

   Hal ini akan men-deploy ulang semua fungsi dan mengaktifkan Wawasan Lambda untuk fungsi-fungsi tersebut yang telah Anda tentukan. Hal ini mengaktifkan Wawasan Lambda dengan menambahkan lapisan Wawasan Lambda dan melampirkan izin yang diperlukan menggunakan kebijakan IAM `arn:aws:iam::aws:policy/CloudWatchLambdaInsightsExecutionRolePolicy`.

Mengaktifkan Wawasan Lambda pada deployment image kontainer Lambda

Untuk mengaktifkan Wawasan Lambda pada fungsi Lambda yang di-deploy sebagai image kontainer, tambahkan baris di Dockerfile Anda. Baris-baris ini akan melakukan instalasi agen Wawasan Lambda sebagai sebuah ekstensi dalam image kontainer Anda. Baris-baris yang ditambahkan berbeda untuk kontainer x86-64 dan kontainer ARM64.

> **Note**
>
> Agen Wawasan Lambda hanya didukung pada runtime Lambda yang menggunakan Amazon Linux 2.

Topik

- Deployment image kontainer x86-64
- Deployment image kontainer ARM64

Deployment image kontainer x86-64

Untuk mengaktifkan Wawasan Lambda pada fungsi Lambda yang di-deploy sebagai image kontainer yang berjalan pada kontainer x86-64, tambahkan baris di bawah ini ke Dockfire Anda. Baris-baris ini akan melakukan instalasi agen Wawasan Lambda sebagai sebuah ekstensi dalam image kontainer Anda.

```
RUN curl -O https://lambda-insights-extension.s3-ap-northeast-1.amazonaws.com/
amazon_linux/lambda-insights-extension.rpm && \
    rpm -U lambda-insights-extension.rpm && \
    rm -f lambda-insights-extension.rpm
```

Setelah Anda membuat fungsi Lambda, tetapkan kebijakan CloudWatchLambdaInsightsExecutionRolePolicyIAM ke peran eksekusi fungsi, dan Lambda Insights diaktifkan pada fungsi Lambda berbasis gambar kontainer.

> **Note**
>
> Untuk menggunakan versi lama dari ekstensi Wawasan Lambda, ganti URL di perintah sebelumnya dengan URL ini: `https://lambda-insights-extension.s3-ap-northeast-1.amazonaws.com/amazon_linux/lambda-insights-extension.1.0.111.0.rpm`. Saat ini, hanya Wawasan Lambda versi 1.0.111.0 dan versi selanjutnya yang tersedia. Untuk informasi selengkapnya, lihat Versi yang tersedia untuk ekstensi Wawasan lambda..

Verifikasi ciri khas paket agen Wawasan Lambda pada sebuah server Linux

1. Masukkan perintah berikut untuk mengunduh kunci publik.

```
shell$ wget https://lambda-insights-extension.s3-ap-northeast-1.amazonaws.com/
lambda-insights-extension.gpg
```

2. Masukkan perintah berikut untuk mengimpor kunci publik ke dalam keyring Anda.

```
shell$ gpg --import lambda-insights-extension.gpg
```

   Hasil akhirnya akan serupa dengan yang berikut ini. Catat nilai `key`, Anda akan membutuhkannya pada langkah berikutnya. Dalam contoh hasil akhir ini, nilai kunci adalah `848ABDC8`.

```
gpg: key 848ABDC8: public key "Amazon Lambda Insights Extension" imported
gpg: Total number processed: 1
gpg: imported: 1  (RSA: 1)
```

3. Verifikasi sidik jari dengan memasukkan perintah berikut. Ganti `key-value` dengan nilai kunci dari langkah sebelumnya.

```
shell$  gpg --fingerprint key-value
```

   String sidik jari di hasil akhir perintah ini seharusnya `E0AF FA11 FFF3 5BD7 349E E222 479C 97A1 848A BDC8`. Jika string sidik jari tidak cocok, jangan instal agen dan hubungi AWS.

4. Setelah Anda memverifikasi sidik jari, Anda dapat menggunakannya untuk memverifikasi paket agen Wawasan Lambda. Unduh berkas tanda pembeda standar paket dengan memasukkan perintah berikut.

```
shell$  wget https://lambda-insights-extension.s3-ap-northeast-1.amazonaws.com/
amazon_linux/lambda-insights-extension.rpm.sig
```

5. Verifikasi tanda pembeda dengan memasukkan perintah berikut.

```
shell$ gpg --verify lambda-insights-extension.rpm.sig lambda-insights-extension.rpm
```

   Output-nya akan terlihat seperti berikut ini:

```
gpg: Signature made Thu 08 Apr 2021 06:41:00 PM UTC using RSA key ID 848ABDC8
gpg: Good signature from "Amazon Lambda Insights Extension"
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: E0AF FA11 FFF3 5BD7 349E  E222 479C 97A1 848A BDC8
```

Dalam output yang diharapkan, mungkin ada peringatan tentang tanda pembeda yang terpercaya. Kunci hanya dapat dipercaya jika Anda atau seseorang yang Anda percaya telah menandatanganinya. Ini tidak berarti bahwa tanda pembeda Anda tidak valid, hanya saja Anda belum memverifikasi kunci publiknya.

Jika outputnya berisi `BAD signature`, periksa apakah Anda melakukan langkah-langkahnya dengan benar. Jika Anda terus mendapatkan `BAD signature` respons, hubungi AWS dan hindari menggunakan file yang diunduh.

Contoh x86-64

Bagian ini mencakup satu contoh pengaktifan Wawasan Lambda pada fungsi Lambda Python berbasis citra kontainer.

Contoh pengaktifan Wawasan Lambda pada citra kontainer Lambda

1. Buat sebuah Dockerfile yang mirip dengan berikut ini:

```
FROM public.ecr.aws/lambda/python:3.8

// extra lines to install the agent here
RUN curl -O https://lambda-insights-extension.s3-ap-northeast-1.amazonaws.com/
amazon_linux/lambda-insights-extension.rpm && \
    rpm -U lambda-insights-extension.rpm && \
    rm -f lambda-insights-extension.rpm

COPY index.py ${LAMBDA_TASK_ROOT}
CMD [ "index.handler" ]
```

2. Buat sebuah file Python bernama `index.py` yang serupa dengan yang berikut ini:

```
def handler(event, context):
  return {
    'message': 'Hello World!'
```

```
  }
```

3. Masukan Dockerfile dan `index.py` dalam direktori yang sama. Kemudian, dalam direktori itu, jalankan langkah-langkah berikut untuk membangun citra docker dan unggah ke Amazon ECR.

```
// create an ECR repository
aws ecr create-repository --repository-name test-repository
// build the docker image
docker build -t test-image .
// sign in to AWS
aws ecr get-login-password | docker login --username AWS --password-stdin
 "${ACCOUNT_ID}".dkr.ecr."${REGION}".amazonaws.com
// tag the image
docker tag test-image:latest "${ACCOUNT_ID}".dkr.ecr."${REGION}".amazonaws.com/
test-repository:latest
// push the image to ECR
docker push "${ACCOUNT_ID}".dkr.ecr."${REGION}".amazonaws.com/test-
repository:latest
```

4. Gunakan image Amazon ECR yang baru saja Anda buat untuk membuat fungsi Lambda.
5. Tetapkan kebijakan CloudWatchLambdaInsightsExecutionRolePolicyIAM ke peran eksekusi fungsi.

Deployment image kontainer ARM64

Untuk mengaktifkan Wawasan Lambda pada fungsi Lambda yang di-deploy sebagai image kontainer yang berjalan pada kontainer AL2_aarch64 (yang menggunakan arsitektur ARM64), tambahkan baris berikut ini ke Dockfire Anda. Baris-baris ini akan melakukan instalasi agen Wawasan Lambda sebagai sebuah ekstensi dalam image kontainer Anda.

```
RUN curl -O https://lambda-insights-extension-arm64.s3-ap-northeast-1.amazonaws.com/
amazon_linux/lambda-insights-extension-arm64.rpm && \
    rpm -U lambda-insights-extension-arm64.rpm && \
    rm -f lambda-insights-extension-arm64.rpm
```

Setelah Anda membuat fungsi Lambda, tetapkan kebijakan CloudWatchLambdaInsightsExecutionRolePolicyIAM ke peran eksekusi fungsi, dan Lambda Insights diaktifkan pada fungsi Lambda berbasis gambar kontainer.

> **Note**
>
> Untuk menggunakan versi lama dari ekstensi Wawasan Lambda, ganti URL di perintah sebelumnya dengan URL ini: `https://lambda-insights-extension-arm64.s3-ap-northeast-1.amazonaws.com/amazon_linux/lambda-insights-extension-arm64.1.0.229.0.rpm`. Saat ini, hanya Wawasan Lambda versi 1.0.229.0 dan versi selanjutnya yang tersedia. Untuk informasi selengkapnya, lihat Versi yang tersedia untuk ekstensi Wawasan lambda..

Verifikasi ciri khas paket agen Wawasan Lambda pada sebuah server Linux

1. Masukkan perintah berikut untuk mengunduh kunci publik.

```
shell$ wget https://lambda-insights-extension-arm64.s3-ap-
northeast-1.amazonaws.com/lambda-insights-extension.gpg
```

2. Masukkan perintah berikut untuk mengimpor kunci publik ke dalam keyring Anda.

```
shell$ gpg --import lambda-insights-extension.gpg
```

   Hasil akhirnya akan serupa dengan yang berikut ini. Catat nilai `key`, Anda akan membutuhkannya pada langkah berikutnya. Dalam contoh hasil akhir ini, nilai kunci adalah `848ABDC8`.

```
gpg: key 848ABDC8: public key "Amazon Lambda Insights Extension" imported
gpg: Total number processed: 1
gpg: imported: 1  (RSA: 1)
```

3. Verifikasi sidik jari dengan memasukkan perintah berikut. Ganti `key-value` dengan nilai kunci dari langkah sebelumnya.

```
shell$  gpg --fingerprint key-value
```

   String sidik jari di hasil akhir perintah ini seharusnya `E0AF FA11 FFF3 5BD7 349E E222 479C 97A1 848A BDC8`. Jika string sidik jari tidak cocok, jangan instal agen dan hubungi AWS.

4. Setelah Anda memverifikasi sidik jari, Anda dapat menggunakannya untuk memverifikasi paket agen Wawasan Lambda. Unduh berkas tanda pembeda standar paket dengan memasukkan perintah berikut.

```
shell$  wget https://lambda-insights-extension-arm64.s3-ap-
northeast-1.amazonaws.com/amazon_linux/lambda-insights-extension-arm64.rpm.sig
```

5. Verifikasi tanda pembeda dengan memasukkan perintah berikut.

```
shell$ gpg --verify lambda-insights-extension-arm64.rpm.sig lambda-insights-
extension-arm64.rpm
```

Output-nya akan terlihat seperti berikut ini:

```
gpg: Signature made Thu 08 Apr 2021 06:41:00 PM UTC using RSA key ID 848ABDC8
gpg: Good signature from "Amazon Lambda Insights Extension"
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: E0AF FA11 FFF3 5BD7 349E  E222 479C 97A1 848A BDC8
```

Dalam output yang diharapkan, mungkin ada peringatan tentang tanda pembeda yang terpercaya. Kunci hanya dapat dipercaya jika Anda atau seseorang yang Anda percaya telah menandatanganinya. Ini tidak berarti bahwa tanda pembeda Anda tidak valid, hanya saja Anda belum memverifikasi kunci publiknya.

Jika outputnya berisi `BAD signature`, periksa apakah Anda melakukan langkah-langkahnya dengan benar. Jika Anda terus mendapatkan `BAD signature` respons, hubungi AWS dan hindari menggunakan file yang diunduh.

Contoh ARM64

Bagian ini mencakup satu contoh pengaktifan Wawasan Lambda pada fungsi Lambda Python berbasis citra kontainer.

Contoh pengaktifan Wawasan Lambda pada citra kontainer Lambda

1. Buat sebuah Dockerfile yang mirip dengan berikut ini:

```
FROM public.ecr.aws/lambda/python:3.8
```

```
// extra lines to install the agent here
RUN curl -O https://lambda-insights-extension-arm64.s3-ap-
northeast-1.amazonaws.com/amazon_linux/lambda-insights-extension-arm64.rpm && \
    rpm -U lambda-insights-extension-arm64.rpm && \
    rm -f lambda-insights-extension-arm64.rpm

COPY index.py ${LAMBDA_TASK_ROOT}
CMD [ "index.handler" ]
```

2. Buat sebuah file Python bernama `index.py` yang serupa dengan yang berikut ini:

```
def handler(event, context):
  return {
    'message': 'Hello World!'
  }
```

3. Masukan Dockerfile dan `index.py` dalam direktori yang sama. Kemudian, dalam direktori itu, jalankan langkah-langkah berikut untuk membangun citra docker dan unggah ke Amazon ECR.

```
// create an ECR repository
aws ecr create-repository --repository-name test-repository
// build the docker image
docker build -t test-image .
// sign in to AWS
aws ecr get-login-password | docker login --username AWS --password-stdin
 "${ACCOUNT_ID}".dkr.ecr."${REGION}".amazonaws.com
// tag the image
docker tag test-image:latest "${ACCOUNT_ID}".dkr.ecr."${REGION}".amazonaws.com/
test-repository:latest
// push the image to ECR
docker push "${ACCOUNT_ID}".dkr.ecr."${REGION}".amazonaws.com/test-
repository:latest
```

4. Gunakan image Amazon ECR yang baru saja Anda buat untuk membuat fungsi Lambda.
5. Tetapkan kebijakan CloudWatchLambdaInsightsExecutionRolePolicyIAM ke peran eksekusi fungsi.

Tinjau metrik Wawasan Lambda Anda

Setelah Anda menginstal ekstensi Lambda Insights pada fungsi Lambda yang telah dipanggil, Anda dapat menggunakan konsol untuk melihat metrik Anda. CloudWatch Anda dapat melihat sebuah gambaran umum multifungsi, atau berfokus pada satu fungsi.

Untuk daftar metrik Wawasan Lambda, silakan lihat Metrik dikumpulkan oleh Wawasan Lambda.

Untuk melihat gambaran umum multifungsi untuk metrik Wawasan Lambda Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi kiri, pada Wawasan Lambda, pilih Multi-fungsi.

   Bagian atas halaman menampilkan grafik dengan metrik gabungan atas semua fungsi Lambda Anda di Wilayah yang Wawasan Lambda-nya telah diaktifkan. Pada bagian bawah halaman terdapat sebuah tabel yang memuat daftar fungsi.
3. Untuk menyaring berdasarkan nama fungsi untuk mengurangi jumlah fungsi yang ditampilkan, ketik bagian nama fungsi di kotak dekat bagian atas halaman.
4. Untuk menambahkan tampilan ini ke dasbor sebagai widget, pilih Tambahkan ke dasbor.

Untuk melihat metrik pada fungsi tunggal

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi kiri, pada Wawasan Lambda, pilih fungsi tunggal.

   Bagian atas halaman menampilkan grafik dengan metrik untuk fungsi yang dipilih.
3. Jika Anda mengaktifkan X-Ray, maka Anda dapat memilih satu ID jejak tunggal. Hal ini membuka halaman Peta Jejak X-Ray untuk invokasi tersebut, dan dari sana Anda dapat memperkecil untuk melihat jejak terdistribusi dan layanan lain yang terlibat dalam penanganan transaksi khusus tersebut. Untuk informasi selengkapnya tentang Peta Jejak X-Ray, silakan lihat Menggunakan Peta Jejak X-Ray.
4. Untuk membuka Wawasan CloudWatch Log dan memperbesar kesalahan tertentu, pilih Lihat log menurut tabel di bagian bawah halaman.
5. Untuk menambahkan tampilan ini ke dasbor sebagai widget, pilih Tambahkan ke dasbor.

Integrasi dengan Wawasan Aplikasi

Amazon CloudWatch Application Insights membantu Anda memantau aplikasi dan mengidentifikasi serta menyiapkan metrik utama, log, dan alarm di seluruh sumber daya aplikasi dan tumpukan teknologi Anda. Untuk informasi selengkapnya, lihat Wawasan CloudWatch Aplikasi Amazon.

Anda dapat mengaktifkan Wawasan Applikasi untuk mengumpulkan data tambahan dari fungsi Lambda Anda. Jika Anda belum melakukannya, Anda dapat mengaktifkannya dengan memilih Konfigurasi Otomatis Wawasan Aplikasi di tab Wawasan Aplikasi di bawah tampilan performa di dasbor Wawasan Lambda.

Jika Anda telah menyiapkan CloudWatch Application Insights untuk memantau fungsi Lambda Anda, dasbor Application Insights muncul di bawah dasbor Lambda Insights, di tab Application Insights.

Metrik dikumpulkan oleh Wawasan Lambda

Wawasan Lambda mengumpulkan beberapa metrik dari fungsi Lambda tempatnya diinstal. Beberapa metrik ini tersedia sebagai data agregat deret waktu di CloudWatch Metrik. Metrik lain tidak digabungkan ke dalam data deret waktu tetapi dapat ditemukan di entri log format metrik yang disematkan dengan menggunakan CloudWatch Wawasan Log.

Metrik berikut tersedia sebagai data agregat deret waktu di CloudWatch Metrik di namespace. `LambdaInsights`

Nama metrik	Dimensi	Deskripsi
`cpu_total_time`	nama_fungsi nama_fungsi, versi	Jumlah `cpu_system_time` dan `cpu_user_time` . Satuan: Milidetik
`init_duration`	nama_fungsi nama_fungsi, versi	Jumlah waktu yang dihabiskan di fase `init` siklus hidup lingkungan pelaksanaan Lambda. Satuan: Milidetik

Nama metrik	Dimensi	Deskripsi
`memory_utilization`	nama_fungsi nama_fungsi, versi	Memori maksimum yang diukur sebagai persentase memori yang dialokasikan untuk fungsi. Satuan: Persen
`rx_bytes`	nama_fungsi nama_fungsi, versi	Jumlah byte yang diterima oleh fungsi. Satuan: Byte
`tmp_used`		Jumlah ruang yang digunakan dalam direktori `/tmp`. Satuan: Byte
`tx_bytes`	nama_fungsi nama_fungsi, versi	Jumlah byte yang dikirimkan oleh fungsi. Satuan: Byte
`total_memory`	nama_fungsi nama_fungsi, versi	Jumlah memori yang dialokasikan untuk fungsi Lambda Anda. Jumlah ini sama dengan ukuran memori fungsi Anda. Satuan: Megabyte

Nama metrik	Dimensi	Deskripsi
`total_network`	nama_fungsi nama_fungsi, versi	Jumlah `rx_bytes` dan `tx_bytes`. Bahkan untuk fungsi yang tidak mengerjakan tugas I/O, nilai ini biasanya lebih besar dari nol dikarenakan call jaringan yang dilakukan oleh runtime Lambda. Satuan: Byte
`used_memory_max`	nama_fungsi nama_fungsi, versi	Memori yang terukur dari sandbox fungsi. Satuan: Megabyte

Metrik berikut dapat ditemukan di entri log format metrik yang disematkan menggunakan Wawasan CloudWatch Log. Untuk informasi selengkapnya tentang Wawasan CloudWatch Log, lihat Menganalisis Data Log dengan Wawasan CloudWatch Log.

Untuk informasi selengkapnya tentang format metrik tersemat, silakan lihat Menyematkan metrik dalam log.

Nama metrik	Deskripsi
`cpu_system_time`	Jumlah waktu yang dihabiskan CPU untuk melaksanakan kode kernel. Satuan: Milidetik
`cpu_total_time`	Jumlah `cpu_system_time` dan `cpu_user_time` . Satuan: Milidetik

Nama metrik	Deskripsi	
cpu_user_time	Jumlah waktu yang dihabiskan CPU untuk melaksanakan kode pengguna. Satuan: Milidetik	
fd_max	Jumlah maksimum deskriptor berkas tersedia. Satuan: Hitungan	
fd_use	Jumlah maksimum deskriptor file yang digunakan. Satuan: Hitungan	
memory_utilization	Memori maksimum yang diukur sebagai persentase memori yang dialokasikan untuk fungsi. Satuan: Persen	
rx_bytes	Jumlah byte yang diterima oleh fungsi. Satuan: Byte	
tx_bytes	Jumlah byte yang dikirimkan oleh fungsi. Satuan: Byte	
threads_max	Jumlah utas yang digunakan oleh proses fungsi. Sebagai penulis fungsi, Anda tidak mengendalikan jumlah awal utas yang dibuat oleh runtime. Satuan: Hitungan	
tmp_max	Jumlah ruang yang tersedia di direktori /tmp. Satuan: Byte	

Nama metrik	Deskripsi	
`total_memory`	Jumlah memori yang dialokasikan untuk fungsi Lambda Anda. Jumlah ini sama dengan ukuran memori fungsi Anda. Satuan: Megabyte	
`total_network`	Jumlah `rx_bytes` dan `tx_bytes`. Bahkan untuk fungsi yang tidak mengerjakan tugas I/O, nilai ini biasanya lebih besar dari nol dikarenakan call jaringan yang dilakukan oleh runtime Lambda. Satuan: Byte	
`used_memory_max`	Memori yang terukur dari sandbox fungsi. Satuan: Byte	

Pemecahan masalah dan masalah yang diketahui

Langkah pertama untuk memecahkan masalah apa pun adalah mengaktifkan pencatatan log debug di ekstensi Wawasan Lambda. Untuk melakukan hal itu, tetapkan variabel lingkungan berikut pada fungsi Lambda Anda: `LAMBDA_INSIGHTS_LOG_LEVEL=info`. Untuk informasi selengkapnya, silakan lihat Menggunakan variabel lingkungan AWS Lambda.

Ekstensi menghasilkan log ke grup log yang sama sebagai fungsi Anda (`/aws/lambda/`*`function-name`*). Tinjau log tersebut untuk melihat apakah kesalahan mungkin berhubungan dengan masalah pengaturan.

Saya tidak melihat metrik apapun dari Wawasan Lambda

Jika Anda tidak melihat metrik Wawasan Lambda yang ingin Anda lihat, periksa kemungkinan berikut:

- Metrik mungkin hanya tertunda — Jika fungsi belum dipanggil atau data belum dibilas, Anda tidak akan melihat metrik di. CloudWatch Untuk informasi selengkapnya, silakan lihat Masalah yang diketahui selanjutnya di bagian ini.

- Konfirmasikan bahwa fungsi Lambda memiliki izin yang benar —Pastikan bahwa kebijakan CloudWatchLambdaInsightsExecutionRolePolicyIAM ditetapkan ke peran eksekusi fungsi.
- Periksa runtime Lambda—Wawasan Lambda hanya mendukung runtime Lambda tertentu. Untuk daftar runtime yang didukung, silakan lihat Wawasan Lambda.

  Misalnya, untuk menggunakan Wawasan Lambda di Java 8, Anda harus menggunakan runtime `java8.al2`, bukan runtime `java8`.
- Periksa akses jaringan —Fungsi Lambda mungkin ada di subnet pribadi VPC tanpa akses internet dan Anda tidak memiliki titik akhir VPC yang dikonfigurasi untuk Log. CloudWatch Untuk membantu debug masalah ini, Anda dapat mengatur variabel lingkungan `LAMBDA_INSIGHTS_LOG_LEVEL=info`.

Masalah yang diketahui

Penundaan data bisa mencapai 20 menit. Ketika pengatur fungsi menyelesaikannya, Lambda membekukan sandbox, yang juga akan membekukan ekstensi Wawasan Lambda. Ketika fungsi berjalan, kami menggunakan strategi pengelompokan adaptif didasarkan pada fungsi TPS untuk mengeluarkan data. Namun demikian, jika fungsi berhenti di-invoke untuk periode yang diperpanjang dan masih ada data event yang sedang diproses, data ini dapat ditunda hingga Lambda menghentikan sandbox yang tidak aktif. Ketika Lambda menghentikan sandbox, kami menghapus data yang diproses.

Contoh event telemetri

Setiap invokasi dari sebuah fungsi Lambda yang memiliki Wawasan Lambda yang diaktifkan akan menyebabkan dituliskannya sebuah peristiwa log ke grup log `/aws/lambda-insights`. Setiap peristiwa log berisikan metrik dalam format metrik tersemat. Untuk informasi selengkapnya tentang format metrik tersemat, silakan lihat Menyematkan metrik dalam log.

Untuk menganalisis peristiwa log ini, Anda dapat menggunakan metode berikut:

- Bagian Lambda Insights dari CloudWatch konsol, seperti yang dijelaskan di. Tinjau metrik Wawasan Lambda Anda
- Kueri peristiwa log menggunakan Wawasan CloudWatch Log. Untuk informasi selengkapnya, lihat Menganalisis Data Log dengan Wawasan CloudWatch Log.
- Metrik dikumpulkan di `LambdaInsights` namespace, yang Anda buat grafik menggunakan metrik. CloudWatch

Berikut adalah contoh peristiwa log Wawasan Lambda dengan format metrik tersemat.

```
{
    "_aws": {
        "Timestamp": 1605034324256,
        "CloudWatchMetrics": [
            {
                "Namespace": "LambdaInsights",
                "Dimensions": [
                    [ "function_name" ],
                    [ "function_name", "version" ]
                ],
                "Metrics": [
                    { "Name": "memory_utilization", "Unit": "Percent" },
                    { "Name": "total_memory", "Unit": "Megabytes" },
                    { "Name": "used_memory_max", "Unit": "Megabytes" },
                    { "Name": "cpu_total_time", "Unit": "Milliseconds" },
                    { "Name": "tx_bytes", "Unit": "Bytes" },
                    { "Name": "rx_bytes", "Unit": "Bytes" },
                    { "Name": "total_network", "Unit": "Bytes" },
                    { "Name": "init_duration", "Unit": "Milliseconds" }
                ]
            }
        ],
        "LambdaInsights": {
            "ShareTelemetry": true
        }
    },
    "event_type": "performance",
    "function_name": "cpu-intensive",
    "version": "Blue",
    "request_id": "12345678-8bcc-42f7-b1de-123456789012",
    "trace_id": "1-5faae118-12345678901234567890",
    "duration": 45191,
    "billed_duration": 45200,
    "billed_mb_ms": 11571200,
    "cold_start": true,
    "init_duration": 130,
    "tmp_free": 538329088,
    "tmp_max": 551346176,
    "threads_max": 11,
    "used_memory_max": 63,
    "total_memory": 256,
    "memory_utilization": 24,
```

```
    "cpu_user_time": 6640,
    "cpu_system_time": 50,
    "cpu_total_time": 6690,
    "fd_use": 416,
    "fd_max": 32642,
    "tx_bytes": 4434,
    "rx_bytes": 6911,
    "timeout": true,
    "shutdown_reason": "Timeout",
    "total_network": 11345,
    "agent_version": "1.0.72.0",
    "agent_memory_avg": 10,
    "agent_memory_max": 10
}
```


Gunakan Contributor Insights untuk menganalisis data kardinalitas tinggi

Anda dapat menggunakan Wawasan Kontributor untuk menganalisis data log dan membuat rangkaian waktu yang menampilkan data kontributor. Anda dapat melihat metrik tentang kontributor-kontributor N teratas, total kontributor unik, dan penggunaannya. Hal ini membantu Anda menemukan pembicara terbaik dan memahami orang atau objek yang memengaruhi performa sistem. Contohnya, Anda dapat menemukan host yang buruk, mengidentifikasi pengguna jaringan yang paling berat, atau menemukan URL yang menghasilkan sebagian besar kesalahan.

Anda dapat membangun aturan Anda dari awal, dan ketika Anda menggunakan AWS Management Console Anda juga dapat menggunakan aturan sampel yang AWS telah dibuat. Aturan menetapkan bidang log yang ingin Anda gunakan untuk menentukan kontributor, seperti `IpAddress`. Anda juga dapat menyaring data log untuk menemukan dan menganalisis perilaku kontributor individual.

CloudWatch juga menyediakan aturan bawaan yang dapat Anda gunakan untuk menganalisis metrik dari AWS layanan lain.

Semua aturan menganalisis data masuk dalam waktu nyata.

Jika Anda masuk ke akun yang disiapkan sebagai akun pemantauan dalam pengamatan CloudWatch lintas akun, Anda dapat membuat aturan Wawasan Kontributor di akun pemantauan yang menganalisis grup log di akun sumber dan di akun pemantauan. Anda juga dapat membuat aturan tunggal yang menganalisis grup log di beberapa akun. Untuk informasi selengkapnya, lihat CloudWatch observabilitas lintas akun.

> **Note**
>
> Jika menggunakan Wawasan Kontributor, Anda akan dikenakan biaya untuk setiap peristiwa log yang sesuai dengan aturan. Untuk informasi selengkapnya, lihat CloudWatch Harga Amazon.

Topik

Membuat Aturan Wawasan Kontributor

Anda dapat membuat aturan untuk menganalisis data log. Setiap log dalam JSON atau Format Catatan Umum (CLF) dapat dievaluasi. Ini termasuk log kustom Anda yang mengikuti salah satu format dan log ini dari AWS layanan seperti log aliran Amazon VPC, log kueri DNS Amazon Route 53, log penampung Amazon ECS, dan log dari, AWS CloudTrail Amazon, SageMaker Amazon RDS, dan API Gateway. AWS AppSync

Pada aturan, ketika Anda menentukan nama atau nilai bidang, semua pencocokan peka huruf besar dan kecil.

Anda dapat menggunakan aturan sampel bawaan ketika membuat aturan atau Anda dapat membuat aturan sendiri dari awal. Wawasan Kontributor mencakup aturan sampel untuk jenis log berikut:

- Log Amazon API Gateway
- Log kueri DNS publik Amazon Route 53
- Log kueri Amazon Route 53 Resolver
- CloudWatch Log Wawasan Kontainer
- Log alur VPC

Jika Anda masuk ke akun yang disiapkan sebagai akun pemantauan dalam pengamatan CloudWatch lintas akun, Anda dapat membuat aturan Wawasan Kontributor untuk grup log di akun sumber yang ditautkan ke akun pemantauan ini, selain membuat aturan untuk grup log di akun pemantauan. Anda juga dapat membuat aturan tunggal yang memantau grup log di akun yang berbeda. Untuk informasi selengkapnya, lihat CloudWatch observabilitas lintas akun.

> **Important**
>
> Saat Anda memberikan `cloudwatch:PutInsightRule` izin kepada pengguna, secara default pengguna tersebut dapat membuat aturan yang mengevaluasi grup log apa pun di CloudWatch Log. Anda dapat menambahkan ketentuan kebijakan IAM yang membatasi izin ini agar pengguna dapat menyertakan dan mengecualikan grup log tertentu. Untuk informasi selengkapnya, lihat Menggunakan kunci syarat untuk membatasi akses pengguna Wawasan Kontributor ke grup log.

Untuk membuat aturan menggunakan aturan sampel bawaan

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Wawasan Wawasan Kontributor.
3. Pilih Buat aturan.

4. Untuk Pilih grup log, pilih satu grup log (atau beberapa grup log) yang Anda ingin pantau aturannya. Anda dapat memilih sebanyak 20 grup log. Jika Anda masuk ke akun pemantauan yang disiapkan untuk pengamatan CloudWatch lintas akun, Anda dapat memilih grup log di akun sumber, dan Anda juga dapat mengatur satu aturan untuk menganalisis grup log di akun yang berbeda.

   - (Opsional) Untuk memilih semua grup log yang memiliki nama yang dimulai dengan string tertentu, pilih Pilih berdasarkan kecocokan awalan, lalu masukkan awalan. Jika ini adalah akun pemantauan, Anda dapat memilih akun yang akan dicari secara opsional, jika tidak semua akun dipilih.

> **Note**
>
> Anda akan dikenakan biaya untuk setiap peristiwa log yang sesuai dengan aturan Anda. Jika Anda memilih menu geser-turun Pilih berdasarkan kecocokan awalan, maka

Anda harus mengetahui seberapa banyak grup log yang dapat dicocokkan oleh awalan tersebut. Jika Anda mencari lebih banyak grup log dari yang Anda inginkan, maka Anda mungkin akan dikenakan biaya tak terduga. Untuk informasi selengkapnya, lihat CloudWatch Harga Amazon.

5. Untuk jenis Aturan, pilih Aturan sampel. Kemudian pilih Pilih aturan sampel dan pilih aturan.
6. Aturan sampel telah mengisi format Log, Kontribusi, Filter, dan Agregat pada bidang. Anda dapat menyesuaikan nilai tersebut, jika Anda mau.
7. Pilih Berikutnya.
8. Untuk Nama aturan, masukkan nama. Karakter yang valid adalah A-Z, a-z, 0-9, (tanda hubung), (garis bawah), dan (titik).
9. Pilih apakah membuat aturan dalam status dinonaktifkan atau diaktifkan. Jika Anda memilih untuk mengaktifkannya, aturan segera mulai menganalisis data Anda. Anda mengeluarkan biaya ketika menjalankan aturan yang diaktifkan. Untuk informasi selengkapnya, lihat Penentuan Harga CloudWatch Amazon.

   Wawasan Kontributor hanya menganalisis peristiwa log baru setelah aturan dibuat. Aturan tidak dapat memproses peristiwa log yang sebelumnya diproses oleh CloudWatch Log.
10. (Opsional) Untuk Tag, tambahkan satu atau beberapa pasangan nilai kunci sebagai tag untuk aturan ini. Tag dapat membantu Anda mengidentifikasi dan mengatur AWS sumber daya Anda dan melacak AWS biaya Anda. Untuk informasi selengkapnya, lihat Pemberian tag pada sumber daya Amazon CloudWatch Anda.
11. Pilih Buat.

Untuk membuat aturan dari awal

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Wawasan Kontributor.
3. Pilih Buat aturan.

4. Untuk Pilih grup log, pilih satu grup log (atau beberapa grup log) yang Anda ingin pantau aturannya. Anda dapat memilih sebanyak 20 grup log. Jika Anda masuk ke akun pemantauan yang disiapkan untuk pengamatan CloudWatch lintas akun, Anda dapat memilih grup log di akun sumber, dan Anda juga dapat mengatur satu aturan untuk menganalisis grup log di akun yang berbeda.

- (Opsional) Untuk memilih semua grup log yang memiliki nama yang dimulai dengan string tertentu, pilih Pilih berdasarkan kecocokan awalan, lalu masukkan awalan.

> **Note**
>
> Anda akan dikenakan biaya untuk setiap peristiwa log yang sesuai dengan aturan Anda. Jika Anda memilih menu geser-turun Pilih berdasarkan kecocokan awalan, maka Anda harus mengetahui seberapa banyak grup log yang dapat dicocokkan oleh awalan tersebut. Jika Anda mencari lebih banyak grup log dari yang Anda inginkan, maka Anda mungkin akan dikenakan biaya tak terduga. Untuk informasi selengkapnya, lihat CloudWatch Harga Amazon.

5. Untuk Jenis aturan, pilih Aturan kustom.
6. Untuk Format log, pilih JSON atau CLF.
7. Anda dapat menyelesaikan pembuatan aturan dengan menggunakan pemandu atau dengan memilih tab Sintaks dan menentukan sintaksis aturan secara manual.

   Untuk melanjutkan menggunakan pemandu, lakukan hal berikut:

   a. Untuk Kontribusi, Kunci, masukkan jenis kontributor yang ingin Anda laporkan. Laporan menampilkan nilai N teratas untuk jenis kontributor ini.

      Catatan yang benar adalah setiap bidang log yang memiliki nilai. Contohnya termasuk **`requestId`**, **`sourceIPaddress`**, dan **`containerID`**.

      Untuk informasi tentang temuan nama bidang log untuk log di grup log tertentu, silakan lihat Kolom log temuan.

      Kunci yang lebih besar dari 1 KB dipotong menjadi 1KB.

   b. (Opsional) Pilih Tambahkan tombol baru untuk menambahkan lebih banyak tombol. Anda dapat memasukkan sebanyak mungkin empat kunci dalam aturan. Jika Anda memasukkan lebih dari satu kunci, kontributor dalam laporan ditentukan oleh kombinasi nilai unik kunci. Sebagai contoh, jika Anda menentukan tiga kunci, setiap kombinasi nilai unik untuk tiga kunci tersebut dihitung sebagai kontributor unik.

   c. (Opsional) Jika Anda ingin menambahkan filter yang mempersempit cakupan hasil Anda, pilih Tambahkan filter. Untuk Kecocokan, masukkan nama bidang log yang ingin Anda

saring. Untuk Kondisi, pilih operator perbandingan, dan masukkan nilai yang ingin Anda saring.

Anda dapat menambahkan sebanyak mungkin empat filter dalam aturan. Beberapa penyaring digabungkan oleh logika DAN, sehingga hanya peristiwa log yang cocok dengan semua penyaring yang dievaluasi.

> **Note**
>
> Array yang mengikuti operator perbandingan, sepertiIn, NotIn, atauStartsWith, dapat mencakup sebanyak 10 nilai string. Untuk informasi selengkapnya tentang sintaks aturan Wawasan Kontributor, silakan lihat Sintaks Aturan Wawasan Kontributor.

d. Untuk melakukan Agregat, pilih Hitung atau Jumlah. Memilih Hitung menyebabkan peringkat kontributor didasarkan pada jumlah kejadian. Memilih Jumlah menyebabkan peringkat didasarkan pada jumlah agregat dari nilai bidang yang Anda tentukan untuk Kontribusi, Nilai.

8. Untuk memasukkan aturan Anda sebagai objek JSON alih-alih menggunakan pemandu, lakukan hal berikut:

   a. Pilih tab Sintaks.

   b. Di Badan aturan, masukkan objek JSON untuk aturan Anda. Untuk informasi tentang sintaks aturan, silakan lihat Sintaks Aturan Wawasan Kontributor.

9. Pilih Berikutnya.

10. Untuk Nama aturan, masukkan nama. Karakter yang benar adalah A-Z, a-z, 0-9, "-", "_', dan ".".

11. Pilih apakah membuat aturan dalam status dinonaktifkan atau diaktifkan. Jika Anda memilih untuk mengaktifkannya, aturan segera mulai menganalisis data Anda. Anda mengeluarkan biaya ketika menjalankan aturan yang diaktifkan. Untuk informasi selengkapnya, lihat Penentuan Harga CloudWatch Amazon.

    Wawasan Kontributor hanya menganalisis peristiwa log baru setelah aturan dibuat. Aturan tidak dapat memproses peristiwa log yang sebelumnya diproses oleh CloudWatch Log.

12. (Opsional) Untuk Tag, tambahkan satu atau beberapa pasangan nilai kunci sebagai tag untuk aturan ini. Tag dapat membantu Anda mengidentifikasi dan mengatur AWS sumber daya Anda dan melacak AWS biaya Anda. Untuk informasi selengkapnya, lihat Pemberian tag pada sumber daya Amazon CloudWatch Anda.

13. Pilih Berikutnya.

14. Konfirmasikan pengaturan yang Anda masukkan, dan pilih Buat aturan.

Anda dapat menonaktifkan, mengaktifkan, atau menghapus aturan yang telah Anda buat.

Untuk mengaktifkan, menonaktifkan, atau menghapus aturan di Wawasan Kontributor

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Wawasan Kontributor.
3. Dalam daftar aturan, pilih kotak centang di samping aturan tunggal.

   Aturan bawaan dibuat oleh AWS layanan dan tidak dapat diedit, dinonaktifkan, atau dihapus.
4. Pilih Tindakan, dan kemudian pilih pilihan yang Anda inginkan.

Menemukan bidang log

Ketika membuat aturan, Anda perlu mengetahui nama bidang pada pencatatan log dalam grup log.

Cara menemukan bidang log di sebuah grup log

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Dalam panel navigasi, di bawah Log, pilih Wawasan.
3. Di atas penyunting pencarian, pilih satu atau beberapa grup log untuk bertanya.

   Saat Anda memilih grup CloudWatch log, Wawasan Log secara otomatis mendeteksi bidang dalam data dalam grup log dan menampilkannya di panel kanan di bidang Ditemukan.

Sintaks Aturan Wawasan Kontributor

Bagian ini menjelaskan aturan sintaksis untuk Wawasan Kontributor. Gunakan sintaks ini hanya ketika Anda membuat aturan dengan memasukkan blok JSON. Jika Anda menggunakan pemandu untuk membuat aturan, Anda tidak perlu mengetahui sintaksnya. Untuk informasi selengkapnya tentang membuat aturan menggunakan pemandu, silakan lihat Membuat Aturan Wawasan Kontributor.

Semua pencocokan aturan untuk peristiwa log dan nilai bidang peristiwa peka huruf besar dan kecil.

Contoh berikut mengilustrasikan sintaks untuk log JSON.

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "LogGroupNames": [
        "API-Gateway-Access-Logs*",
        "Log-group-name2"
    ],
    "LogFormat": "JSON",
    "Contribution": {
        "Keys": [
            "$.ip"
        ],
        "ValueOf": "$.requestBytes",
        "Filters": [
            {
                "Match": "$.httpMethod",
                "In": [
                    "PUT"
                ]
            }
        ]
    },
    "AggregateOn": "Sum"
}
```

Bidang dalam Aturan Wawasan Kontributor

Skema

Nilai `Schema` untuk aturan yang menganalisis data CloudWatch Log harus selalu `{"Name": "CloudWatchLogRule", "Version": 1}`

LogGroupNames

Sebuah susunan string. Untuk setiap elemen dalam susunan, Anda dapat secara opsional menggunakan * di akhir rangkaian untuk menyertakan semua grup log dengan nama yang dimulai dengan awalan itu.

Berhati-hatilah menggunakan karakter pengganti dengan nama grup log. Anda dikenakan biaya untuk setiap peristiwa log yang sesuai dengan aturan. Jika secara tidak sengaja mencari lebih

banyak grup log dari yang Anda inginkan, Anda mungkin akan dikenakan biaya tak terduga. Untuk informasi selengkapnya, lihat CloudWatch Harga Amazon.

LogGroupARN

Jika Anda membuat aturan ini di akun pemantauan observabilitas CloudWatch lintas akun, Anda dapat menggunakannya `LogGroupARNs` untuk menentukan grup log di akun sumber yang ditautkan ke akun pemantauan, dan untuk menentukan grup log di akun pemantauan itu sendiri. Anda harus menentukan salah satu `LogGroupNames` atau `LogGroupARNs` dalam aturan Anda, tetapi tidak keduanya.

`LogGroupARNs` adalah sebuah susunan string. Untuk setiap elemen dalam susunan, Anda dapat secara opsional menggunakan * sebagai wildcard dalam situasi tertentu. Misalnya Anda dapat menentukan `arn:aws:logs:us-west-1:*:log-group/MyLogGroupName2` grup log yang diberi nama `MyLogGroupName2` di semua akun sumber dan di akun pemantauan, di Wilayah AS Barat (California Utara). Anda juga dapat menentukan `arn:aws:logs:us-west-1:111122223333:log-group/GroupNamePrefix*` untuk menentukan semua grup log di AS Barat (California Utara) di 111122223333 yang memiliki nama yang dimulai dengan`GroupNamePrefix`.

Anda tidak dapat menentukan sebagian ID AWS akun sebagai awalan dengan kartu liar.

Berhati-hatilah menggunakan karakter pengganti dengan ARN grup log. Anda dikenakan biaya untuk setiap peristiwa log yang sesuai dengan aturan. Jika secara tidak sengaja mencari lebih banyak grup log dari yang Anda inginkan, Anda mungkin akan dikenakan biaya tak terduga. Untuk informasi selengkapnya, lihat CloudWatch Harga Amazon.

LogFormat

Nilai yang valid adalah `JSON` dan `CLF`.

Kontribusi

Objek ini mencakup `Keys` susunan dengan sebanyak empat anggota, secara opsional satu `ValueOf`, dan secara opsional susunan sebanyak empat `Filters`.

Kunci

Susunan hingga empat bidang log yang digunakan sebagai dimensi untuk mengklasifikasi kontributor. Jika Anda memasukkan lebih dari satu kunci, setiap kombinasi nilai unik untuk kunci tersebut dihitung sebagai kontributor unik. Bidang harus ditentukan menggunakan notasi format properti JSON.

ValueOf

(Opsional) Tentukan ini hanya ketika Anda menentukan `Sum` sebagai nilai dari `AggregateOn`. `ValueOf` menentukan bidang log dengan nilai numerik. Dalam jenis aturan ini, kontributor diberikan peringkat berdasarkan jumlah nilai bidang ini, bukan jumlah kejadiannya dalam pencatatan log. Sebagai contoh, jika ingin menyortir kontributor berdasarkan total `BytesSent` selama satu periode, Anda akan menentukan `ValueOf` untuk `BytesSent` dan menentukan `Sum` untuk `AggregateOn`.

Filter

(Opsional) Menentukan susunan sebanyak empat penyaring untuk mempersempit peristiwa log yang disertakan dalam laporan. Jika Anda menetapkan beberapa penyaring, Wawasan Kontributor mengevaluasinya dengan operator DAN logis. Anda dapat menggunakan ini untuk menyaring peristiwa log yang tidak relevan dalam pencarian atau Anda dapat menggunakannya untuk memilih kontributor tunggal untuk menganalisis perilaku mereka.

Setiap anggota dalam susunan harus menyertakan `Match` bidang dan bidang yang menunjukkan jenis operator yang cocok untuk digunakan.

Bidang `Match` menetapkan bidang log untuk mengevaluasi di penyaring. Bidang log ditentukan menggunakan notasi format properti JSON.

Bidang operator yang cocok harus salah satu dari berikut: `In`, `NotIn`, `StartsWith`, `GreaterThan`, `LessThan`, `EqualTo`, `NotEqualTo`, atau `IsPresent`. Bilang bidang operator adalah `In`, `NotIn`, atau `StartsWith`, itu diikuti dengan susunan rangkaian nilai untuk diperiksa. Wawasan Kontributor mengevaluasi susunan nilai rangkaian dengan operator OR. Susunan dapat mencakup sebanyak 10 nilai rangkaian.

Jika bidang operator adalah `GreaterThan`, `LessThan`, `EqualTo`, atau `NotEqualTo`, diikuti dengan nilai numerik tunggal untuk dibandingkan.

Jika bidang operator adalah `IsPresent`, itu diikuti oleh `true` atau `false`. Operator ini mencocokkan log acara berdasarkan pada apakah bidang log yang ditentukan ada di dalam peristiwa log. `isPresent` hanya bekerja dengan nilai di simpul daun properti JSON. Misalnya, penyaring yang mencari kecocokan dengan `c-count` tidak mengevaluasi peristiwa log dengan nilai `details.c-count.c1`.

Lihat empat contoh penyaring berikut:

```
{"Match": "$.httpMethod", "In": [ "PUT", ] }
```

```
{"Match": "$.StatusCode", "EqualTo": 200 }
{"Match": "$.BytesReceived", "GreaterThan": 10000}
{"Match": "$.eventSource", "StartsWith": [ "ec2", "ecs" ] }
```

AggregateOn

Nilai yang valid adalah `Count` dan `Sum`. Menentukan apakah akan menggabungkan laporan berdasarkan pada hitungan kejadian atau jumlah dari nilai bidang yang ditentukan dalam bidang `ValueOf`.

Rotasi format properti JSON

Bidang `Keys`, `ValueOf`, dan `Match` mengikuti format properti JSON dengan notasi titik, ketika $ mewakili akar dari objek JSON. Hal ini diikuti dengan periode dan kemudian rangkaian alfanumerik dengan nama subproperti. Beberapa tingkat properti didukung.

Karakter pertama dari string hanya bisa A-Z atau a-z. Karakter string berikut ini dapat berupa A-Z, a-z, atau 0-9.

Daftar berikut menggambarkan contoh format properti JSON yang benar:

```
$.userAgent
$.endpoints[0]
$.users[1].name
$.requestParameters.instanceId
```

Bidang tambahan di dalam aturan untuk log CLF

Peristiwa log Format Log Umum (CLF) tidak memiliki nama untuk bidang seperti JSON. Untuk menyediakan bidang yang digunakan untuk aturan Wawasan Kontributor, peristiwa log CLF dapat diperlakukan sebagai susunan dengan indeks mulai dari 1. Anda dapat menentukan bidang pertama sebagai **"1"**, bidang kedua sebagai **"2"**, dan seterusnya.

Untuk membuat aturan untuk log CLF lebih mudah dibaca, Anda dapat menggunakan `Fields`. Ini memungkinkan Anda untuk memberikan nama alias untuk lokasi bidang CLF. Misalnya, Anda dapat menentukan bahwa lokasi "4" adalah alamat IP. Setelah ditentukan, `IpAddress` dapat digunakan sebagai properti di `Keys`, `ValueOf`, dan `Filters` dalam aturan.

Berikut adalah contoh aturan untuk log CLF yang menggunakan bidang `Fields`.

```
{
```

```
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "LogGroupNames": [
        "API-Gateway-Access-Logs*"
    ],
    "LogFormat": "CLF",
    "Fields": {
        "4": "IpAddress",
        "7": "StatusCode"
    },
    "Contribution": {
        "Keys": [
            "IpAddress"
        ],
        "Filters": [
            {
                "Match": "StatusCode",
                "EqualTo": 200
            }
        ]
    },
    "AggregateOn": "Count"
}
```

Contoh Aturan Wawasan Kontributor

Bagian ini memuat contoh yang mengilustrasikan kasus penggunaan untuk aturan Wawasan Kontributor.

Log Alur VPC: Byte transfer berdasarkan alamat IP sumber dan tujuan

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "LogGroupNames": [
        "/aws/containerinsights/sample-cluster-name/flowlogs"
    ],
    "LogFormat": "CLF",
    "Fields": {
```

```
        "4": "srcaddr",
        "5": "dstaddr",
        "10": "bytes"
    },
    "Contribution": {
        "Keys": [
            "srcaddr",
            "dstaddr"
        ],
        "ValueOf": "bytes",
        "Filters": []
    },
    "AggregateOn": "Sum"
}
```

Log Alur VPC: Jumlah tertinggi dari permintaan HTTPS

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "LogGroupNames": [
        "/aws/containerinsights/sample-cluster-name/flowlogs"
    ],
    "LogFormat": "CLF",
    "Fields": {
        "5": "destination address",
        "7": "destination port",
        "9": "packet count"
    },
    "Contribution": {
        "Keys": [
            "destination address"
        ],
        "ValueOf": "packet count",
        "Filters": [
            {
                "Match": "destination port",
                "EqualTo": 443
            }
        ]
    },
```

```
    "AggregateOn": "Sum"
}
```

Log Alur VPC: koneksi TCP yang ditolak

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "LogGroupNames": [
        "/aws/containerinsights/sample-cluster-name/flowlogs"
    ],
    "LogFormat": "CLF",
    "Fields": {
        "3": "interfaceID",
        "4": "sourceAddress",
        "8": "protocol",
        "13": "action"
    },
    "Contribution": {
        "Keys": [
            "interfaceID",
            "sourceAddress"
        ],
        "Filters": [
            {
                "Match": "protocol",
                "EqualTo": 6
            },
            {
                "Match": "action",
                "In": [
                    "REJECT"
                ]
            }
        ]
    },
    "AggregateOn": "Sum"
}
```

Route 53 tanggapan NXDomain dari alamat sumber

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "AggregateOn": "Count",
    "Contribution": {
        "Filters": [
            {
                "Match": "$.rcode",
                "StartsWith": [
                    "NXDOMAIN"
                ]
            }
        ],
        "Keys": [
            "$.srcaddr"
        ]
    },
    "LogFormat": "JSON",
    "LogGroupNames": [
        "<loggroupname>"
    ]
}
```

Kirim 53 pertanyaan penyelesaian berdasarkan nama domain

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "AggregateOn": "Count",
    "Contribution": {
        "Filters": [],
        "Keys": [
            "$.query_name"
        ]
    },
    "LogFormat": "JSON",
    "LogGroupNames": [
        "<loggroupname>"
    ]
```

```
}
```

Kirim 53 pertanyaan penyelesaian berdasarkan jenis pertanyaan dan alamat sumber

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "AggregateOn": "Count",
    "Contribution": {
        "Filters": [],
        "Keys": [
            "$.query_type",
            "$.srcaddr"
        ]
    },
    "LogFormat": "JSON",
    "LogGroupNames": [
        "<loggroupname>"
    ]
}
```

Melihat Laporan Wawasan Kontributor

Untuk melihat grafik data laporan dan daftar kontributor berperingkat yang ditemukan oleh aturan Anda, ikuti langkah-langkah ini.

Untuk melihat laporan aturan Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Wawasan Kontributor.
3. Pada daftar aturan, pilih nama aturan.

   Grafik menampilkan hasil aturan pada tiga jam terakhir. Tabel di bawah grafik menunjukkan 10 kontributor teratas.
4. Untuk mengubah jumlah kontributor yang ditampilkan di tabel, pilih 10 kontributor teratas di bagian atas grafik.

5. Untuk menyaring grafik agar hanya menampilkan hasil dari kontributor tunggal, pilih kontributor tersebut dalam keterangan tabel. Untuk menampilkan kembali semua kontributor, pilih kontributor yang sama lagi di keterangan.
6. Untuk mengubah rentang waktu yang ditampilkan dalam laporan, pilih 15 m, 30m, 1 jam, 2 jam, 3 jam, atau khusus di bagian atas grafik.

   Rentang waktu maksimal untuk laporan adalah 24 jam, tetapi Anda dapat memilih jendela 24 jam yang terjadi hingga 15 hari yang lalu. Untuk memilih jendela waktu di masa lalu, pilih khusus, mutlak, dan kemudian tentukan jendela waktu Anda.
7. Untuk mengubah durasi periode waktu yang digunakan untuk gabungan dan peringkat kontributor, pilih periode di bagian atas grafik. Melihat periode waktu yang lebih lama umumnya menunjukkan laporan yang lebih lancar dengan beberapa lonjakan. Memilih periode waktu yang lebih singkat lebih mungkin untuk menampilkan lonjakan.
8. Untuk menambahkan grafik ini ke CloudWatch dasbor, pilih Tambahkan ke dasbor.
9. Untuk membuka jendela kueri Wawasan Log, dengan grup log dalam laporan ini sudah dimuat di kotak kueri, pilih Lihat log. CloudWatch
10. Untuk mengekspor data laporan ke clipboard atau file CSV, pilih Ekspor.

Membuat grafik metrik-metrik yang dihasilkan oleh aturan

Wawasan Kontributor menyediakan sebuah fungsi matematika metrik `INSIGHT_RULE_METRIC`. Anda dapat menggunakan fungsi ini untuk menambahkan data dari laporan Contributor Insights ke grafik di tab Metrik konsol. CloudWatch Anda juga dapat mengatur alarm didasarkan pada fungsi matematika ini. Untuk informasi selengkapnya tentang fungsi matematika metrik, silakan lihat Gunakan matematika metrik

Untuk menggunakan fungsi matematika metrik ini, Anda harus masuk ke akun yang memiliki `cloudwatch:GetMetricData` dan `cloudwatch:GetInsightRuleReport` izin.

Sintaks adalah `INSIGHT_RULE_METRIC(`*`ruleName`*`, `*`metricName`*`)`. *`ruleName`* adalah nama dari aturan Wawasan Kontributor. *`metricName`* adalah satu dari nilai di dalam daftar berikut. Nilai dari *`metricName`* menentukan jenis data yang dikembalikan oleh fungsi matematika.

- `UniqueContributors` — jumlah kontributor unik untuk setiap titik data.
- `MaxContributorValue` — nilai kontributor teratas untuk setiap titik data. Identitas kontributor mungkin berubah untuk setiap titik data dalam grafik.

Jika aturan ini diagregat dengan `Count`, kontributor teratas untuk setiap titik data adalah kontributor dengan kemunculan paling banyak dalam periode tersebut. Jika aturan diagregat dengan `Sum`, kontributor teratas adalah kontributor dengan jumlah terbesar di dalam bidang log yang ditentukan oleh aturan `Value` selama periode tersebut.

- `SampleCount` — jumlah titik data yang sesuai dengan aturan.
- `Sum` — jumlah nilai dari semua kontributor selama periode waktu yang diwakili oleh titik data tersebut.
- `Minimum` — nilai minimum dari satu observasi selama periode waktu yang diwakili oleh titik data tersebut.
- `Maximum` — nilai maksimal dari satu observasi selama periode waktu yang diwakili oleh titik data tersebut.
- `Average` — nilai rata-rata dari semua kontributor selama periode waktu yang diwakili oleh titik data tersebut.

Mengatur Alarm pada Data Metrik Wawasan Kontributor

Dengan menggunakan fungsi `INSIGHT_RULE_METRIC` ini, Anda akan dapat menyetel alarm pada metrik yang dihasilkan oleh Wawasan Kontributor. Sebagai contoh, Anda dapat membuat alarm berdasarkan persentase koneksi protokol kontrol transmisi (TCP) yang ditolak. Untuk memulai dengan jenis alarm ini, Anda dapat membuat aturan-aturan seperti yang ditunjukkan dalam dua contoh berikut:

Contoh aturan: "RejectedConnectionsRule"

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "LogGroupNames": [
        "/aws/containerinsights/sample-cluster-name/flowlogs"
    ],
    "LogFormat": "CLF",
    "Fields": {
        "3": "interfaceID",
        "4": "sourceAddress",
        "8": "protocol",
```

```
        "13": "action"
    },
    "Contribution": {
        "Keys": [
            "interfaceID",
            "sourceAddress"
        ],
        "Filters": [
            {
                "Match": "protocol",
                "EqualTo": 6
            },
            {
                "Match": "action",
                "In": [
                    "REJECT"
                ]
            }
        ]
    },
    "AggregateOn": "Sum"
}
```

Contoh aturan: "TotalConnectionsRule"

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "LogGroupNames": [
        "/aws/containerinsights/sample-cluster-name/flowlogs"
    ],
    "LogFormat": "CLF",
    "Fields": {
        "3": "interfaceID",
        "4": "sourceAddress",
        "8": "protocol",
        "13": "action"
    },
    "Contribution": {
        "Keys": [
            "interfaceID",
```

```
            "sourceAddress"
        ],
        "Filters": [
            {
                "Match": "protocol",
                "EqualTo": 6
            }
    "AggregateOn": "Sum"
}
```

Setelah membuat aturan, Anda dapat memilih tab Metrik di CloudWatch Konsol, di mana Anda dapat menggunakan contoh ekspresi matematika metrik berikut untuk membuat grafik data yang dilaporkan Contributor Insights:

Contoh: Ekspresi matematika metrik

```
e1 INSIGHT_RULE_METRIC("RejectedConnectionsRule", "Sum")
e2 INSIGHT_RULE_METRIC("TotalConnectionsRule", "Sum")
e3 (e1/e2)*100
```

Dalam contoh, ekspresi matematika metrik `e3` akan mengembalikan semua koneksi TCP yang ditolak. Jika Anda ingin diberi tahu ketika 20 persen koneksi TCP ditolak, Anda dapat mengubah ekspresi dengan mengubah ambang batas dari `100` menjadi `20`.

> **Note**
>
> Anda dapat mengatur alarm pada metrik yang Anda pantau dari bagian Metrik. Saat berada di tab Metrik bergrafik, Anda dapat memilih ikon Buat alarm di bawah kolom Tindakan. Ikon Buat alarm terlihat seperti bel.

Untuk informasi selengkapnya tentang membuat grafik metrik dan penggunaan fungsi matematika metrik, silakan lihat bagian berikut: Tambahkan ekspresi matematika ke CloudWatch grafik.

Menggunakan Aturan Bawaan Wawasan Kontributor

Anda dapat menggunakan aturan bawaan Contributor Insights untuk menganalisis metrik dari layanan lain. AWS Layanan berikut mendukung aturan-aturan bawaan:

- Wawasan Kontributor untuk Amazon DynamoDB di Panduan Pengembang Amazon DynamoDB.

- Gunakan aturan-aturan Wawasan Kontributor bawaan dalam Panduan AWS PrivateLink .

Wawasan CloudWatch Aplikasi Amazon

Amazon CloudWatch Application Insights memfasilitasi pengamatan untuk aplikasi Anda dan sumber daya yang mendasarinya AWS . Aplikasi ini akan dapat membantu Anda mengatur monitor terbaik untuk sumber daya aplikasi Anda agar dapat secara terus-menerus melakukan analisis data untuk melihat tanda-tanda dari munculnya masalah dengan aplikasi Anda. Application Insights, yang didukung oleh SageMakerdan AWS teknologi lainnya, menyediakan dasbor otomatis yang menunjukkan potensi masalah dengan aplikasi yang dipantau, yang membantu Anda mengisolasi masalah yang sedang berlangsung dengan cepat dengan aplikasi dan infrastruktur Anda. Peningkatan visibilitas ke dalam kondisi aplikasi Anda yang disediakan oleh Wawasan Aplikasi akan membantu mengurangi waktu rata-rata untuk melakukan perbaikan (MTTR) untuk memecahkan masalah yang terjadi pada aplikasi Anda.

Saat Anda menambahkan aplikasi ke Amazon CloudWatch Application Insights, aplikasi akan memindai sumber daya dalam aplikasi dan merekomendasikan serta mengonfigurasi metrik dan log on CloudWatchuntuk komponen aplikasi. Contoh komponen-komponen aplikasi meliputi basis data backend Server SQL dan tingkat Microsoft IIS/Web. Wawasan Aplikasi menganalisis pola-pola metrik dengan menggunakan data historis untuk mendeteksi anomali, dan mendeteksi kesalahan serta pengecualian dari aplikasi, sistem operasi, dan log infrastruktur Anda secara terus-menerus. Wawasan Aplikasi ini mengorelasikan observasi ini menggunakan sebuah kombinasi antara algoritma klasifikasi dan aturan bawaan. Kemudian, Wawasan Aplikasi akan secara otomatis membuat dasbor yang menampilkan observasi yang relevan dan informasi kepelikan masalah untuk membantu Anda menentukan prioritas tindakan. Untuk masalah-masalah umum yang terjadi dalam tumpukan aplikasi .NET dan SQL, seperti latensi aplikasi, kegagalan pencadangan Server SQL, kebocoran memori, permintaan HTTP yang besar, dan operasi I/O yang dibatalkan, Wawasan Aplikasi akan memberikan wawasan tambahan yang menunjukkan hal-hal yang mungkin menjadi akar penyebabnya, serta menunjukkan langkah-langkah penyelesaiannya. Integrasi bawaan dengan AWS SSM OpsCenter memungkinkan Anda menyelesaikan masalah dengan menjalankan dokumen Otomasi Systems Manager yang relevan.

Bagian-bagian

- Apa itu Wawasan CloudWatch Aplikasi Amazon?
- Cara kerja Amazon CloudWatch Application Insights
- Memulai dengan Amazon CloudWatch Application Insights

- Observabilitas lintas akun Wawasan Aplikasi
- Cara menggunakan konfigurasi komponen
- Membuat dan mengonfigurasi pemantauan Wawasan CloudWatch Aplikasi menggunakan templat CloudFormation
- Tutorial: Cara menyiapkan pemantauan untuk SAP ASE
- Tutorial: Cara menyiapkan pemantauan untuk SAP HANA
- Tutorial: Mengatur pemantauan untuk SAP NetWeaver
- Melihat dan memecahkan masalah yang terdeteksi oleh Amazon CloudWatch Application Insights
- Log dan metrik yang didukung oleh Amazon CloudWatch Application Insights

Apa itu Wawasan CloudWatch Aplikasi Amazon?

CloudWatch Application Insights membantu Anda memantau aplikasi yang menggunakan instans Amazon EC2 bersama dengan sumber daya aplikasi lainnya. Kemampuan ini mengidentifikasi dan menyiapkan metrik-metrik, log, dan alarm kunci di seluruh sumber daya aplikasi dan tumpukan teknologi (misalnya, basis data Microsoft SQL Server, server web (IIS) dan aplikasi, OS, penyeimbang beban, dan antrean). Wawasan Aplikasi CloudWatch ini terus memantau metrik dan log untuk mendeteksi serta menghubungkan anomali dan kesalahan. Ketika kesalahan dan anomali terdeteksi, Application Insights menghasilkan CloudWatch Peristiwa yang dapat Anda gunakan untuk mengatur pemberitahuan atau mengambil tindakan. Untuk membantu pemecahan masalah, Wawasan Aplikasi ini akan membuat dasbor otomatis untuk masalah yang terdeteksi, yang mencakup anomali metrik terkorelasi dan kesalahan log, bersama dengan wawasan tambahan untuk menunjukkan akar masalah yang mungkin menjadi penyebabnya. Dasbor otomatis tersebut membantu Anda mengambil tindakan perbaikan untuk menjaga aplikasi agar tetap dalam kondisi sehat dan mencegah dampaknya agar tidak berpengaruh terhadap pengguna akhir aplikasi Anda. Ini juga menciptakan OpsItems sehingga Anda dapat menyelesaikan masalah menggunakan AWS SSM OpsCenter.

Anda dapat mengonfigurasi penghitung penting, seperti Mirrored Write Transaction/sec, Recovery Queue Length, dan Transaction Delay, serta Windows Event Logs on. CloudWatch Ketika peristiwa failover atau masalah terjadi dengan beban kerja SQL HA Anda, seperti akses terbatas untuk menanyakan database target, CloudWatch Application Insights menyediakan wawasan otomatis.

CloudWatch Application Insights terintegrasi dengan AWS Launch Wizarduntuk memberikan pengalaman penyiapan pemantauan satu klik untuk menerapkan beban kerja SQL Server HA. AWSSaat Anda memilih opsi untuk mengatur pemantauan dan wawasan dengan Application Insights

di konsol Launch Wizard, CloudWatch Application Insights secara otomatis menyiapkan metrik, log, dan alarm yang relevan CloudWatch, dan mulai memantau beban kerja yang baru diterapkan. Anda dapat melihat wawasan otomatis dan masalah yang terdeteksi, bersama dengan kesehatan beban kerja SQL Server HA Anda, di konsol. CloudWatch

Daftar Isi

Fitur-fitur

Wawasan Aplikasi menyediakan fitur-fitur berikut.

Pengaturan otomatis monitor untuk sumber daya aplikasi

CloudWatch Application Insights mengurangi waktu yang diperlukan untuk mengatur pemantauan untuk aplikasi Anda. Ini dilakukan dengan memindai sumber daya aplikasi Anda, menyediakan daftar metrik dan log yang direkomendasikan yang dapat disesuaikan, dan mengaturnya CloudWatch untuk memberikan visibilitas yang diperlukan ke sumber daya aplikasi Anda, seperti Amazon EC2 dan Elastic Load Balancers (ELB). Wawasan Aplikasi ini juga mengatur alarm dinamis pada metrik-metrik yang dipantau. Alarm-alarm tersebut akan secara otomatis diperbarui berdasarkan pada anomali-anomali yang terdeteksi dua minggu sebelumnya.

Deteksi dan notifikasi masalah

CloudWatch Application Insights mendeteksi tanda-tanda potensi masalah dengan aplikasi Anda, seperti anomali metrik dan kesalahan log. Wawasan Aplikasi ini mengorelasikan observasi ini untuk mengungkapkan potensi masalah yang mungkin terjadi pada aplikasi Anda. Kemudian menghasilkan CloudWatch Acara, yang dapat dikonfigurasi untuk menerima pemberitahuan atau mengambil tindakan. Hal ini akan meniadakan kebutuhan Anda untuk membuat alarm untuk setiap kesalahan metrik atau log.

Memecahkan masalah

CloudWatch Application Insights membuat dasbor CloudWatch otomatis untuk masalah yang terdeteksi. Dasbor tersebut akan menampilkan detail masalah, termasuk anomali metrik terkait dan kesalahan log untuk membantu Anda memecahkan masalah. Dasbor juga akan memberikan wawasan tambahan yang menunjukkan hal-hal yang mungkin menjadi akar masalah dari anomali dan kesalahan ditemukan.

Konsep

Konsep-konsep berikut penting untuk memahami cara Wawasan Aplikasi memantau aplikasi Anda.

Komponen

Penggabungan secara otomatis, mandiri, atau pengelompokan kustom sumber daya serupa yang membentuk sebuah aplikasi. Kami menyarankan pengelompokan sumber daya yang serupa ke dalam komponen-komponen kustom untuk pemantauan yang lebih baik.

Observasi

Sebuah peristiwa individu (anomali metrik, kesalahan log, atau pengecualian) yang terdeteksi dalam suatu aplikasi atau sumber daya aplikasi

Masalah

Masalah-masalah dideteksi dengan mengorelasikan, mengklasifikasikan, dan mengelompokkan observasi-observasi terkait.

Untuk definisi konsep kunci lainnya untuk Wawasan CloudWatch Aplikasi, lihat CloudWatch Konsep Amazon.

Harga

CloudWatch Application Insights menyiapkan metrik dan log yang direkomendasikan untuk sumber daya aplikasi yang dipilih menggunakan CloudWatch metrik, Log, dan Peristiwa untuk pemberitahuan tentang masalah yang terdeteksi. Fitur-fitur ini dibebankan ke AWS akun Anda sesuai dengan CloudWatch harga. Untuk masalah yang terdeteksi, SSM juga OpsItems dibuat oleh Application Insights untuk memberi tahu Anda tentang masalah. Selain itu, Application Insights membuat parameter SSM Parameter Store untuk mengonfigurasi CloudWatch agen pada instans Anda. Fitur-fitur Amazon EC2 Systems Manager dikenakan biaya sesuai dengan penetapan harga SSM. Anda tidak akan dikenakan biaya untuk bantuan pengaturan, pemantauan, analisis data, atau deteksi masalah.

Biaya untuk Wawasan CloudWatch Aplikasi

Biaya untuk Amazon EC2 mencakup penggunaan fitur-fitur berikut ini:

- CloudWatch Agen
  - CloudWatch Grup log agen
  - CloudWatch Metrik agen
  - Grup log Prometheus (untuk beban kerja JMX)

Biaya untuk semua sumber daya termasuk penggunaan fitur-fitur berikut ini:

- CloudWatch alarm (sebagian besar biaya)
- SSM OpsItems (biaya minimal)

Contoh kalkulasi biaya

Biaya-biaya dalam contoh ini dipertimbangkan sesuai dengan skenario berikut.

Anda telah membuat sebuah grup sumber daya yang mencakup komponen-komponen berikut ini:

- Sebuah Instans Amazon EC2 dengan instalasi Server SQL.
- Sebuah Volume Amazon EBS yang terlampir.

Saat Anda memasukkan grup sumber daya ini dengan CloudWatch Application Insights, beban kerja SQL Server yang diinstal pada instans Amazon EC2 akan terdeteksi. CloudWatch Application Insights mulai memantau metrik berikut.

Metrik-metrik berikut dipantau untuk instans contoh Server SQL:

- CPUUtilization
- StatusCheckFailed
- Persentase Memori Bytes yang Dipesan dan Digunakan
- Mbyte Memori yang Tersedia
- Total Byte Antarmuka Jaringan/detik
- Persentase Penggunaan File Halaman
- Persentase Cakram Fisik Disk Time

- Persentase Pemroses Processor Time
- SQLServer:Rasio hit cache Manajer Buffer
- SQLServer:Harapan hidup Manajer Buffer
- SQLServer:Proses Statistik Umum yang diblokir
- SQLServer:Koneksi Pengguna Statistik Umum
- SQLServer:Jumlah Kunci Penguncian/detik
- SQLServer:Permintaan Batch Statistik SQL/detik
- Panjang Antrean Prosesor Sistem

Metrik-metrik berikut dipantau untuk volume yang terlampir ke instans Server SQL:

- VolumeReadBytes
- VolumeWriteBytes
- VolumeReadOps
- VolumeWriteOps
- VolumeTotalReadTime
- VolumeTotalWriteTime
- VolumeIdleTime
- VolumeQueueLength
- VolumeThroughputPercentage
- VolumeConsumedReadWriteOps
- BurstBalance

Untuk skenario ini, biaya dihitung sesuai dengan halaman CloudWatch harga dan halaman harga SSM:

- Metrik-metrik kustom

  Untuk skenario ini, 13 metrik di atas dipancarkan untuk CloudWatch menggunakan agen. CloudWatch Metrik-metrik ini diperlakukan sebagai metrik kustom. Biayanya untuk masing-masing metrik kustom adalah $0,3/bulan. Total biaya untuk metrik-metrik kustom ini adalah 13 * $0,3 = $3,90/bulan.
- Alarm

Untuk skenario ini, CloudWatch Application Insights memonitor 26 metrik secara total, yang menghasilkan 26 alarm. Biayanya untuk setiap alarm adalah $0,1/bulan. Total biaya untuk alarm-alarm tersebut adalah 26 * $0,1 = $2,60/bulan.

- Penyerapan data dan log kesalahan

  Biaya penyerapan data adalah $0,05/GB dan penyimpanan untuk log kesalahan Server SQL adalah $0,03/GB. Total biaya untuk penyerapan data dan log kesalahan adalah $0,05/GB+$0,03/GB= $0,08/GB.

- Amazon EC2 Systems Manager OpsItems

  SSM OpsItem dibuat untuk setiap masalah yang terdeteksi oleh CloudWatch Application Insights. Untuk n jumlah masalah dalam aplikasi Anda, total biaya adalah $0,00267 * n/bulan.

Layanan terkait

Layanan berikut digunakan bersama dengan CloudWatch Application Insights:

AWS Layanan terkait

- Amazon CloudWatch menyediakan visibilitas seluruh sistem ke dalam pemanfaatan sumber daya, kinerja aplikasi, dan kesehatan operasional. Ini mengumpulkan dan melacak metrik, mengirim pemberitahuan alarm, secara otomatis memperbarui sumber daya yang Anda pantau berdasarkan aturan yang Anda tentukan, dan memungkinkan Anda memantau metrik kustom Anda sendiri. CloudWatch Application Insights dimulai melalui CloudWatch —khususnya, dalam dasbor operasional CloudWatch default. Untuk informasi selengkapnya, lihat Panduan CloudWatch Pengguna Amazon.
- CloudWatch Container Insights mengumpulkan, mengumpulkan, dan merangkum metrik dan log dari aplikasi dan layanan mikro dalam kontainer Anda. Anda dapat menggunakan Wawasan Kontainer untuk memantau platform Amazon ECS, Amazon Elastic Kubernetes Service, dan platform Kubernetes di Amazon EC2. Saat Wawasan Aplikasi diaktifkan di konsol Wawasan Kontainer atau Wawasan Aplikasi, Wawasan Aplikasi akan menampilkan masalah-masalah yang terdeteksi di dasbor Wawasan Kontainer Anda. Untuk informasi selengkapnya, silakan lihat Wawasan Kontainer .
- Amazon DynamoDB adalah sebuah layanan basis data NoSQL yang memungkinkan Anda memindahkan beban administrasi untuk mengoperasikan dan menskalakan basis data terdistribusi sehingga Anda tidak perlu khawatir dengan penyediaan perangkat keras, penyiapan dan

konfigurasi, replikasi, patching perangkat lunak, atau penskalaan klaster. DynamoDB juga menawarkan enkripsi saat diam, yang menghilangkan beban operasional dan kerumitan yang biasanya Anda alami dalam melindungi data sensitif.

- Amazon EC2 menyediakan kapasitas komputasi yang dapat diskalakan di Cloud. AWS Anda dapat menggunakan Amazon EC2 untuk meluncurkan beberapa server virtual sebanyak atau sesedikit yang Anda butuhkan, mengonfigurasi keamanan dan jaringan, serta mengelola penyimpanan. Anda dapat menaikkan skala atau menurunkan skala untuk menangani perubahan-perubahan dalam persyaratan atau lonjakan popularitas, yang mengurangi kebutuhan untuk memperkirakan lalu lintas. Untuk informasi selengkapnya, silakan lihat Panduan Pengguna Amazon EC2 untuk Instans Linux atau Panduan EC2 Amazon untuk Instans Windows.
- Amazon Elastic Block Store (Amazon EBS) menyediakan volume penyimpanan tingkat blok untuk digunakan bersama dengan instans Amazon EC2 Anda. Volume EBS Amazon berfungsi seperti perangkat blok mentah yang tidak terformat. Anda dapat menginstal volume ini sebagai perangkat di instans Anda. Volume Amazon EBS yang terpasang pada sebuah instans dipaparkan sebagai volume penyimpanan yang tetap ada secara independen terlepas dari masa pakai instans. Anda dapat membuat sistem file di atas volume ini, atau menggunakannya dengan cara apa pun Anda akan menggunakan perangkat blok (seperti hard drive). Anda dapat secara dinamis mengubah konfigurasi volume yang dipasang ke suatu instans. Untuk informasi selengkapnya, silakan lihat Panduan Pengguna Amazon EBS.
- Amazon EC2 Auto Scaling akan membantu memastikan Anda memiliki jumlah instans Amazon EC2 yang tepat serta tersedia untuk menangani beban aplikasi Anda. Untuk informasi selengkapnya, silakan lihat Panduan Pengguna Amazon EC2 Auto Scaling.
- Penyeimbang Beban Elastis mendistribusikan aplikasi masuk atau lalu lintas jaringan di beberapa target, seperti instans EC2, kontainer, dan alamat IP, di beberapa Zona Ketersediaan. Untuk informasi selengkapnya, silakan lihat Panduan Pengguna Penyeimbang Beban Elastis.
- IAM adalah layanan web yang membantu Anda mengontrol akses ke AWS sumber daya dengan aman bagi pengguna Anda. Gunakan IAM untuk mengontrol siapa yang dapat menggunakan AWS sumber daya Anda (otentikasi), dan untuk mengontrol sumber daya yang dapat mereka gunakan dan bagaimana mereka dapat menggunakannya (otorisasi). Untuk informasi lebih lanjut, lihat Kontrol Autentikasi dan Akses untuk Amazon CloudWatch.
- AWS Lambdamemungkinkan Anda membangun aplikasi tanpa server yang terdiri dari fungsi yang dipicu oleh peristiwa dan secara otomatis menerapkannya menggunakan dan. CodePipeline AWS CodeBuild Untuk informasi selengkapnya, silakan lihat AplikasiAWS Lambda.
- AWS Launch Wizard untuk SQL Server mengurangi waktu yang diperlukan untuk menyebarkan solusi ketersediaan tinggi SQL Server ke cloud. Anda memasukkan persyaratan aplikasi Anda,

termasuk kinerja, jumlah node, dan konektivitas pada konsol layanan, dan AWS Launch Wizard mengidentifikasi AWS sumber daya yang tepat untuk menerapkan dan menjalankan aplikasi SQL Server Always On Anda.

- AWS Resource Groups membantu Anda mengatur sumber daya yang membentuk aplikasi Anda. Dengan Grup Sumber Daya, Anda dapat mengelola dan mengotomatisasi tugas pada sejumlah besar sumber daya sekaligus. Hanya satu Grup Sumber Daya yang dapat didaftarkan untuk satu aplikasi tunggal. Untuk informasi selengkapnya, silakan lihat Panduan Pengguna Grup Sumber DayaAWS .
- Amazon SQS menawarkan sebuah antrean hostingan yang aman, tahan lama, dan tersedia yang memungkinkan Anda untuk mengintegrasikan dan memisahkan sistem dan komponen perangkat lunak terdistribusi. Untuk informasi selengkapnya, silakan lihat Panduan Pengguna Amazon SQS.
- AWS Step Functionsadalah komposer fungsi tanpa server yang memungkinkan Anda mengurutkan berbagai AWS layanan dan sumber daya, termasuk AWS Lambda fungsi, ke dalam alur kerja visual yang terstruktur. Untuk informasi selengkapnya, silakan lihat Panduan PenggunaAWS Step Functions.
- AWS SSM OpsCenter mengumpulkan dan menstandarisasi OpsItems di seluruh layanan sambil memberikan data investigasi kontekstual tentang masing-masing OpsItem, terkait, dan sumber daya terkait. OpsItems OpsCenter juga menyediakan dokumen Systems Manager Automation (runbook) yang dapat Anda gunakan untuk menyelesaikan masalah dengan cepat. Anda dapat menentukan data kustom yang dapat dicari untuk masing-masing data. OpsItem Anda juga dapat melihat laporan ringkasan yang dibuat secara otomatis berdasarkan status dan sumber. OpsItems Untuk informasi selengkapnya, silakan lihat Panduan Pengguna AWS Systems Manager .
- Amazon API Gateway adalah AWS layanan untuk membuat, menerbitkan, memelihara, memantau, dan mengamankan REST, HTTP, dan WebSocket API pada skala apa pun. Pengembang API dapat membuat API yang mengakses AWS atau layanan web lainnya, serta data yang disimpan di AWS Cloud. Untuk informasi selengkapnya, silakan lihat Panduan Pengguna Amazon API Gateway.

  > **Note**
  >
  > Wawasan Aplikasi hanya mendukung protokol API REST (v1 dari layanan API Gateway).

- Amazon Elastic Container Service (Amazon ECS) adalah sebuah layanan orkestrasi kontainer yang terkelola sepenuhnya. Anda dapat menggunakan Amazon ECS untuk menjalankan aplikasi-aplikasi Anda yang paling sensitif dan kritis bagi misi. Untuk informasi selengkapnya, silakan lihat Panduan Developer Amazon Elastic Container Service.

- Amazon Elastic Kubernetes Service (Amazon EKS) adalah layanan terkelola yang dapat Anda gunakan untuk menjalankan AWS Kubernetes tanpa harus menginstal, mengoperasikan, dan memelihara control plane atau node Kubernetes Anda sendiri. Kubernetes adalah sebuah sistem sumber terbuka untuk melakukan otomatisasi terhadap deployment, penskalaan, dan pengelolaan aplikasi terkontainer. Untuk informasi selengkapnya, silakan lihat Panduan Pengguna Amazon EKS.
- Kubernetes di Amazon EC2. Kubernetes adalah perangkat lunak sumber terbuka yang akan membantu Anda dalam menerapkan dan mengelola aplikasi terkontainer dalam skala besar. Kubernetes mengelola klaster-klaster instans komputasi Amazon EC2 dan menjalankan kontainer pada instans tersebut dengan proses untuk deployment, pemeliharaan, dan penskalaan. Dengan Kubernetes Anda dapat menjalankan semua jenis aplikasi terkontainer dengan toolset yang sama secara on-premise dan di cloud. Untuk informasi selengkapnya, silakan lihat Dokumentasi Kubernetes: Memulai.
- Amazon FSx akan membantu Anda dalam meluncurkan dan menjalankan sistem file populer yang terkelola sepenuhnya oleh AWS. Dengan Amazon FSx, Anda akan dapat memanfaatkan serangkaian fitur dan performa sistem file berlisensi komersial sumber terbuka dan umum untuk menghindari tugas administratif yang memakan waktu. Untuk informasi selengkapnya, silakan lihat Dokumentasi Amazon FSx.
- Amazon Simple Notification Service (SNS) adalah layanan pesan yang dikelola sepenuhnya untuk komunikasi dan komunikasi. application-to-application application-to-person Anda dapat mengonfigurasi Amazon SNS untuk melakukan pemantauan berdasarkan Wawasan Aplikasi. Saat Amazon SNS dikonfigurasi sebagai sumber daya untuk melakukan pemantauan, Wawasan Aplikasi akan melacak metrik-metrik SNS untuk membantu menentukan apa yang menjadi penyebab pesan SNS mungkin mengalami masalah atau kegagalan.
- Amazon Elastic File System (Amazon EFS) adalah sistem file NFS elastis yang dikelola sepenuhnya untuk digunakan dengan AWS Cloud layanan dan sumber daya lokal. Amazon EFS dibangun untuk skala petabyte sesuai permintaan tanpa mengganggu aplikasi. Amazon EFS tumbuh dan menyusut secara otomatis saat Anda menambahkan dan menghapus file, yang akan menghilangkan kebutuhan untuk menyediakan dan mengelola kapasitas untuk mengakomodasi pertumbuhan. Untuk informasi selengkapnya, silakan lihat dokumentasi Amazon Elastic File System.

Layanan-layanan pihak ketiga terkait

- Untuk beberapa beban kerja dan aplikasi yang dipantau di Application Insights, eksportir Prometheus JMX diinstal menggunakan Distributor sehingga Application Insights dapat mengambil

metrik khusus Java. AWS Systems Manager CloudWatch Ketika Anda memilih untuk memantau aplikasi Java, Wawasan Aplikasi akan secara otomatis melakukan instalasi pengekspor Prometheus JMX untuk Anda.

Komponen-komponen aplikasi yang didukung

CloudWatch Application Insights memindai grup sumber daya Anda untuk mengidentifikasi komponen aplikasi. Komponen-komponen itu dapat berdiri sendiri, dikelompokkan secara otomatis (seperti instans dalam grup Auto Scaling atau di belakang penyeimbang beban), atau kustom (dengan mengelompokkan masing-masing instans Amazon EC2).

Komponen berikut didukung oleh CloudWatch Application Insights:

AWS komponen

- Amazon EC2
- Amazon EBS
- Amazon RDS
- Penyeimbang Beban Elastis: Penyeimbang Beban Aplikasi dan Penyeimbang Beban Klasik (semua instans target dari penyeimbang beban ini diidentifikasi dan dikonfigurasi).
- Grup Auto Scaling Amazon EC2: Auto AWS Scaling (grup Auto Scaling dikonfigurasi secara dinamis untuk semua instans target; jika aplikasi Anda ditingkatkan, Application CloudWatch Insights secara otomatis mengonfigurasi instans baru). Grup Auto Scaling tidak didukung untuk grup sumber daya berbasis CloudFormation tumpukan.
- AWS Lambda
- Amazon Simple Queue Service (Amazon SQS)
- Tabel Amazon DynamoDB
- Metrik bucket Amazon S3
- AWS Step Functions
- Tahapan API REST Amazon API Gateway
- Amazon Elastic Container Service (Amazon ECS): klaster, layanan, dan tugas
- Amazon Elastic Kubernetes Service (Amazon EKS): klaster
- Kubernetes di Amazon EC2: Klaster Kubernetes yang berjalan di EC2
- Topik Amazon SNS

Sumber daya jenis komponen lainnya saat ini tidak dilacak oleh CloudWatch Application Insights. Jika sebuah tipe komponen yang didukung tidak muncul di aplikasi Wawasan Aplikasi Anda, maka komponen tersebut mungkin sudah terdaftar dan dikelola dengan aplikasi lain yang Anda miliki dan dipantau oleh Wawasan Aplikasi.

Tumpukan teknologi yang didukung

Anda dapat menggunakan CloudWatch Application Insights untuk memantau aplikasi yang berjalan di sistem operasi Windows Server dan Linux dengan memilih opsi menu tarik-turun tingkat aplikasi untuk salah satu teknologi berikut:

- Front-end: Server Web Layanan Informasi Internet (IIS) Microsoft
- Tingkat pekerja:
  - .NET Framework
  - .NET Core
- Aplikasi:
  - Java
  - Penerapan NetWeaver standar SAP, terdistribusi, dan ketersediaan tinggi
- Direktori Aktif
- SharePoint
- Basis data:
  - Microsoft SQL Server yang berjalan di Amazon RDS atau Amazon EC2 (termasuk konfigurasi Server SQL Ketersediaan Tinggi). Lihat, Contoh-contoh konfigurasi komponen).
  - MySQL yang berjalan di Amazon RDS, Amazon Aurora, atau Amazon EC2
  - PostgreSQL yang berjalan di Amazon RDS atau Amazon EC2
  - Tabel Amazon DynamoDB
  - Oracle yang berjalan di Amazon RDS atau Amazon EC2
  - Basi data SAP HANA pada sebuah instans Amazon EC2 tunggal dan beberapa instans EC2
  - Pengaturan ketersediaan tinggi basis data lintas AZ SAP HANA
  - Database SAP Sybase ASE pada satu instans Amazon EC2
  - Pengaturan ketersediaan tinggi basis data SAP Sybase ASE lintas AZ

Jika tidak ada tumpukan teknologi yang tercantum di atas yang berlaku terhadap sumber daya aplikasi Anda, maka Anda dapat memantau tumpukan aplikasi Anda dengan memilih Kustom dari menu geser-turun tingkat aplikasi pada halaman Mengelola pemantauan.

Cara kerja Amazon CloudWatch Application Insights

Bagian ini berisi informasi tentang cara kerja CloudWatch Application Insights, termasuk:

- Cara Wawasan Aplikasi memantau aplikasi
- Retensi data
- Kuota
- AWS Paket Systems Manager (SSM) yang digunakan oleh CloudWatch Application Insights
- AWS Dokumen Systems Manager (SSM) yang digunakan oleh CloudWatch Application Insights

Cara Wawasan Aplikasi memantau aplikasi

Wawasan Aplikasi memantau aplikasi sebagai berikut.

Penemuan dan konfigurasi aplikasi

Pertama kali aplikasi ditambahkan ke CloudWatch Application Insights, aplikasi memindai komponen aplikasi untuk merekomendasikan metrik kunci, log, dan sumber data lainnya untuk memantau aplikasi Anda. Kemudian Anda dapat melakukan konfigurasi atas aplikasi Anda dengan berdasarkan rekomendasi ini.

Prapemrosesan data

CloudWatch Application Insights terus menganalisis sumber data yang dipantau di seluruh sumber daya aplikasi untuk menemukan anomali metrik dan kesalahan log (pengamatan).

Deteksi masalah cerdas

Mesin CloudWatch Application Insights mendeteksi masalah dalam aplikasi Anda dengan menghubungkan pengamatan menggunakan algoritma klasifikasi dan aturan bawaan. Untuk membantu pemecahan masalah, ia membuat CloudWatch dasbor otomatis, yang mencakup informasi kontekstual tentang masalah.

Peringatan dan tindakan

Ketika CloudWatch Application Insights mendeteksi masalah dengan aplikasi Anda, itu menghasilkan CloudWatch Acara untuk memberi tahu Anda tentang masalah tersebut. Lihat Wawasan Aplikasi

CloudWatch Acara dan pemberitahuan untuk masalah yang terdeteksi untuk informasi selengkapnya mengenai cara menyiapkan Event.

Contoh skenario

Anda memiliki sebuah aplikasi ASP .NET yang didukung oleh basis data Server SQL. Tiba-tiba, basis data Anda mulai mengalami malfungsi karena adanya tekanan memori yang tinggi. Hal ini akan menyebabkan penurunan performa aplikasi dan kemungkinan kesalahan HTTP 500 di server web Anda dan penyeimbang beban.

Dengan CloudWatch Application Insights dan analitik cerdasnya, Anda dapat mengidentifikasi lapisan aplikasi yang menyebabkan masalah dengan memeriksa dasbor yang dibuat secara dinamis yang menunjukkan metrik terkait dan cuplikan file log. Dalam hal ini, masalahnya mungkin ada di lapisan basis data SQL.

Retensi data

CloudWatch Application Insights mempertahankan masalah selama 55 hari dan pengamatan selama 60 hari.

Kuota

Untuk kuota default untuk Wawasan CloudWatch Aplikasi, lihat titik akhir dan kuota Amazon CloudWatch Application Insights. Kecuali dinyatakan lain, setiap kuota adalah per AWS Wilayah. Silakan hubungi AWS Support untuk meminta kenaikan kuota layanan Anda. Banyak layanan yang memiliki kuota yang tidak dapat diubah. Untuk informasi selengkapnya tentang kuota untuk sebuah layanan tertentu, silakan lihat dokumentasi untuk layanan tersebut.

AWS Paket Systems Manager (SSM) yang digunakan oleh CloudWatch Application Insights

Paket yang tercantum dalam bagian ini digunakan oleh Application Insights, dan dapat dikelola dan digunakan secara independen dengan AWS Systems Manager Distributor. Untuk informasi selengkapnya tentang SSM Distributor, silakan lihat AWS Distributor Systems Manager di Panduan PenggunaAWS Systems Manager.

Paket:

- AWSObservabilityExporter-JMXExporterInstallAndConfigure
- AWSObservabilityExporter-SAP-HANADBExporterInstallAndConfigure
- AWSObservabilityExporter-HAClusterExporterInstallAndConfigure

- AWSObservabilityExporter-SAP-SAPHostExporterInstallAndConfigure
- AWSObservabilityExporter-SQLExporterInstallAndConfigure

`AWSObservabilityExporter-JMXExporterInstallAndConfigure`

Anda dapat mengambil metrik-metrik Java beban kerja khusus dari pengekspor Prometheus JMX untuk Wawasan Aplikasi untuk melakukan konfigurasi dan pemantauan alarm. Dalam konsol Wawasan Aplikasi, pada halaman Mengelola pemantauan, silakan pilih Aplikasi JAVA dari menu geser-turun Tingkat aplikasi. Kemudian pada Konfigurasi pengekspor JAVA Prometheus, pilih Metode pengumpulan dan Nomor port JMX Anda.

Untuk menggunakan AWS Systems Manager Distributor untuk mengemas, menginstal, dan mengonfigurasi paket eksportir Prometheus JMX yang AWSdisediakan secara independen dari Application Insights, selesaikan langkah-langkah berikut.

Prasyarat untuk menggunakan paket SSM pengekspor Prometheus JMX

- Menginstal agen SSM versi 2.3.1550.0 atau versi yang lebih baru
- Mengatur variabel lingkungan JAVA_HOME

Menginstal dan mengonfigurasi paket `AWSObservabilityExporter-JMXExporterInstallAndConfigure`

Paket `AWSObservabilityExporter-JMXExporterInstallAndConfigure` adalah sebuah paket SSM Distributor yang dapat Anda gunakan untuk melakukan instalasi dan mengonfigurasi Pengekspor Prometheus JMX. Ketika metrik Java dikirim oleh eksportir Prometheus JMX, agen dapat dikonfigurasi untuk mengambil metrik CloudWatch untuk layanan. CloudWatch

1. Berdasarkan preferensi Anda, siapkan file konfigurasi YAMM eksportir Prometheus JMX yang terletak di repositori Prometheus. GitHub Gunakan contoh konfigurasi dan deskripsi opsi tersebut untuk memandu Anda.
2. Salin file konfigurasi YAML pengekspor Prometheus JMX yang dikodekan sebagai Base64 ke sebuah parameter SSM baru di Penyimpanan Parameter SSM.
3. Navigasikan ke konsol SSM Distributor dan buka tab Dimiliki oleh Amazon. Pilih AWSObservabilityExporter-JMX ExporterInstallAndConfigure dan pilih Install satu kali.
4. Perbarui parameter SSM yang sudah Anda buat pada langkah pertama dengan mengganti "Argumen Tambahan" dengan yang berikut ini:

```
{
  "SSM_EXPORTER_CONFIGURATION": "{{ssm:<SSM_PARAMETER_STORE_NAME>}}",
  "SSM_EXPOSITION_PORT": "9404"
}
```

> **Note**
>
> Port 9404 adalah port default yang digunakan untuk mengirimkan metrik-metrik Prometheus JMX. Anda dapat memperbarui port ini.

Contoh: Konfigurasikan CloudWatch agen untuk mengambil metrik Java

1. Instal pengekspor Prometheus JMX, seperti yang telah dijelaskan dalam prosedur sebelumnya. Kemudian lakukan verifikasi bahwa pengekspor tersebut telah diinstal dengan benar pada instans Anda dengan memeriksa status port.

   Instalasi yang berhasil pada contoh instans Windows

   ```
   PS C:\> curl http://localhost:9404 (http://localhost:9404/)
   StatusCode : 200
   StatusDescription : OK
   Content : # HELP jvm_info JVM version info
   ```

   Instalasi yang berhasil pada contoh instans Linux

   ```
   $ curl localhost:9404
   # HELP jmx_config_reload_failure_total Number of times configuration have failed to
    be reloaded.
   # TYPE jmx_config_reload_failure_total counter
   jmx_config_reload_failure_total 0.0
   ```

2. Membuat file YAML penemuan layanan Prometheus. File penemuan layanan contoh berikut melakukan hal-hal berikut:

   - Menentukan port host pengekspor Prometheus JMX sebagai `localhost: 9404`.
   - Melampirkan label (`Application`,`ComponentName`, dan`InstanceId`) ke metrik, yang dapat ditetapkan sebagai dimensi CloudWatch metrik.

```
$ cat prometheus_sd_jmx.yaml
- targets:
  - 127.0.0.1:9404
  labels:
    Application: myApp
    ComponentName: arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/sampl-Appli-MMZW8E3GH4H2/aac36d7fea2a6e5b
    InstanceId: i-12345678901234567
```

3. Membuat file YAML konfigurasi pengeskpor Prometheus JMX. File konfigurasi contoh berikut menentukan hal-hal berikut:

   - Interval tugas pengambilan metrik dan periode habis waktu.
   - Pekerjaan pengambilan metrik (`jmx` dan `sap`), juga dikenal sebagai scraping, yang mencakup nama pekerjaan, deret waktu maksimum yang dikembalikan sekaligus pada satu waktu, dan jalur file penemuan layanan.

```
$ cat prometheus.yaml
global:
  scrape_interval: 1m
  scrape_timeout: 10s
scrape_configs:
  - job_name: jmx
    sample_limit: 10000
    file_sd_configs:
      - files: ["/tmp/prometheus_sd_jmx.yaml"]
  - job_name: sap
    sample_limit: 10000
    file_sd_configs:
      - files: ["/tmp/prometheus_sd_sap.yaml"]
```

4. Verifikasi bahwa CloudWatch agen diinstal pada instans Amazon EC2 Anda dan versinya adalah 1.247346.1b249759 atau yang lebih baru. Untuk menginstal CloudWatch agen pada instans EC2 Anda, lihat Menginstal CloudWatch Agen. Untuk memverifikasi versi, lihat Menemukan informasi tentang versi CloudWatch agen.
5. Konfigurasikan CloudWatch agen. Untuk informasi selengkapnya tentang cara mengonfigurasi file konfigurasi CloudWatch agen, lihat Membuat atau mengedit file konfigurasi CloudWatch agen secara manual. Contoh file konfigurasi CloudWatch agen berikut melakukan hal berikut:

- Menentukan jalur file konfigurasi pengekspor Prometheus JMX.
- Menentukan grup log target yang akan digunakan untuk menerbitkan log metrik EMF.
- Menentukan dua set dimensi untuk masing-masing nama metrik.
- Mengirim 8 (4 nama metrik* 2 set dimensi per nama metrik) CloudWatch metrik.

```
{
   "logs":{
      "logs_collected":{
         ....
      },
      "metrics_collected":{
         "prometheus":{
            "cluster_name":"prometheus-test-cluster",
            "log_group_name":"prometheus-test",
            "prometheus_config_path":"/tmp/prometheus.yaml",
            "emf_processor":{
               "metric_declaration_dedup":true,
               "metric_namespace":"CWAgent",
               "metric_unit":{
                  "jvm_threads_current":"Count",
                  "jvm_gc_collection_seconds_sum":"Second",
                  "jvm_memory_bytes_used":"Bytes"
               },
               "metric_declaration":[
                  {
                     "source_labels":[
                        "job"
                     ],
                     "label_matcher":"^jmx$",
                     "dimensions":[
                        [
                           "InstanceId",
                           "ComponentName"
                        ],
                        [
                           "ComponentName"
                        ]
                     ],
                     "metric_selectors":[
                        "^java_lang_threading_threadcount$",
```

```
                        "^java_lang_memory_heapmemoryusage_used$",
                        "^java_lang_memory_heapmemoryusage_committed$"
                    ]
                }
            ]
        }
    }
  },
  "metrics":{
     ....
  }
}
```

AWSObservabilityExporter-SAP-HANADBExporterInstallAndConfigure

Anda dapat mengambil metrik SAP HANA specifik beban kerja dari pengekspor basis data Prometheus HANA untuk Wawasan Aplikasi untuk mengonfigurasi dan memantau alarm. Untuk informasi selengkapnya, lihat Menyiapkan basis data SAP HANA Anda untuk pemantauan dalam panduan ini.

Untuk menggunakan AWS Systems Manager Distributor untuk mengemas, menginstal, dan mengkonfigurasi paket eksportir database Prometheus HANA yang AWSdisediakan secara independen dari Application Insights, selesaikan langkah-langkah berikut.

Prasyarat untuk menggunakan paket SSM pengekspor basis data Prometheus HANA

- Menginstal agen SSM versi 2.3.1550.0 atau versi yang lebih baru
- Basis data SAP Hana
- Sistem operasi Linux (SUSE Linux, RedHat Linux)
- Sebuah rahasia dengan kredensial pemantauan basis data SAP HANA, dengan menggunakan AWS Secrets Manager. Membuat sebuah rahasia dengan menggunakan format pasangan kunci/nilai, menentukan nama pengguna kunci, dan memasukkan pengguna basis data untuk nilainya. Menambahkan sebuah kata sandi kunci kedua, dan memasukkan kata sandi untuk nilainya. Untuk informasi selengkapnya tentang cara membuat sebuah rahasia, silakan lihat Membuat rahasia di Panduan PenggunaAWS Secrets Manager . Rahasia harus diformat sebagai berikut:

```
{
  "username": "<database_user>",
```

```
  "password": "<database_password>"
}
```

Menginstal dan mengonfigurasi paket **AWSObservabilityExporter-SAP-HANADBExporterInstallAndConfigure**

Paket `AWSObservabilityExporter-SAP-HANADBExporterInstallAndConfigure` adalah sebuah paket SSM Distributor yang dapat Anda gunakan untuk melakukan instalasi dan mengonfigurasi Pengekspor basis data HANA Prometheus. Ketika metrik database HANA dikirim oleh eksportir database Prometheus HANA, CloudWatch agen dapat dikonfigurasi untuk mengambil metrik untuk layanan. CloudWatch

1. Membuat sebuah parameter SSM di SSM Parameter Store untuk menyimpan konfigurasi Pengekspor. Berikut ini adalah sebuah contoh nilai parameter.

```
{\"exposition_port\":9668,\"multi_tenant\":true,\"timeout\":600,\"hana\":{\"host\":
\"localhost\",\"port\":30013,\"aws_secret_name\":\"HANA_DB_CREDS\",\"scale_out_mode
\":true}}
```

> **Note**
>
> Dalam contoh ini, ekspor hanya berjalan pada instans Amazon EC2 dengan basis data SYSTEM aktif, dan ekspor akan tetap menganggur pada instans EC2 lainnya untuk menghindari adanya metrik-metrik duplikat. Pengekspor dapat mengambil semua informasi penyewa basis data dari basis data SYSTEM.

2. Membuat sebuah parameter SSM di SSM Parameter Store untuk menyimpan kueri metrik Pengekspor. Paket tersebut dapat menerima lebih dari satu parameter metrik. Setiap parameter harus memiliki format objek JSON yang benar. Berikut ini adalah sebuah contoh nilai parameter:

```
{\"SELECT MAX(TIMESTAMP) TIMESTAMP, HOST, MEASURED_ELEMENT_NAME CORE,
 SUM(MAP(CAPTION, 'User Time', TO_NUMBER(VALUE), 0)) USER_PCT, SUM(MAP(CAPTION,
 'System Time', TO_NUMBER(VALUE), 0)) SYSTEM_PCT, SUM(MAP(CAPTION, 'Wait
 Time', TO_NUMBER(VALUE), 0)) WAITIO_PCT, SUM(MAP(CAPTION, 'Idle Time', 0,
 TO_NUMBER(VALUE))) BUSY_PCT, SUM(MAP(CAPTION, 'Idle Time', TO_NUMBER(VALUE), 0))
 IDLE_PCT FROM sys.M_HOST_AGENT_METRICS WHERE MEASURED_ELEMENT_TYPE = 'Processor'
 GROUP BY HOST, MEASURED_ELEMENT_NAME;\":{\"enabled\":true,\"metrics\":[{\"name\":
\"hanadb_cpu_user\",\"description\":\"Percentage of CPU time spent by HANA DB in user
 space, over the last minute (in seconds)\",\"labels\":[\"HOST\",\"CORE\"],\"value\":
```

```
\"USER_PCT\",\"unit\":\"percent\",\"type\":\"gauge\"},{\"name\":\"hanadb_cpu_system
\",\"description\":\"Percentage of CPU time spent by HANA DB in Kernel space,
 over the last minute (in seconds)\",\"labels\":[\"HOST\",\"CORE\"],\"value\":
\"SYSTEM_PCT\",\"unit\":\"percent\",\"type\":\"gauge\"},{\"name\":\"hanadb_cpu_waitio
\",\"description\":\"Percentage of CPU time spent by HANA DB in IO mode, over the
 last minute (in seconds)\",\"labels\":[\"HOST\",\"CORE\"],\"value\":\"WAITIO_PCT\",
\"unit\":\"percent\",\"type\":\"gauge\"},{\"name\":\"hanadb_cpu_busy\",\"description
\":\"Percentage of CPU time spent by HANA DB, over the last minute (in seconds)\",
\"labels\":[\"HOST\",\"CORE\"],\"value\":\"BUSY_PCT\",\"unit\":\"percent\",\"type\":
\"gauge\"},{\"name\":\"hanadb_cpu_idle\",\"description\":\"Percentage of CPU time not
 spent by HANA DB, over the last minute (in seconds)\",\"labels\":[\"HOST\",\"CORE
\"],\"value\":\"IDLE_PCT\",\"unit\":\"percent\",\"type\":\"gauge\"}]}}
```

Untuk informasi selengkapnya tentang kueri metrik, lihat SUSE / hanadb_exporterrepo di. GitHub

3. Navigasikan ke konsol SSM Distributor dan buka tab Dimiliki oleh Amazon. Pilih AWSObservabilityExporterExporterInstallAndConfigure-SAP-HANADB* dan pilih Instal satu kali.
4. Perbarui parameter SSM yang sudah Anda buat pada langkah pertama dengan mengganti "Argumen Tambahan" dengan yang berikut ini:

```
{
  "SSM_EXPORTER_CONFIG": "{{ssm:<*SSM_CONFIGURATIONS_PARAMETER_STORE_NAME>*}}",
  "SSM_SID": "<SAP_DATABASE_SID>",
  "SSM_EXPORTER_METRICS_1": "{{ssm:<SSM_FIRST_METRICS_PARAMETER_STORE_NAME>}}",
  "SSM_EXPORTER_METRICS_2": "{{ssm:<SSM_SECOND_METRICS_PARAMETER_STORE_NAME>}}"
}
```

5. Pilih instans Amazon EC2 dengan basis data SAP HANA, kemudian pilih Jalankan.

`AWSObservabilityExporter-HAClusterExporterInstallAndConfigure`

Anda dapat mengambil metrik-metrik klaster High Availability (HA) spesifik beban kerja dari pengekspor klaster HANA Prometheus untuk Wawasan Aplikasi untuk mengonfigurasi dan memantau alarm untuk pengaturan Ketersediaan Tinggi basis data SAP HANA. Untuk informasi selengkapnya, silakan lihat Menyiapkan basis data SAP HANA Anda untuk pemantauan dalam panduan ini.

Untuk menggunakan AWS Systems Manager Distributor untuk mengemas, menginstal, dan mengonfigurasi paket eksportir klaster Prometheus HA yang AWSdisediakan secara independen dari Application Insights, selesaikan langkah-langkah berikut.

Prasyarat untuk menggunakan paket SSM pengekspor klaster Prometheus HA

- Menginstal agen SSM versi 2.3.1550.0 atau versi yang lebih baru
- Klaster HA untuk Pacemaker, Corosync, SBD, dan DRBD
- Sistem operasi Linux (SUSE Linux, RedHat Linux)

Menginstal dan mengonfigurasi paket **AWSObservabilityExporter-HAClusterExporterInstallAndConfigure**

Paket `AWSObservabilityExporter-HAClusterExporterInstallAndConfigure` adalah sebuah paket SSM Distributor yang dapat Anda gunakan untuk melakukan instalasi dan mengonfigurasi Pengekspor Klaster Prometheus HA. Ketika metrik klaster dikirim oleh eksportir database Prometheus HANA, agen dapat dikonfigurasi CloudWatch untuk mengambil metrik untuk layanan. CloudWatch

1. Membuat sebuah parameter SSM di SSM Parameter Store untuk menyimpan konfigurasi Pengekspor dalam format JSON. Berikut ini adalah sebuah contoh nilai parameter.

   ```
   {\"port\":\"9664\",\"address\":\"0.0.0.0\",\"log-level\":\"info\",\"crm-mon-path
   \":\"/usr/sbin/crm_mon\",\"cibadmin-path\":\"/usr/sbin/cibadmin\",\"corosync-
   cfgtoolpath-path\":\"/usr/sbin/corosync-cfgtool\",\"corosync-quorumtool-path\":\"/
   usr/sbin/corosync-quorumtool\",\"sbd-path\":\"/usr/sbin/sbd\",\"sbd-config-path\":
   \"/etc/sysconfig/sbd\",\"drbdsetup-path\":\"/sbin/drbdsetup\",\"enable-timestamps
   \":false}
   ```

   Untuk informasi selengkapnya tentang konfigurasi eksportir, lihat `ClusterLabs / ha_cluster_exporter`repo di. GitHub
2. Navigasikan ke konsol SSM Distributor dan buka tab Dimiliki oleh Amazon. Pilih AWSObservabilityExporter-HA ClusterExporterInstallAndConfigure * dan pilih Instal satu kali.
3. Perbarui parameter SSM yang sudah Anda buat pada langkah pertama dengan mengganti "Argumen Tambahan" dengan yang berikut ini:

   ```
   {
     "SSM_EXPORTER_CONFIG": "{{ssm:<*SSM_CONFIGURATIONS_PARAMETER_STORE_NAME>*}}"
   }
   ```
4. Pilih instans Amazon EC2 dengan basis data SAP HANA, kemudian pilih Jalankan.

`AWSObservabilityExporter-SAP-SAPHostExporterInstallAndConfigure`

Anda dapat mengambil NetWeaver metrik SAP khusus beban kerja dari eksportir host Prometheus SAP untuk Application Insights untuk mengonfigurasi dan memantau alarm untuk penyebaran SAP Distributed dan High Availability. NetWeaver Untuk informasi selengkapnya, lihat Memulai dengan Amazon CloudWatch Application Insights.

Untuk menggunakan AWS Distributor Systems Manager untuk paket, Anda harus melakukan instalasi, dan mengonfigurasi paket pengekspor Host SAP secara independen dari Wawasan Aplikasi, dan selesaikan langkah-langkah berikut.

Prasyarat untuk menggunakan paket SSM pengekspor host Prometheus SAP

- Menginstal agen SSM versi 2.3.1550.0 atau versi yang lebih baru
- Server NetWeaver aplikasi SAP
- Sistem operasi Linux (SUSE Linux, RedHat Linux)

Menginstal dan mengonfigurasi paket `AWSObservabilityExporter-SAP-SAPHostExporterInstallAndConfigure`

`AWSObservabilityExporter-SAP-SAPHostExporterInstallAndConfigure`Paket ini adalah paket Distributor SSM yang dapat Anda gunakan untuk menginstal dan mengkonfigurasi eksportir metrik SAP Prometheus NetWeaver . Ketika NetWeaver metrik SAP dikirim oleh eksportir Prometheus, CloudWatch agen dapat dikonfigurasi untuk mengambil metrik untuk layanan. CloudWatch

1. Membuat sebuah parameter SSM di SSM Parameter Store untuk menyimpan konfigurasi Pengekspor dalam format JSON. Berikut ini adalah sebuah contoh nilai parameter.

   ```
   {\"address\":\"0.0.0.0\",\"port\":\"9680\",\"log-level\":\"info\",\"is-HA\":false}
   ```

   - alamat

     Alamat target untuk mengirim metrik-metrik Prometheus. Nilai bawaannya adalah `localhost`.

   - port

     Port target untuk mengirim metrik-metrik Prometheus. Nilai bawaannya adalah `9680`.

   - is-HA

`true`untuk penerapan SAP NetWeaver High Availability. Untuk semua deployment lainnya nilainya adalah. `false`

2. Navigasikan ke konsol SSM Distributor dan buka tab Dimiliki oleh Amazon. Pilih AWSObservabilityExporter-SAP-SAP HostExporterInstallAndConfigure dan pilih Install satu kali.
3. Perbarui parameter SSM yang sudah Anda buat pada langkah pertama dengan mengganti "Argumen Tambahan" dengan yang berikut ini:

```
{
  "SSM_EXPORTER_CONFIG": "{{ssm:<SSM_CONFIGURATIONS_PARAMETER_STORE_NAME>}}",
  "SSM_SID": "<SAP_DATABASE_SID>",
  "SSM_INSTANCES_NUM": "<instances_number seperated by comma>"
}
```

Contoh

```
{
  "SSM_EXPORTER_CONFIG": "{{ssm:exporter_config_paramter}}",
  "SSM_INSTANCES_NUM": "11,12,10",
  "SSM_SID": "PR1"
}
```

4. Pilih instans Amazon EC2 dengan NetWeaver aplikasi SAP, dan pilih Jalankan.

> **Note**
>
> Eksportir Prometheus melayani metrik SAP pada titik akhir lokal. NetWeaver Titik akhir lokal tersebut hanya dapat diakses oleh pengguna sistem operasi pada instans Amazon EC2. Oleh karena itu, setelah paket pengekspor diinstal, metrik-metrik tersedia untuk semua pengguna sistem operasi. Titik akhir lokal default-nya adalah `localhost:9680/metrics`.

`AWSObservabilityExporter-SQLExporterInstallAndConfigure`

Anda dapat mengambil metrik-metrik Server SQL Server spesifik beban kerja dari pengekspor Prometheus SQL untuk Wawasan Aplikasi untuk memantau metrik-metrik kunci.

Untuk menggunakan AWS Systems Manager Distributor untuk paket, Anda harus melakukan instalasi, dan mengonfigurasi paket pengekspor SQL secara independen dari Wawasan Aplikasi, dan selesaikan langkah-langkah berikut.

Prasyarat untuk menggunakan paket SSM pengekspor Prometheus SQL

- Menginstal agen SSM versi 2.3.1550.0 atau versi yang lebih baru
- Instans Amazon EC2 yang menjalankan Server SQL di Windows dengan autentikasi pengguna Server SQL yang diaktifkan.
- Seorang pengguna Server SQL dengan izin-izin berikut ini:

```
GRANT VIEW ANY DEFINITION TO
```

```
GRANT VIEW SERVER STATE TO
```

- Sebuah rahasia yang berisi string koneksi basis data menggunakan AWS Secrets Manager. Untuk informasi selengkapnya tentang cara membuat sebuah rahasia, silakan lihat Membuat rahasia di Panduan PenggunaAWS Secrets Manager . Rahasia harus diformat sebagai berikut:

```
{
  "data_source_name":"sqlserver://<username>:<password>@localhost:1433"
}
```

> **Note**
>
> Jika kata sandi atau nama pengguna berisi karakter khusus, maka Anda harus mengkodekan karakter khusus tersebut untuk memastikan koneksi yang berhasil ke basis data.

Menginstal dan mengonfigurasi paket **`AWSObservabilityExporter-SQLExporterInstallAndConfigure`**

Paket `AWSObservabilityExporter-SQLExporterInstallAndConfigure` adalah sebuah paket SSM Distributor yang dapat Anda gunakan untuk melakukan instalasi dan mengonfigurasi pengekspor metrik SQL Prometheus. Ketika metrik dikirim oleh eksportir Prometheus, agen dapat dikonfigurasi CloudWatch untuk mengambil metrik untuk layanan. CloudWatch

1. Berdasarkan pilihan Anda, siapkan konfigurasi YAML Pengekspor SQL. Konfigurasi sampel berikut memiliki satu metrik tunggal yang dikonfigurasi. Gunakan konfigurasi sampel untuk memperbarui konfigurasi tersebut dengan metrik tambahan atau Anda bisa membuat konfigurasi Anda sendiri.

```
---
global:
  scrape_timeout_offset: 500ms
  min_interval: 0s
  max_connections: 3
  max_idle_connections: 3
target:
  aws_secret_name: <SECRET_NAME>
  collectors:
    - mssql_standard
collectors:
  - collector_name: mssql_standard
    metrics:
    - metric_name: mssql_batch_requests
      type: counter
      help: 'Number of command batches received.'
      values: [cntr_value]
      query: |
        SELECT cntr_value
        FROM sys.dm_os_performance_counters WITH (NOLOCK)
        WHERE counter_name = 'Batch Requests/sec'
```

2. Salin file konfigurasi YAML pengekspor Prometheus SQL yang dikodekan sebagai Base64 ke sebuah parameter SSM baru di Penyimpanan Parameter SSM.
3. Navigasikan ke konsol SSM Distributor dan buka tab Dimiliki oleh Amazon. Pilih AWSObservabilityExporter-SQL ExporterInstallAndConfigure dan pilih Install satu kali.
4. Ganti "Argumen Tambahan" dengan informasi berikut ini. `SSM_PARAMETER_NAME` adalah nama parameter yang sudah Anda buat di Langkah 2.

```
{
  "SSM_EXPORTER_CONFIGURATION":
    "{{ssm:<SSM_PARAMETER_STORE_NAME>}}",
    "SSM_PROMETHEUS_PORT": "9399",
    "SSM_WORKLOAD_NAME": "SQL"
}
```

5. Pilih instans Amazon EC2 dengan basis data Server SQL, kemudian pilih jalankan.

AWS Dokumen Systems Manager (SSM) yang digunakan oleh CloudWatch Application Insights

Wawasan Aplikasi menggunakan Dokumen SSM yang tercantum di bagian ini untuk menentukan tindakan-tindakan yang dilakukan oleh AWS Systems Manager pada instans-instans terkelola Anda. Dokumen-dokumen ini menggunakan kemampuan `Run Command` Systems Manager untuk melakukan otomatisasi pada tugas-tugas yang diperlukan untuk melaksanakan kemampuan pemantauan Wawasan Aplikasi. Jadwal eksekusi untuk dokumen-dokumen ini dikelola oleh Wawasan Aplikasi dan tidak dapat diubah.

Untuk informasi selengkapnya tentang Dokumen SSM, silakan lihat DokumenAWS Systems Manager di Panduan PenggunaAWS Systems Manager .

Dokumen yang dikelola oleh CloudWatch Application Insights

Tabel berikut mencantumkan dokumen-dokumen SSM yang dikelola oleh Wawasan Aplikasi.

Nama dokumen	Deskripsi	Jadwal eksekusi
`AWSEC2-DetectWorkload`	Secara otomatis mendeteksi aplikasi yang berjalan di lingkungan aplikasi Anda yang dapat diatur agar bisa dipantau oleh Wawasan Aplikasi.	Dokumen ini berjalan setiap jam di lingkungan aplikasi Anda untuk mendapatkan detail up-to-date aplikasi.
`AWSEC2-CheckPerformanceCounterSets`	Memeriksa apakah namespace Penghitung Performa sudah diaktifkan pada instans Amazon EC2 Anda.	Dokumen ini berjalan setiap jam di lingkungan aplikasi Anda dan hanya memantau metrik-metrik Penghitung Performa jika namespace yang sesuai diaktifkan.
`AWSEC2-ApplicationInsightsCloudwatch`	Menginstal dan mengonfigurasi CloudWatch Agen berdasarkan konfigurasi	Dokumen ini berjalan setiap 30 menit untuk memastikan bahwa konfigurasi CloudWatc

Nama dokumen	Deskripsi	Jadwal eksekusi
`AgentInstallAndConfigure`	pemantauan komponen aplikasi Anda.	h Agen selalu akurat dan up-to-date. Dokumen tersebut juga akan langsung berjalan setelah ada perubahan yang dilakukan pada pengaturan pemantauan aplikasi Anda, misalnya terjadi penambahan atau penghapusan metrik atau pembaruan konfigurasi log.

Dokumen yang dikelola oleh AWS Systems Manager

Dokumen-dokumen berikut digunakan oleh CloudWatch Application Insights dan dikelola oleh Systems Manager.

`AWS-ConfigureAWSPackage`

Application Insights menggunakan dokumen ini untuk menginstal dan menghapus paket distributor eksportir Prometheus, untuk mengumpulkan metrik spesifik beban kerja, dan untuk memungkinkan pemantauan komprehensif beban kerja pada instans Amazon EC2 pelanggan. CloudWatch Application Insights menginstal paket distributor eksportir Prometheus hanya jika beban kerja target yang berkorelasi berjalan pada instance Anda.

Tabel berikut mencantumkan paket distributor pengekspor Prometheus dan beban kerja target yang berkorelasi.

Nama paket distributor pengekspor Prometheus	Beban kerja target
`AWSObservabilityExporter-HAClusterExporterInstallAndConfigure`	SAP HANA HA
`AWSObservabilityExporter-JMXExporterInstallAndConfigure`	Java/JMX

Nama paket distributor pengekspor Prometheus	Beban kerja target
`AWSObservabilityExporter-SAP-HANADBExporterInstallAndConfigure`	SAP HANA
`AWSObservabilityExporter-SAP-SAPHostExporterInstallAndConfigure`	NetWeaver
`AWSObservabilityExporter-SQLExporterInstallAndConfigure`	Server SQL (Windows) dan SAP ASE (Linux)

`AmazonCloudWatch-ManageAgent`

Application Insights menggunakan dokumen ini untuk mengelola status dan konfigurasi CloudWatch Agen pada instans Anda dan untuk mengumpulkan metrik dan log tingkat sistem internal dari instans Amazon EC2 di seluruh sistem operasi.

Memulai dengan Amazon CloudWatch Application Insights

Untuk memulai dengan CloudWatch Application Insights, verifikasi bahwa Anda telah memenuhi prasyarat berikut dan telah membuat kebijakan IAM. Kemudian, Anda dapat memulai menggunakan tautan konsol untuk mengaktifkan CloudWatch Application Insights. Untuk mengonfigurasi sumber daya aplikasi Anda, ikuti langkah-langkah pada Menyiapkan, mengonfigurasi, dan mengelola aplikasi Anda untuk pemantauan.

Daftar Isi

Akses Wawasan CloudWatch Aplikasi

Anda dapat mengakses dan mengelola Wawasan CloudWatch Aplikasi melalui salah satu antarmuka berikut:

- CloudWatch konsol. Untuk menambahkan monitor untuk aplikasi Anda, pilih Wawasan Aplikasi di bawah Wawasan di panel navigasi kiri konsol. CloudWatch Setelah aplikasi Anda dikonfigurasi, Anda dapat menggunakan CloudWatch konsol untuk melihat dan menganalisis masalah yang terdeteksi.
- AWS Antarmuka Baris Perintah (AWS CLI). Anda dapat menggunakan AWS CLI untuk mengakses operasi AWS API. Untuk informasi selengkapnya, lihat Menginstal Antarmuka Baris AWSAWS Perintah di Panduan Pengguna Antarmuka Baris Perintah. Untuk informasi Application Insights API, lihat Referensi API Amazon CloudWatch Application Insights.

Prasyarat

Anda harus menyelesaikan prasyarat berikut untuk mengonfigurasi aplikasi dengan CloudWatch Application Insights:

- AWS Systems Manager pemberdayaan — Instal Systems Manager Agent (Agen SSM) di instans Amazon EC2 Anda, dan aktifkan instans untuk SSM. Untuk informasi tentang cara melakukan instalasi agen SSM, silakan lihat Pengaturan AWS Systems Manager di Panduan PenggunaAWS Systems Manager .
- Peran instans EC2 – Anda harus melampirkan peran instans Amazon EC2 berikut untuk mengaktifkan Systems Manager
  - Anda harus melampirkan peran `AmazonSSMManagedInstanceCore` untuk mengaktifkan Systems Manager. Untuk informasi selengkapnya, silakan lihat contoh kebijakan berbasis identitasAWS Systems Manager.
  - Anda harus melampirkan `CloudWatchAgentServerPolicy` kebijakan untuk mengaktifkan metrik instans dan log yang akan dipancarkan. CloudWatch Untuk informasi selengkapnya, lihat Membuat peran IAM dan pengguna untuk digunakan dengan CloudWatch agen.
- AWS grup sumber daya — Untuk onboard aplikasi Anda ke CloudWatch Application Insights, buat grup sumber daya yang menyertakan semua sumber AWS daya terkait yang digunakan oleh tumpukan aplikasi Anda. Ini termasuk penyeimbang beban aplikasi, instans Amazon EC2 yang menjalankan IIS dan web front-end, .NET level pekerja, dan basis data Server SQL. Untuk informasi selengkapnya tentang komponen aplikasi dan tumpukan teknologi yang didukung oleh

Application Insights, lihat. Komponen-komponen aplikasi yang didukung CloudWatch Wawasan Aplikasi secara otomatis menyertakan grup Auto Scaling menggunakan tag CloudFormation atau tumpukan yang sama dengan grup sumber daya Anda, karena grup Auto Scaling tidak didukung oleh grup sumber daya. CloudFormation Untuk informasi selengkapnya, silakan lihat Memulai dengan Grup Sumber Daya AWS.

- Izin IAM — Untuk pengguna yang tidak memiliki akses administratif, Anda harus membuat kebijakan AWS Identity and Access Management (IAM) yang memungkinkan Wawasan Aplikasi untuk membuat peran terkait layanan dan melampirkannya ke identitas pengguna. Untuk informasi selengkapnya mengenai cara membuat kebijakan IAM, silakan lihat Kebijakan IAM.
- Peran terkait layanan — Application Insights menggunakan peran terkait layanan AWS Identity and Access Management (IAM). Sebuah peran terkait layanan dibuat untuk Anda saat Anda membuat aplikasi Wawasan Aplikasi pertama Anda di konsol Wawasan Aplikasi. Untuk informasi selengkapnya, lihat Menggunakan peran tertaut layanan untuk Wawasan Aplikasi CloudWatch.
- Dukungan metrik Penghitung Performa untuk instans Windows EC2 – Untuk memantau metrik–metrik Penghitung Performa di instans Windows Amazon EC2, Penghitung Performa harus diinstal pada instans tersebut. Untuk metrik-metrik Penghitung Performa dan nama rangkaian Penghitung Performa terkait, silakan lihat metrik Penghitung Performa. Untuk informasi selengkapnya tentang Penghitung Performa, silakan lihat Penghitung Performa.
- CloudWatch Agen Amazon - Application Insights menginstal dan mengonfigurasi agen. CloudWatch Jika Anda telah menginstal CloudWatch agen, Application Insights mempertahankan konfigurasi Anda. Untuk menghindari konflik gabungan, hapus konfigurasi sumber daya yang ingin Anda gunakan di Application Insights dari file konfigurasi CloudWatch agen yang ada. Untuk informasi selengkapnya, lihat Buat atau edit file konfigurasi CloudWatch agen secara manual.

Kebijakan IAM

Untuk menggunakan CloudWatch Application Insights, Anda harus membuat kebijakanAWS Identity and Access Management (IAM) dan melampirkannya ke pengguna, grup, atau peran Anda. Untuk informasi selengkapnya tentang pengguna, grup, dan peran, silakan lihat Identitas IAM (pengguna, grup pengguna, dan peran). Kebijakan IAM mendefinisikan izin pengguna.

Cara membuat sebuah kebijakan IAM dengan menggunakan konsol IAM

Untuk membuat sebuah kebijakan IAM dengan menggunakan konsol IAM, Anda harus melakukan langkah-langkah berikut.

1. Buka Konsol IAM. Pada panel navigasi sebelah kiri, pilih Kebijakan.

2. Di bagian atas halaman, pilih Buat kebijakan.
3. Pilih tab JSON.
4. Salin dan tempel dokumen JSON berikut yang ada tab JSON.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "applicationinsights:*",
                "iam:CreateServiceLinkedRole",
                "iam:ListRoles",
                "resource-groups:ListGroups"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

5. Pilih Kebijakan Peninjauan.
6. Masukkan Nama untuk kebijakan, misalnya, “AppInsightsPolicy.” Jika mau, masukkan Deskripsi.
7. Pilih Buat Kebijakan.
8. Pada panel navigasi sebelah kiri, pilih Grup pengguna, Pengguna, atau Peran.
9. Pilih nama grup pengguna, pengguna, atau peran yang ingin Anda lampiri dengan kebijakan ini.
10. Pilih Tambahkan izin.
11. Pilih Lampirkan kebijakan yang ada secara langsung.
12. Cari kebijakan yang baru saja Anda buat, dan centang kotak yang ada di sebelah kiri nama kebijakan.
13. Pilih Berikutnya: Tinjauan.
14. Pastikan bahwa kebijakan yang benar sudah tercantum, kemudian pilih Tambahkan izin.
15. Pastikan Anda masuk dengan pengguna yang terkait dengan kebijakan yang baru saja Anda buat saat menggunakan CloudWatch Application Insights.

Untuk membuat kebijakan IAM menggunakan AWS CLI

Untuk membuat kebijakan IAM menggunakan AWS CLI, jalankan operasi create-policy dari baris perintah menggunakan dokumen JSON di atas sebagai file di folder Anda saat ini.

Untuk membuat kebijakan IAM menggunakan AWS Tools for Windows PowerShell

Untuk membuat kebijakan IAM menggunakan AWS Tools for Windows PowerShell, jalankan cmdlt New-Iampolicy menggunakan dokumen JSON di atas sebagai file di folder Anda saat ini.

Izin peran IAM untuk melakukan onboarding aplikasi berbasis akun

Jika Anda ingin melakukan onboarding untuk semua sumber daya yang ada di akun Anda, dan Anda memilih untuk tidak menggunakan Kebijakan terkelola Wawasan Aplikasi untuk akses penuh ke fungsionalitas Wawasan Aplikasi, maka Anda harus melampirkan izin-izin berikut ke peran IAM Anda sehingga Wawasan Aplikasi dapat menemukan semua sumber daya yang ada di akun Anda:

```
"ec2:DescribeInstances"
"ec2:DescribeNatGateways"
"ec2:DescribeVolumes"
"ec2:DescribeVPCs"
"rds:DescribeDBInstances"
"rds:DescribeDBClusters"
"sqs:ListQueues"
"elasticloadbalancing:DescribeLoadBalancers"
"autoscaling:DescribeAutoScalingGroups"
"lambda:ListFunctions"
"dynamodb:ListTables"
"s3:ListAllMyBuckets"
"sns:ListTopics"
"states:ListStateMachines"
"apigateway:GET"
"ecs:ListClusters"
"ecs:DescribeTaskDefinition"
"ecs:ListServices"
"ecs:ListTasks"
"eks:ListClusters"
"eks:ListNodegroups"
"fsx:DescribeFileSystems"
"route53:ListHealthChecks"
"route53:ListHostedZones"
"route53:ListQueryLoggingConfigs"
"route53resolver:ListFirewallRuleGroups"
"route53resolver:ListFirewallRuleGroupAssociations"
"route53resolver:ListResolverEndpoints"
```

```
"route53resolver:ListResolverQueryLogConfigs"
"route53resolver:ListResolverQueryLogConfigAssociations"
"logs:DescribeLogGroups"
"resource-explorer:ListResources"
```

Menyiapkan, mengonfigurasi, dan mengelola aplikasi Anda untuk pemantauan

Bagian ini menyediakan langkah-langkah untuk mengatur, mengonfigurasi, dan mengelola CloudWatch aplikasi Application Insights Anda menggunakan konsol, konsol AWS CLI, dan AWS Tools for Windows PowerShellaplikasi.

Topik

- Siapkan, konfigurasikan, dan kelola aplikasi Anda untuk pemantauan dari CloudWatch konsol
- Menyiapkan, mengonfigurasi, dan mengelola aplikasi Anda untuk pemantauan dengan menggunakan baris perintah
- Wawasan Aplikasi CloudWatch Acara dan pemberitahuan untuk masalah yang terdeteksi

Siapkan, konfigurasikan, dan kelola aplikasi Anda untuk pemantauan dari CloudWatch konsol

Bagian ini menyediakan langkah-langkah untuk mengatur, mengkonfigurasi, dan mengelola aplikasi Anda untuk pemantauan dari CloudWatch konsol.

Prosedur konsol

- Menambahkan dan mengonfigurasi sebuah aplikasi
- Aktifkan Wawasan Aplikasi untuk melakukan pemantauan sumber daya Amazon ECS dan Amazon EKS
- Menonaktifkan pemantauan untuk sebuah komponen aplikasi
- Menghapus sebuah aplikasi

Menambahkan dan mengonfigurasi sebuah aplikasi

Tambahkan dan konfigurasikan aplikasi dari CloudWatch konsol

Untuk memulai CloudWatch Application Insights dari CloudWatch konsol, lakukan langkah-langkah berikut.

1. Mulai. Membuka CloudWatch halaman arahan konsol. Dari panel navigasi yang ada di bagian kiri, pada Pemantauan infrastruktur, pilih Wawasan Aplikasi. Halaman yang terbuka

menunjukkan daftar aplikasi yang dipantau dengan Wawasan CloudWatch Aplikasi, bersama dengan status pemantauannya.

2. Menambahkan sebuah aplikasi. Untuk menyiapkan pemantauan aplikasi Anda, pilih Tambahkan aplikasi. Ketika Anda memilih Tambahkan aplikasi, Anda akan diminta untuk Memilih Jenis Aplikasi.

   - Aplikasi berbasis grup sumber daya. Saat Anda memilih opsi ini, Anda dapat memilih grup sumber daya mana yang ada di akun ini yang ingin Anda pantau. Untuk menggunakan beberapa aplikasi pada sebuah komponen, Anda harus menggunakan pemantauan berbasis grup sumber daya.
   - Aplikasi berbasis akun. Saat Anda memilih opsi ini, Anda akan dapat memantau semua sumber daya yang ada di akun ini. Jika Anda ingin memantau semua sumber daya dalam sebuah akun, kami sarankan Anda untuk memilih opsi ini dari pada opsi berbasis grup sumber daya karena proses pelaksanaan onboarding-nya lebih cepat.

   > **Note**
   >
   > Anda tidak dapat menggabungkan pemantauan berbasis grup sumber daya dengan pemantauan berbasis akun dengan menggunakan Wawasan Aplikasi. Untuk mengubah jenis aplikasi, Anda harus menghapus semua aplikasi yang saat ini sedang dipantau, dan Pilih Jenis Aplikasi.

   Saat Anda menambahkan aplikasi pertama untuk pemantauan, CloudWatch Application Insights membuat peran terkait layanan di akun Anda, yang memberikan izin Application Insights untuk memanggil layanan lain AWS atas nama Anda. Untuk informasi selengkapnya tentang peran terkait layanan yang dibuat di akun Anda berdasarkan Wawasan Aplikasi, silakan lihat Menggunakan peran tertaut layanan untuk Wawasan Aplikasi CloudWatch.

3. Resource-based application monitoring

   1. Memilih grup sumber daya. Pada halaman Tentukan detail aplikasi, pilih grup AWS sumber daya yang berisi sumber daya aplikasi Anda dari daftar tarik-turun. Sumber daya ini akan mencakup server front-end, penyeimbang beban, grup auto scaling, dan server basis data.

      Jika Anda belum membuat sebuah grup sumber daya untuk aplikasi Anda, maka Anda dapat membuatnya dengan memilih Buat grup sumber daya baru. Untuk informasi

selengkapnya tentang membuat grup sumber daya, silakan lihat Panduan PenggunaAWS Resource Groups .

2. Pantau CloudWatch Acara. Pilih kotak centang untuk mengintegrasikan pemantauan Wawasan Aplikasi dengan CloudWatch Acara untuk mendapatkan wawasan dari Amazon EBS, Amazon EC2, Amazon ECS AWS CodeDeploy, AWS Health API dan Pemberitahuan, Amazon RDS, Amazon S3, dan. AWS Step Functions
3. Integrasi dengan AWS Systems Manager OpsCenter Untuk melihat dan mendapatkan pemberitahuan saat masalah terdeteksi untuk aplikasi yang dipilih, pilih kotak centang Generate Systems Manager OpsCenter OpsItems for remedial actions. Untuk melacak operasi yang diambil untuk menyelesaikan item kerja operasional (OpsItems) yang terkait dengan AWS sumber daya Anda, berikan topik SNS ARN.
4. Tag - opsional. CloudWatch Application Insights mendukung grup sumber daya CloudFormation berbasis tag dan berbasis (dengan pengecualian grup Auto Scaling). Untuk informasi selengkapnya, silakan lihat Bekerja dengan Penyunting Tag.
5. Pilih Berikutnya.

   Sebuah ARN dibuat untuk aplikasi dalam format berikut.

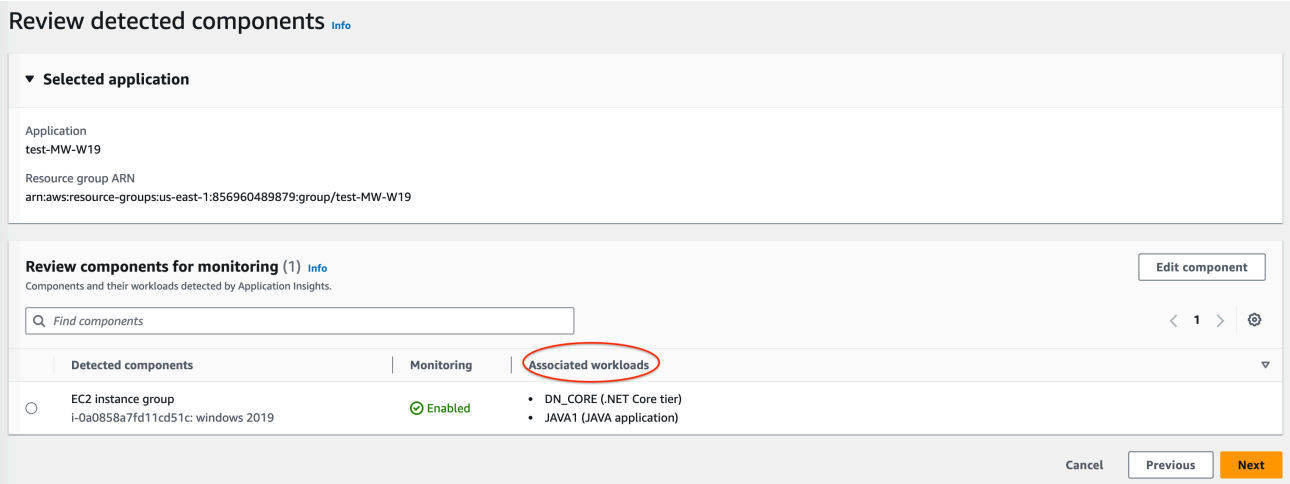
   ```
   arn:partition:applicationinsights:region:account-id:application/resource-group/resource-group-name
   ```

   Contoh

   ```
   arn:aws:applicationinsights:us-east-1:123456789012:application/resource-group/my-resource-group
   ```

6. Pada halaman Meninjau komponen yang terdeteksi, pada Meninjau komponen untuk pemantauan, tabel akan mencantumkan komponen-komponen yang terdeteksi dan beban kerja terkait yang terdeteksi.

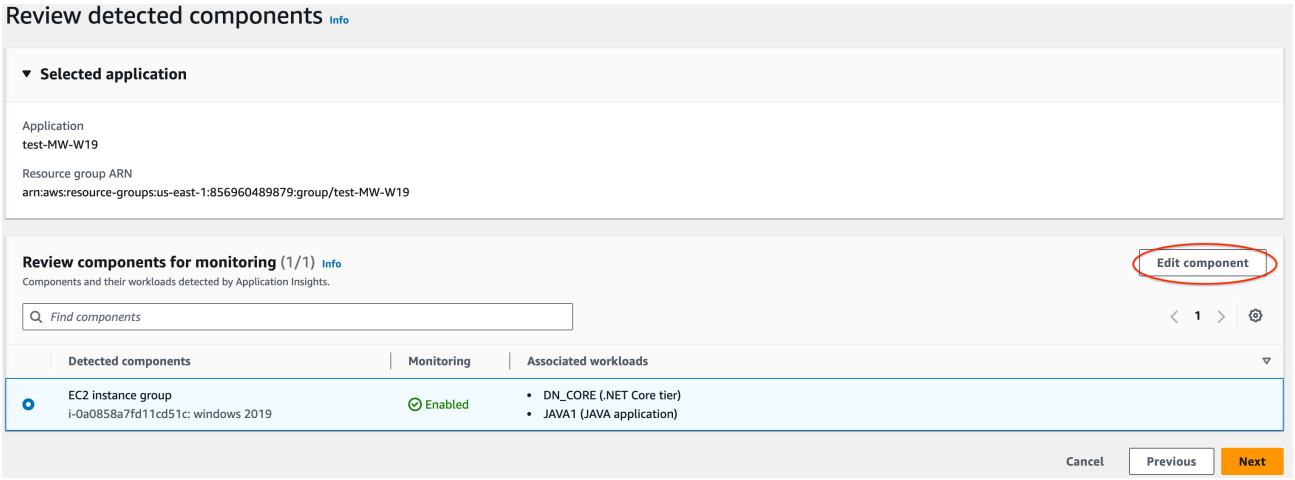
   > **Note**
   >
   > Untuk komponen-komponen yang mendukung beberapa beban kerja yang disesuaikan, Anda dapat memantau hingga lima beban kerja untuk masing-masing komponen. Beban kerja ini akan dipantau secara terpisah dari komponen tersebut.



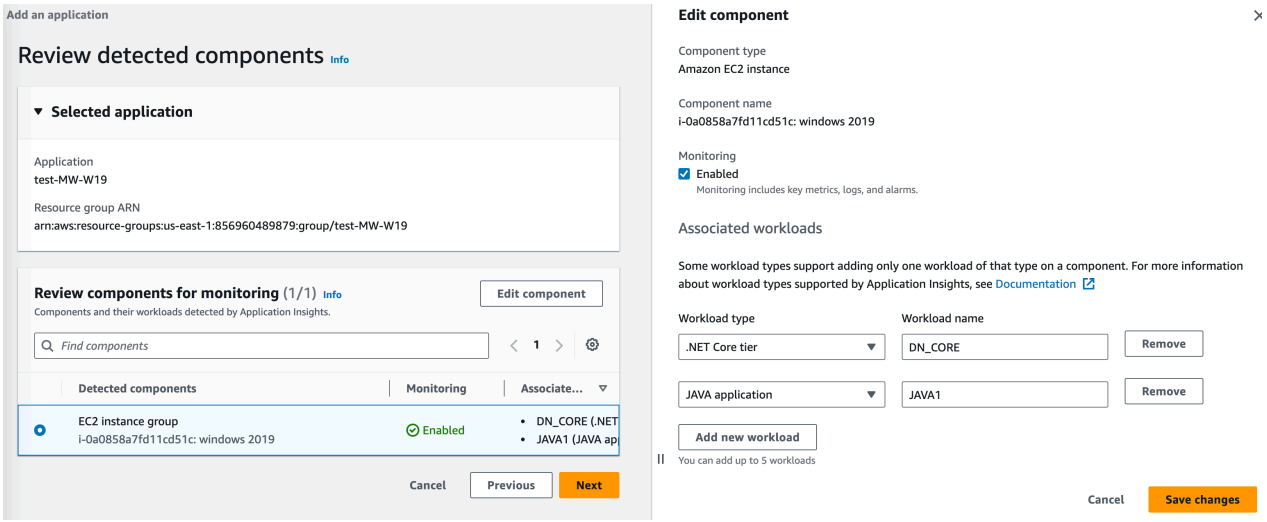
Pada Beban kerja terkait, ada beberapa kemungkinan pesan yang muncul jika beban kerja tidak terdaftar.

- Tidak dapat mendeteksi beban kerja – Sebuah masalah terjadi pada saat mencoba mendeteksi beban kerja. Pastikan bahwa Anda telah menyelesaikan Prasyarat. Jika Anda perlu menambahkan beban kerja, silakan pilih Edit komponen.
- Tidak ada beban kerja yang terdeteksi – Kami tidak mendeteksi beban kerja apa pun. Anda mungkin perlu menambahkan beban kerja. Untuk melakukan hal ini, pilih Edit komponen.
- Tidak berlaku – Komponen tidak mendukung beban kerja yang disesuaikan dan akan dipantau dengan metrik, alarm, dan log default. Anda tidak dapat menambahkan beban kerja ke komponen-komponen ini.

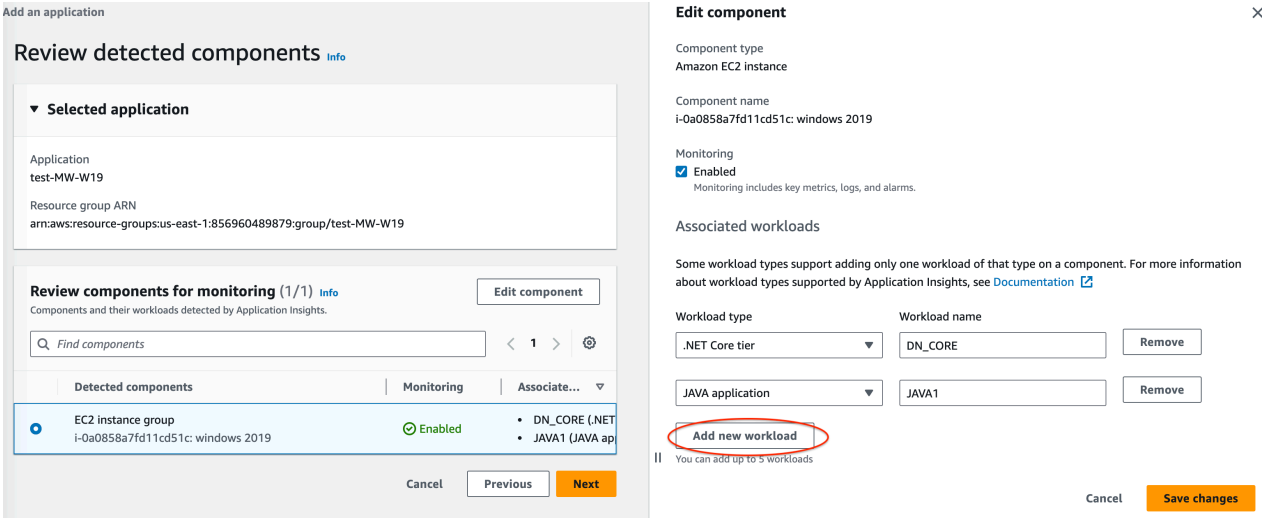
7. Untuk menyunting komponen, pilih sebuah komponen, kemudian pilih Edit komponen. Sebuah panel samping akan terbuka dengan beban kerja yang terdeteksi pada komponen tersebut. Pada panel ini, Anda dapat menyunting detail komponen dan menambahkan beban kerja baru.



- Untuk menyunting jenis atau nama beban kerja, gunakan daftar geser-turun.

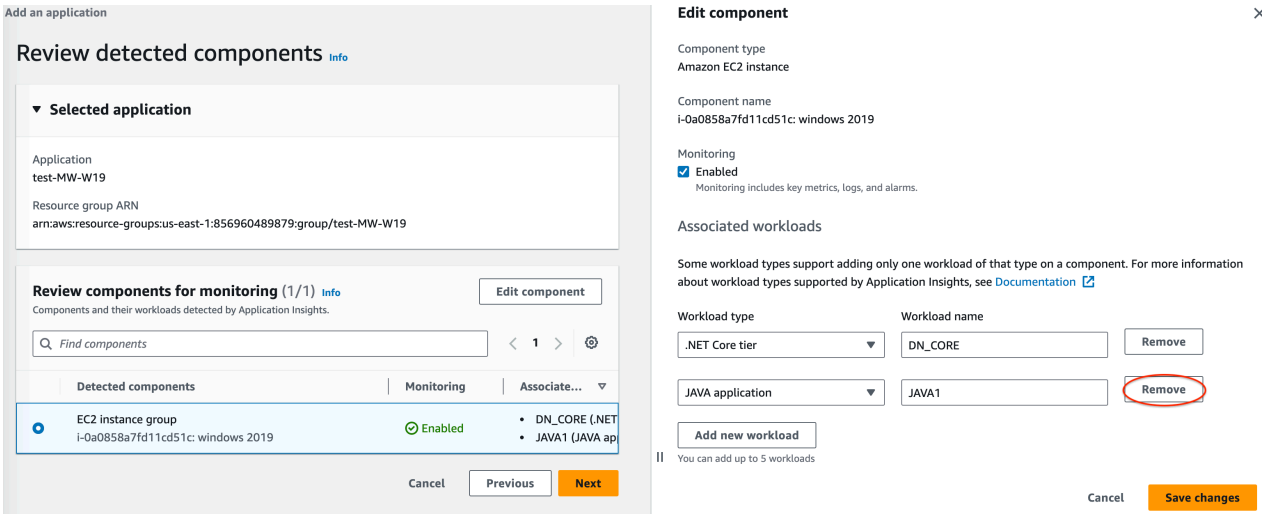


- Untuk menambahkan sebuah beban kerja ke komponen, pilih Tambahkan beban kerja baru.

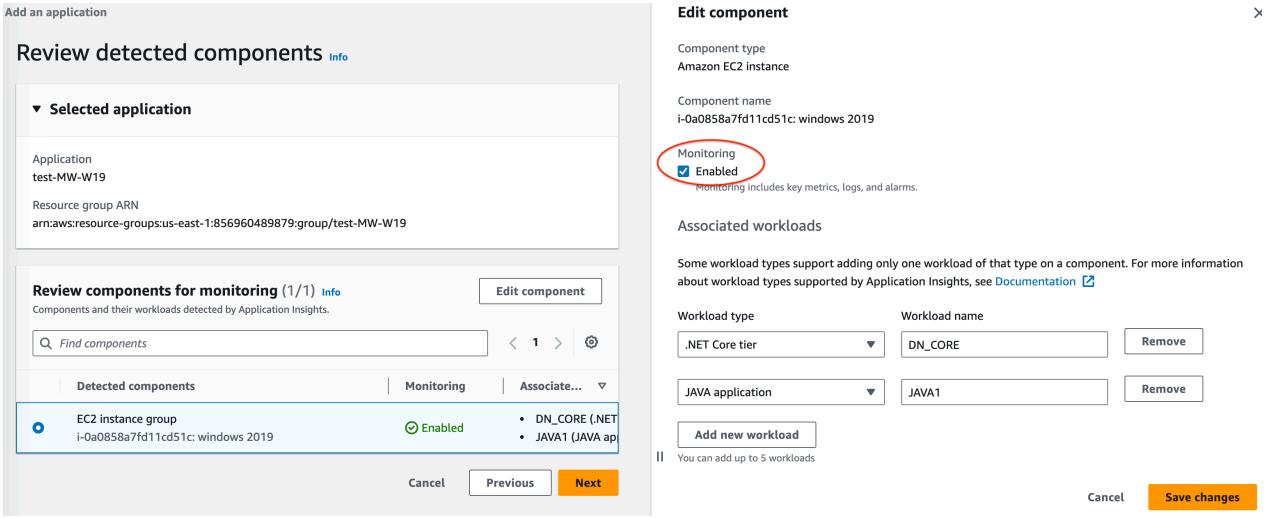


- Jika Tambahkan beban kerja baru tidak muncul, maka komponen ini tidak mendukung banyak beban kerja.
- Jika judul Beban kerja terkait tidak muncul, maka komponen ini tidak mendukung beban kerja yang disesuaikan.

- Untuk menghapus sebuah beban kerja, pilih Hapus yang ada di samping beban kerja yang ingin Anda hapus dari pemantauan.



- Untuk menonaktifkan pemantauan untuk seluruh komponen, hapus kotak centang Pemantauan.



- Setelah selesai menyunting komponen, pilih Simpan perubahan yang ada di sudut kanan bawah. Setiap perubahan pada beban kerja untuk sebuah komponen akan bisa Anda lihat pada tabel Tinjauan komponen untuk pemantauan pada Beban kerja terkait.

8. Pada halaman Tinjau komponen yang terdeteksi, pilih Berikutnya.

9. Halaman Tentukan detail komponen mencakup semua komponen-komponen dengan beban kerja terkait yang dapat disesuaikan dari langkah sebelumnya.

   > **Note**
   >
   > Jika judul komponen memiliki tanda opsional, maka detail tambahan untuk beban kerja dalam komponen tersebut akan bersifat opsional.

   Jika sebuah komponen tidak muncul di halaman ini, artinya komponen tersebut tidak memiliki detail tambahan yang dapat ditentukan dalam langkah ini.

10. Pilih Berikutnya.
11. Pada halaman Tinjau dan kirim, lakukan peninjauan pada semua komponen dan detail beban kerja yang dipantau.
12. Pilih Kirim.

Account-based application monitoring

1. Nama aplikasi Masukkan sebuah nama untuk aplikasi berbasis akun Anda.
2. Pemantauan otomatis sumber daya baru. Secara default, Wawasan Aplikasi menggunakan pengaturan yang disarankan untuk mengonfigurasi pemantauan komponen sumber daya yang ditambahkan ke akun Anda setelah Anda melakukan onboarding untuk aplikasi. Anda dapat mengecualikan pemantauan untuk sumber daya yang ditambahkan setelah melakukan onboarding untuk aplikasi Anda dengan mengosongkan kotak centang.
3. Pantau CloudWatch Acara. Pilih kotak centang untuk mengintegrasikan pemantauan Wawasan Aplikasi dengan CloudWatch Acara untuk mendapatkan wawasan dari Amazon EBS, Amazon EC2, Amazon ECS AWS CodeDeploy, AWS Health API dan Pemberitahuan, Amazon RDS, Amazon S3, dan. AWS Step Functions
4. Integrasi dengan AWS Systems Manager OpsCenter Untuk melihat dan mendapatkan pemberitahuan saat masalah terdeteksi untuk aplikasi yang dipilih, pilih kotak centang Generate Systems Manager OpsCenter OpsItems for remedial actions. Untuk melacak operasi yang diambil untuk menyelesaikan item kerja operasional (OpsItems) yang terkait dengan AWS sumber daya Anda, berikan topik SNS ARN.

5. Tag - opsional. CloudWatch Application Insights mendukung grup sumber daya CloudFormation berbasis tag dan berbasis (dengan pengecualian grup Auto Scaling). Untuk informasi selengkapnya, silakan lihat Bekerja dengan Penyunting Tag.
6. Sumber daya yang ditemukan. Semua sumber daya yang ditemukan di akun Anda akan ditambahkan ke daftar ini. Jika Wawasan Aplikasi tidak dapat menemukan semua sumber daya yang ada di akun Anda, maka pesan kesalahan akan muncul di bagian atas halaman. Pesan ini menyertakan sebuah tautan ke dokumentasi tentang cara menambahkan izin yang diperlukan.
7. Pilih Berikutnya.

   Sebuah ARN dibuat untuk aplikasi dalam format berikut.

   ```
   arn:partition:applicationinsights:region:account-id:application/
   TBD/application-name
   ```

   Contoh

   ```
   arn:aws:applicationinsights:us-east-1:123456789012:application/TBD/my-
   application
   ```

4. Setelah Anda mengirimkan konfigurasi pemantauan aplikasi Anda, Anda akan dibawa ke halaman detail untuk aplikasi tersebut, di mana Anda dapat melihat Ringkasan aplikasi, daftar Komponen yang dipantau dan Komponen yang tidak dipantau, dan, dengan memilih tab yang ada di sebelah Komponen, Riwayat konfigurasi, Pola log, dan Tanda apa pun yang telah Anda terapkan.

   Untuk menampilkan wawasan aplikasi, pilih Tampilkan Wawasan.

   Anda dapat memperbarui pilihan untuk pemantauan dan integrasi CloudWatch Acara dengan AWS Systems Manager OpsCenter dengan memilih Edit.

   Pada Komponen, Anda dapat memilih menu Tindakan untuk Membuat, Mengubah, atau Melakukan ungroup pada sebuah grup instans.

   Anda dapat mengelola pemantauan untuk komponen, termasuk tingkatan aplikasi, grup log, log peristiwa, metrik, dan alarm kustom, dengan memilih bullet yang ada di sebelah komponen dan memilih Kelola pemantauan.

Aktifkan Wawasan Aplikasi untuk melakukan pemantauan sumber daya Amazon ECS dan Amazon EKS

Anda dapat mengaktifkan Wawasan Aplikasi untuk melakukan pemantauan terhadap aplikasi dan layanan mikro terkontainer dari konsol Wawasan Kontainer. Wawasan Aplikasi mendukung pemantauan untuk sumber daya berikut ini:

- Klaster-klaster Amazon ECS
- Layanan-layanan Amazon ECS
- Tugas-tugas Amazon ECS
- Klaster-klaster Amazon EKS

Saat Application Insights diaktifkan, Application Insights menyediakan metrik dan log yang direkomendasikan, mendeteksi potensi masalah, menghasilkan CloudWatch Peristiwa, dan membuat dasbor otomatis untuk aplikasi dan layanan mikro Anda yang terkontainer.

Anda dapat mengaktifkan Wawasan Aplikasi untuk sumber daya terkontainer dari Wawasan Kontainer atau konsol Wawasan Aplikasi.

Mengaktifkan Wawasan Aplikasi dari konsol Wawasan Kontainer

Dari konsol Wawasan Kontainer, di dasbor Pemantauan performa Wawasan Kontainer, pilih Konfigurasi Wawasan Aplikasi Secara Otomatis. Ketika Wawasan Aplikasi sudah diaktifkan, ia akan menampilkan detail tentang masalah-masalah yang terdeteksi.

Mengaktifkan Wawasan Aplikasi dari konsol Wawasan Aplikasi

Saat klaster ECS muncul di daftar komponen, Wawasan Aplikasi akan secara otomatis mengaktifkan pemantauan kontainer tambahan dengan Wawasan Kontainer.

Untuk klaster EKS, Anda dapat mengaktifkan pemantauan tambahan dengan Wawasan Kontainer untuk menyediakan informasi-informasi diagnostik, seperti kegagalan mulai ulang kontainer, untuk membantu Anda mengisolasi dan mengatasi masalah-masalah yang terjadi. Langkah-langkah tambahan yang perlu dilakukan untuk menyiapkan Wawasan Kontainer untuk EKS. Untuk selengkapnya, silakan lihat Menyiapkan Wawasan Kontainer di Amazon EKS dan Kubernetes untuk langkah-langkah menyiapkan Wawasan Kontainer di EKS.

Pemantauan tambahan untuk EKS dengan Wawasan Kontainer didukung pada instans Linux dengan EKS.

Untuk informasi selengkapnya mengenai dukungan Wawasan Kontainer untuk klaster ECS dan EKS, silakan lihat Wawasan Kontainer.

Menonaktifkan pemantauan untuk sebuah komponen aplikasi

Untuk menonaktifkan pemantauan untuk sebuah komponen aplikasi, dari halaman detail aplikasi, pilih komponen yang ingin Anda nonaktifkan pemantauannya. Pilih Tindakan, kemudian pilih Hapus dari pemantauan.

Menghapus sebuah aplikasi

Untuk menghapus aplikasi, dari CloudWatch dasbor, di panel navigasi kiri, pilih Wawasan Aplikasi di bawah Pemantauan infrastruktur. Pilih aplikasi yang ingin Anda hapus. Pada Tindakan, pilih Hapus aplikasi. Langkah ini akan menghapus pemantauan dan menghapus semua monitor yang disimpan untuk komponen-komponen aplikasi. Sumber daya aplikasi tidak dihapus.

Menyiapkan, mengonfigurasi, dan mengelola aplikasi Anda untuk pemantauan dengan menggunakan baris perintah

Bagian ini menyediakan langkah-langkah untuk menyiapkan, mengonfigurasi, dan mengelola aplikasi Anda untuk pemantauan menggunakan AWS CLI dan AWS Tools for Windows PowerShell.

Prosedur baris perintah

- Menambahkan dan mengelola sebuah aplikasi
- Mengelola dan memperbarui pemantauan
- Mengonfigurasi pemantauan untuk SQL Always On Availability Groups
- Mengonfigurasi pemantauan untuk MySQL RDS
- Mengonfigurasi pemantauan untuk MySQL EC2
- Mengonfigurasi pemantauan untuk PostgreSQL RDS
- Mengonfigurasi pemantauan untuk PostgreSQL EC2
- Mengonfigurasi pemantauan untuk Oracle RDS
- Mengonfigurasi pemantauan untuk Oracle EC2

Menambahkan dan mengelola sebuah aplikasi

Anda dapat menambahkan, mendapatkan informasi, mengelola, dan mengonfigurasi aplikasi Wawasan Aplikasi Anda dengan menggunakan baris perintah.

Topik

Menambahkan sebuah aplikasi

Tambahkan aplikasi menggunakan AWS CLI

Untuk menggunakan AWS CLI untuk menambahkan aplikasi untuk grup sumber daya Anda yang dipanggil`my-resource-group`, dengan OpsCenter diaktifkan untuk mengirimkan OpsitEm yang dibuat ke topik SNS `arn:aws:sns:us-east-1:123456789012:MyTopic` ARN, gunakan perintah berikut.

```
aws application-insights create-application --resource-group-name my-resource-group --
ops-center-enabled --ops-item-sns-topic-arn arn:aws:sns:us-east-1:123456789012:MyTopic
```

Tambahkan aplikasi menggunakan AWS Tools for Windows PowerShell

Untuk menggunakan AWS Tools for Windows PowerShell untuk menambahkan aplikasi untuk grup sumber daya Anda yang dipanggil `my-resource-group` dengan OpsCenter diaktifkan untuk mengirimkan OpsitEm yang dibuat ke topik SNS `arn:aws:sns:us-east-1:123456789012:MyTopic` ARN, gunakan perintah berikut.

```
New-CWAIApplication -ResourceGroupName my-resource-group -OpsCenterEnabled true -
OpsItemSNSTopicArn arn:aws:sns:us-east-1:123456789012:MyTopic
```

Mendeskripsikan sebuah aplikasi

Jelaskan aplikasi menggunakan AWS CLI

Untuk menggunakan AWS CLI untuk menggambarkan aplikasi yang dibuat pada kelompok sumber daya yang disebut`my-resource-group`, gunakan perintah berikut.

```
aws application-insights describe-application --resource-group-name my-resource-group
```

Jelaskan aplikasi menggunakan AWS Tools for Windows PowerShell

Untuk menggunakan AWS Tools for Windows PowerShell untuk menggambarkan aplikasi yang dibuat pada kelompok sumber daya yang disebut`my-resource-group`, gunakan perintah berikut.

```
Get-CWAIApplication -ResourceGroupName my-resource-group
```

Membuat daftar komponen dalam sebuah aplikasi

Daftar komponen dalam aplikasi menggunakan AWS CLI

Untuk menggunakan daftar komponen yang dibuat pada grup sumber daya yang disebut`my-resource-group`, gunakan perintah berikut. AWS CLI

```
aws application-insights list-components --resource-group-name my-resource-group
```

Daftar komponen dalam aplikasi menggunakan AWS Tools for Windows PowerShell

Untuk menggunakan daftar komponen yang dibuat pada grup sumber daya yang disebut`my-resource-group`, gunakan perintah berikut. AWS Tools for Windows PowerShell

```
Get-CWAIComponentList -ResourceGroupName my-resource-group
```

Mendeskripsikan sebuah komponen

Jelaskan komponen menggunakan AWS CLI

Anda dapat menggunakan AWS CLI perintah berikut untuk mendeskripsikan komponen `my-component` yang disebut milik aplikasi yang dibuat pada grup sumber daya yang disebut`my-resource-group`.

```
aws application-insights describe-component --resource-group-name my-resource-group --
component-name my-component
```

Jelaskan komponen menggunakan AWS Tools for Windows PowerShell

Anda dapat menggunakan AWS Tools for Windows PowerShell perintah berikut untuk mendeskripsikan komponen `my-component` yang disebut milik aplikasi yang dibuat pada grup sumber daya yang disebut`my-resource-group`.

```
Get-CWAIComponent -ComponentName my-component -ResourceGroupName my-resource-group
```

Mengelompokkan sumber daya serupa ke dalam sebuah komponen kustom

Kami merekomendasikan Anda untuk melakukan pengelompokan sumber daya yang serupa, seperti instans server web .NET, ke dalam komponen-komponen kustom sehingga Anda bisa melakukan onboarding dengan lebih mudah dan melakukan pemantauan serta wawasan yang lebih baik. Saat ini, CloudWatch Application Insights mendukung grup kustom untuk instans EC2.

Untuk mengelompokkan sumber daya ke dalam sebuah komponen kustom dengan menggunakan AWS CLI

Untuk menggunakan AWS CLI untuk mengelompokkan tiga instance (`arn:aws:ec2:us-east-1:123456789012:instance/i-11111`, `arn:aws:ec2:us-east-1:123456789012:instance/i-22222`, dan `arn:aws:ec2:us-east-1:123456789012:instance/i-33333`) bersama-sama ke dalam komponen kustom yang dipanggil `my-component` untuk aplikasi yang dibuat untuk grup sumber daya yang disebut`my-resource-group`, gunakan perintah berikut.

```
aws application-insights create-component --resource-group-name my-
resource-group --component-name my-component --resource-list arn:aws:ec2:us-
east-1:123456789012:instance/i-11111 arn:aws:ec2:us-east-1:123456789012:instance/
i-22222 arn:aws:ec2:us-east-1:123456789012:instance/i-33333
```

Untuk mengelompokkan sumber daya ke dalam sebuah komponen kustom menggunakan AWS Tools for Windows PowerShell

Untuk digunakan AWS Tools for Windows PowerShell untuk mengelompokkan tiga instance (`arn:aws:ec2:us-east-1:123456789012:instance/i-11111`,`arn:aws:ec2:us-east-1:123456789012:instance/i-22222`, dan`arn:aws:ec2:us-east-1:123456789012:instance/i-33333`) bersama-sama ke dalam komponen kustom yang disebut`my-component`, untuk aplikasi yang dibuat untuk grup sumber daya yang disebut`my-resource-group`, gunakan perintah berikut.

```
New-CWAIComponent -ResourceGroupName my-resource-group -ComponentName my-component
 -ResourceList arn:aws:ec2:us-east-1:123456789012:instance/i-11111,arn:aws:ec2:us-
east-1:123456789012:instance/i-22222,arn:aws:ec2:us-east-1:123456789012:instance/
i-33333
```

Memisahkan grup sebuah komponen kustom

Untuk memisahkan komponen kustom menggunakan AWS CLI

Untuk menggunakan AWS CLI unroup komponen kustom bernama `my-component` dalam aplikasi yang dibuat pada grup sumber dayamy`-resource-group`, gunakan perintah berikut.

```
aws application-insights delete-component --resource-group-name my-resource-group --
component-name my-new-component
```

Untuk memisahkan komponen kustom menggunakan AWS Tools for Windows PowerShell

Untuk menggunakan AWS Tools for Windows PowerShell unroup komponen kustom bernama `my-component` dalam aplikasi yang dibuat pada grup sumber dayamy`-resource-group`, gunakan perintah berikut.

```
Remove-CWAIComponent -ComponentName my-component -ResourceGroupName my-resource-group
```

Memperbarui sebuah aplikasi

Perbarui aplikasi menggunakan AWS CLI

Anda dapat menggunakan AWS CLI untuk memperbarui aplikasi untuk menghasilkan AWS Systems Manager OpsCenter OpsItems untuk masalah yang terdeteksi dengan aplikasi, dan untuk mengaitkan yang dibuat OpsItems ke topik SNS `arn:aws:sns:us-east-1:123456789012:MyTopic`, menggunakan perintah berikut.

```
aws application-insights update-application --resource-group-name my-resource-group --
ops-center-enabled --ops-item-sns-topic-arn arn:aws:sns:us-east-1:123456789012:MyTopic
```

Perbarui aplikasi menggunakan AWS Alat untuk Windows PowerShell

Anda dapat menggunakan AWS Tools for Windows PowerShell untuk memperbarui aplikasi untuk menghasilkan AWS SSM OpsCenter OpsItems untuk masalah yang terdeteksi dengan aplikasi, dan untuk mengaitkan yang dibuat OpsItems ke topik SNS `arn:aws:sns:us-east-1:123456789012:MyTopic`, menggunakan perintah berikut.

```
Update-CWAIApplication -ResourceGroupName my-resource-group -OpsCenterEnabled true -
OpsItemSNSTopicArn arn:aws:sns:us-east-1:123456789012:MyTopic
```

Memperbarui sebuah komponen kustom

Perbarui komponen kustom menggunakan AWS CLI

Anda dapat menggunakan AWS CLI untuk memperbarui komponen kustom yang dipanggil `my-component` dengan nama komponen baru`my-new-component`, dan grup instance yang diperbarui, dengan menggunakan perintah berikut.

```
aws application-insights update-component --resource-group-name my-resource-group --component-name my-component --new-component-name my-new-component --resource-list arn:aws:ec2:us-east-1:123456789012:instance/i-44444 arn:aws:ec2:us-east-1:123456789012:instance/i-55555
```

Perbarui komponen khusus menggunakan AWS Alat untuk Windows PowerShell

Anda dapat menggunakan AWS Tools for Windows PowerShell untuk memperbarui komponen kustom yang dipanggil `my-component` dengan nama komponen baru`my-new-component`, dan grup instance yang diperbarui, dengan menggunakan perintah berikut.

```
Update-CWAIComponent -ComponentName my-component -NewComponentName my-new-component -ResourceGroupName my-resource-group -ResourceList arn:aws:ec2:us-east-1:123456789012:instance/i-44444,arn:aws:ec2:us-east-1:123456789012:instance/i-55555
```

Mengelola dan memperbarui pemantauan

Anda dapat mengelola dan memperbarui pemantauan untuk aplikasi Wawasan Aplikasi Anda dengan menggunakan baris perintah.

Topik

- Membuat daftar masalah yang terjadi pada aplikasi Anda
- Mendeskripsikan sebuah masalah aplikasi
- Mendeskripsikan anomali atau kesalahan terkait sebuah masalah
- Membuat deskripsi anomali atau kesalahan yang terjadi pada aplikasi
- Mendeskripsikan konfigurasi pemantauan suatu komponen
- Mendeskripsikan konfigurasi pemantauan suatu komponen yang direkomendasikan
- Memperbarui konfigurasi pemantauan untuk sebuah komponen
- Menghapus sebuah grup sumber daya tertentu dari pemantauan Wawasan Aplikasi

Membuat daftar masalah yang terjadi pada aplikasi Anda

Buat daftar masalah dengan aplikasi Anda menggunakan AWS CLI

Untuk menggunakan daftar masalah dengan aplikasi Anda yang terdeteksi antara 1.000 dan 10.000 milidetik sejak Unix Epoch untuk aplikasi yang dibuat pada grup sumber daya yang disebut`my-resource-group`, gunakan perintah berikut. AWS CLI

```
aws application-insights list-problems --resource-group-name my-resource-group --start-time 1000 --end-time 10000
```

Buat daftar masalah dengan aplikasi Anda menggunakan AWS Tools untuk Windows PowerShell

Untuk menggunakan daftar masalah dengan aplikasi Anda yang terdeteksi antara 1.000 dan 10.000 milidetik sejak Unix Epoch untuk aplikasi yang dibuat pada grup sumber daya yang disebut`my-resource-group`, gunakan perintah berikut. AWS Tools for Windows PowerShell

```
$startDate = "8/6/2019 3:33:00"
$endDate = "8/6/2019 3:34:00"
Get-CWAIProblemList -ResourceGroupName my-resource-group -StartTime $startDate -EndTime $endDate
```

Mendeskripsikan sebuah masalah aplikasi

Jelaskan masalah aplikasi menggunakan AWS CLI

Untuk menggunakan AWS CLI untuk menggambarkan masalah dengan masalah idp-1234567890, gunakan perintah berikut.

```
aws application-insights describe-problem –problem-id p-1234567890
```

Jelaskan masalah aplikasi menggunakan AWS Tools untuk Windows PowerShell

Untuk menggunakan AWS Tools for Windows PowerShell untuk menggambarkan masalah dengan masalah idp-1234567890, gunakan perintah berikut.

```
Get-CWAIProblem -ProblemId p-1234567890
```

Mendeskripsikan anomali atau kesalahan terkait sebuah masalah

Jelaskan anomali atau kesalahan yang terkait dengan masalah menggunakan AWS CLI

Untuk menggunakan AWS CLI untuk menggambarkan anomali atau kesalahan yang terkait dengan masalah dengan id masalah`p-1234567890`, gunakan perintah berikut.

```
aws application-insights describe-problem-observations --problem-id -1234567890
```

Mendeskripsikan anomali atau kesalahan terkait sebuah masalah dengan menggunakan AWS Tools for Windows PowerShell

Untuk menggunakan AWS Tools for Windows PowerShell untuk menggambarkan anomali atau kesalahan yang terkait dengan masalah dengan id masalah`p-1234567890`, gunakan perintah berikut.

```
Get-CWAIProblemObservation -ProblemId p-1234567890
```

Membuat deskripsi anomali atau kesalahan yang terjadi pada aplikasi

Membuat deskripsi anomali atau kesalahan yang terjadi pada aplikasi dengan menggunakan CLI AWS

Untuk menggunakan AWS CLI untuk menggambarkan anomali atau kesalahan dengan aplikasi dengan id observasio`-1234567890`, gunakan perintah berikut.

```
aws application-insights describe-observation –observation-id o-1234567890
```

Jelaskan anomali atau kesalahan dengan aplikasi menggunakan AWS Tools untuk Windows PowerShell

Untuk menggunakan AWS Tools for Windows PowerShell untuk menggambarkan anomali atau kesalahan dengan aplikasi dengan id observasio`-1234567890`, gunakan perintah berikut.

```
Get-CWAIObservation -ObservationId o-1234567890
```

Mendeskripsikan konfigurasi pemantauan suatu komponen

Mendeskripsikan konfigurasi pemantauan suatu komponen dengan menggunakan AWS CLI

Untuk menggunakan AWS CLI untuk menggambarkan konfigurasi pemantauan komponen yang disebut `my-component` dalam aplikasi yang dibuat pada grup sumber dayamy`-resource-group`, gunakan perintah berikut.

```
aws application-insights describe-component-configuration –resource-group-name my-resource-group –component-name my-component
```

Jelaskan konfigurasi pemantauan komponen menggunakan AWS Alat untuk Windows PowerShell

Untuk menggunakan AWS Tools for Windows PowerShell untuk menggambarkan konfigurasi pemantauan komponen yang disebut`my-component`, dalam aplikasi yang dibuat pada grup sumber dayamy`-resource-group`, gunakan perintah berikut.

```
Get-CWAIComponentConfiguration -ComponentName my-component -ResourceGroupName my-resource-group
```

Untuk informasi lebih lanjut tentang konfigurasi komponen dan misalnya file JSON, lihat Cara menggunakan konfigurasi komponen.

Mendeskripsikan konfigurasi pemantauan suatu komponen yang direkomendasikan

Jelaskan konfigurasi pemantauan komponen yang direkomendasikan menggunakan AWS CLI

Ketika komponen merupakan bagian dari aplikasi.NET Worker, Anda dapat menggunakan AWS CLI untuk menggambarkan konfigurasi pemantauan yang disarankan dari komponen yang disebut `my-component` dalam aplikasi yang dibuat pada grup sumber dayamy`-resource-group`, dengan menggunakan perintah berikut.

```
aws application-insights describe-component-configuration-recommendation --resource-group-name my-resource-group --component-name my-component --tier DOT_NET_WORKER
```

Jelaskan konfigurasi pemantauan yang direkomendasikan dari komponen yang menggunakan AWS Tools for Windows PowerShell

Ketika komponen merupakan bagian dari aplikasi.NET Worker, Anda dapat menggunakan AWS Tools for Windows PowerShell untuk menggambarkan konfigurasi pemantauan yang disarankan dari komponen yang disebut `my-component` dalam aplikasi yang dibuat pada grup sumber dayamy`-resource-group`, dengan menggunakan perintah berikut.

```
Get-CWAIComponentConfigurationRecommendation -ComponentName my-component -ResourceGroupName my-resource-group -Tier DOT_NET_WORKER
```

Untuk informasi lebih lanjut tentang konfigurasi komponen dan misalnya file JSON, lihat Cara menggunakan konfigurasi komponen.

Memperbarui konfigurasi pemantauan untuk sebuah komponen

Memperbarui konfigurasi pemantauan untuk sebuah komponen dengan menggunakan AWS CLI

Untuk menggunakan AWS CLI untuk memperbarui komponen yang disebut `my-component` dalam aplikasi yang dibuat pada kelompok sumber daya yang disebut`my-resource-group`, gunakan perintah berikut. Perintah ini mencakup tindakan-tindakan berikut:

1. Mengaktifkan pemantauan komponen.
2. Mengatur tingkatan komponen menjadi `.NET Worker`.
3. Memperbarui konfigurasi JSON komponen untuk dibaca dari file lokal `configuration.txt`.

```
aws application-insights update-component-configuration --resource-group-name my-
resource-group --component-name my-component --tier DOT_NET_WORKER --monitor --
component-configuration "file://configuration.txt"
```

Memperbarui konfigurasi pemantauan untuk sebuah komponen dengan menggunakan AWS Tools for Windows PowerShell

Untuk menggunakan AWS Tools for Windows PowerShell untuk memperbarui komponen yang disebut `my-component` dalam aplikasi yang dibuat pada kelompok sumber daya yang disebut`my-resource-group`, gunakan perintah berikut. Perintah ini mencakup tindakan-tindakan berikut:

1. Mengaktifkan pemantauan komponen.
2. Mengatur tingkatan komponen menjadi `.NET Worker`.
3. Memperbarui konfigurasi JSON komponen untuk dibaca dari file lokal `configuration.txt`.

```
[string]$config = Get-Content -Path configuration.txt
Update-CWAIComponentConfiguration -ComponentName my-component -ResourceGroupName my-
resource-group -Tier DOT_NET_WORKER -Monitor 1 -ComponentConfiguration $config
```

Untuk informasi selengkapnya tentang konfigurasi komponen dan contoh file JSON, silakan lihat Cara menggunakan konfigurasi komponen.

Menghapus sebuah grup sumber daya tertentu dari pemantauan Wawasan Aplikasi

Menghapus grup sumber daya tertentu dari pemantauan Application Insights menggunakan AWS CLI

Untuk menggunakan AWS CLI untuk menghapus aplikasi yang dibuat pada kelompok sumber daya yang dipanggil `my-resource-group` dari pemantauan, gunakan perintah berikut.

```
aws application-insights delete-application --resource-group-name my-resource-group
```

Menghapus grup sumber daya tertentu dari pemantauan Application Insights menggunakan AWS Tools for Windows PowerShell

Untuk menggunakan AWS Tools for Windows PowerShell untuk menghapus aplikasi yang dibuat pada kelompok sumber daya yang dipanggil `my-resource-group` dari pemantauan, gunakan perintah berikut.

```
Remove-CWAIApplication -ResourceGroupName my-resource-group
```

Mengonfigurasi pemantauan untuk SQL Always On Availability Groups

1. Membuat sebuah aplikasi untuk grup sumber daya dengan instans SQL HA EC2.

   ```
   aws application-insights create-application --region <REGION> --resource-group-name
     <RESOURCE_GROUP_NAME>
   ```

2. Menentukan instans EC2 yang mewakili klaster SQL HA dengan membuat sebuah komponen aplikasi baru.

   ```
   aws application-insights create-component --resource-group-name
    "<RESOURCE_GROUP_NAME>" --component-name SQL_HA_CLUSTER --resource-list
     "arn:aws:ec2:<REGION>:<ACCOUNT_ID>:instance/<CLUSTER_INSTANCE_1_ID>"
    "arn:aws:ec2:<REGION>:<ACCOUNT_ID>:instance/<CLUSTER_INSTANCE_2_ID>
   ```

3. Mengonfigurasi komponen SQL HA.

   ```
   aws application-insights  update-component-configuration --resource-group-name
    "<RESOURCE_GROUP_NAME>" --region <REGION> --component-name "SQL_HA_CLUSTER" --
   monitor --tier SQL_SERVER_ALWAYSON_AVAILABILITY_GROUP --monitor  --component-
   configuration '{
     "subComponents" : [ {
       "subComponentType" : "AWS::EC2::Instance",
       "alarmMetrics" : [ {
         "alarmMetricName" : "CPUUtilization",
         "monitor" : true
       }, {
   ```

```
      "alarmMetricName" : "StatusCheckFailed",
      "monitor" : true
    }, {
      "alarmMetricName" : "Processor % Processor Time",
      "monitor" : true
    }, {
      "alarmMetricName" : "Memory % Committed Bytes In Use",
      "monitor" : true
    }, {
      "alarmMetricName" : "Memory Available Mbytes",
      "monitor" : true
    }, {
      "alarmMetricName" : "Paging File % Usage",
      "monitor" : true
    }, {
      "alarmMetricName" : "System Processor Queue Length",
      "monitor" : true
    }, {
      "alarmMetricName" : "Network Interface Bytes Total/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "PhysicalDisk % Disk Time",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Buffer Manager Buffer cache hit ratio",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Buffer Manager Page life expectancy",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:General Statistics Processes blocked",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:General Statistics User Connections",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Locks Number of Deadlocks/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:SQL Statistics Batch Requests/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica File Bytes Received/sec",
      "monitor" : true
```

```
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Log Bytes Received/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Log remaining for undo",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Log Send Queue",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Mirrored Write Transaction/
sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Recovery Queue",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Redo Bytes Remaining",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Redone Bytes/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Total Log requiring undo",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Transaction Delay",
      "monitor" : true
    } ],
    "windowsEvents" : [ {
      "logGroupName" : "WINDOWS_EVENTS-Application-<RESOURCE_GROUP_NAME>",
      "eventName" : "Application",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL", "INFORMATION" ],
      "monitor" : true
    }, {
      "logGroupName" : "WINDOWS_EVENTS-System-<RESOURCE_GROUP_NAME>",
      "eventName" : "System",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL" ],
      "monitor" : true
    }, {
      "logGroupName" : "WINDOWS_EVENTS-Security-<RESOURCE_GROUP_NAME>",
      "eventName" : "Security",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL" ],
      "monitor" : true
```

```
    } ],
    "logs" : [ {
      "logGroupName" : "SQL_SERVER_ALWAYSON_AVAILABILITY_GROUP-
<RESOURCE_GROUP_NAME>",
      "logPath" : "C:\\Program Files\\Microsoft SQL Server\\MSSQL**.MSSQLSERVER\
\MSSQL\\Log\\ERRORLOG",
      "logType" : "SQL_SERVER",
      "monitor" : true,
      "encoding" : "utf-8"
    } ]
  }, {
    "subComponentType" : "AWS::EC2::Volume",
    "alarmMetrics" : [ {
      "alarmMetricName" : "VolumeReadBytes",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeWriteBytes",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeReadOps",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeWriteOps",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeQueueLength",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeThroughputPercentage",
      "monitor" : true
    }, {
    "alarmMetricName" : "BurstBalance",
      "monitor" : true
    } ]
  } ]
}'
```

> **Note**
>
> Wawasan Aplikasi harus menyerap log Peristiwa Aplikasi (tingkat informasi) untuk mendeteksi aktivitas-aktivitas klaster, misalnya failover.

Mengonfigurasi pemantauan untuk MySQL RDS

1. Membuat sebuah aplikasi untuk grup sumber daya dengan instans basis data MySQL RDS.

```
aws application-insights create-application --region <REGION> --resource-group-name
  <RESOURCE_GROUP_NAME>
```

2. Log kesalahan diaktifkan secara bawaan. Log kueri lambat dapat diaktifkan dengan menggunakan grup parameter data. Untuk informasi selengkapnya, silakan lihat Mengakses Kueri Lambat MySQL dan Log Umum.
   - `set slow_query_log = 1`
   - `set log_output = FILE`
3. Ekspor log yang akan dipantau ke CloudWatch log. Untuk informasi selengkapnya, lihat Menerbitkan Log MySQL ke Log. CloudWatch
4. Mengonfigurasi komponen MySQL RDS.

```
aws application-insights  update-component-configuration --resource-group-name
 "<RESOURCE_GROUP_NAME>" --region <REGION> --component-name "<DB_COMPONENT_NAME>"
 --monitor --tier DEFAULT --monitor  --component-configuration "{\"alarmMetrics\":
[{\"alarmMetricName\":\"CPUUtilization\",\"monitor\":true}],\"logs\":[{\"logType\":
\"MYSQL\",\"monitor\":true},{\"logType\": \"MYSQL_SLOW_QUERY\",\"monitor\":false}]}"
```

Mengonfigurasi pemantauan untuk MySQL EC2

1. Membuat sebuah aplikasi untuk grup sumber daya dengan instans SQL HA EC2.

```
aws application-insights create-application --region <REGION> --resource-group-name
  <RESOURCE_GROUP_NAME>
```

2. Log kesalahan diaktifkan secara bawaan. Log kueri lambat dapat diaktifkan dengan menggunakan grup parameter data. Untuk informasi selengkapnya, silakan lihat Mengakses Kueri Lambat MySQL dan Log Umum.
   - `set slow_query_log = 1`
   - `set log_output = FILE`
3. Mengonfigurasi komponen MySQL EC2.

```
aws application-insights  update-component-configuration --resource-group-name
 "<RESOURCE_GROUP_NAME>" --region <REGION> --component-name "<DB_COMPONENT_NAME>"
 --monitor --tier MYSQL --monitor  --component-configuration "{\"alarmMetrics\":
[{\"alarmMetricName\":\"CPUUtilization\",\"monitor\":true}],\"logs\":[{\"logGroupName
\":\"<UNIQUE_LOG_GROUP_NAME>\",\"logPath\":\"C:\\\\ProgramData\\\\MySQL\\\\MySQL
 Server **\\\\Data\\\\<FILE_NAME>.err\",\"logType\":\"MYSQL\",\"monitor\":true,
\"encoding\":\"utf-8\"}]}"
```

Mengonfigurasi pemantauan untuk PostgreSQL RDS

1. Membuat sebuah aplikasi untuk grup sumber daya dengan instans basis data PostgreSQL RDS.

```
aws application-insights create-application --region <REGION> --resource-group-name
  <RESOURCE_GROUP_NAME>
```

2. Menerbitkan log PostgreSQL CloudWatch ke tidak diaktifkan secara default. Untuk mengaktifkan pemantauan, buka konsol RDS kemudian pilih basis data yang akan dipantau. Pilih Modifikasi yang ada di sudut kanan atas, dan pilih kotak centang berlabel log PostgreSQL. Pilih Lanjutkan untuk menyimpan pengaturan ini.
3. Log PostgreSQL Anda diekspor ke. CloudWatch
4. Mengonfigurasi komponen PostgreSQL RDS.

```
aws application-insights update-component-configuration --region <REGION> --resource-
group-name <RESOURCE_GROUP_NAME> --component-name <DB_COMPONENT_NAME> --monitor --
tier DEFAULT --component-configuration
"{
   \"alarmMetrics\":[
      {
         \"alarmMetricName\": \"CPUUtilization\",
         \"monitor\": true
      }
   ],
   \"logs\":[
      {
         \"logType\": \"POSTGRESQL\",
         \"monitor\": true
      }
   ]
}"
```

Mengonfigurasi pemantauan untuk PostgreSQL EC2

1. Membuat sebuah aplikasi untuk grup sumber daya dengan instans basis data PostgreSQL EC2.

```
aws application-insights create-application --region <REGION> --resource-group-name
  <RESOURCE_GROUP_NAME>
```

2. Mengonfigurasi komponen PostgreSQL EC2.

```
aws application-insights update-component-configuration --region <REGION> --resource-
group-name <RESOURCE_GROUP_NAME> --component-name <DB_COMPONENT_NAME> --monitor --
tier POSTGRESQL --component-configuration
"{
   \"alarmMetrics\":[
      {
         \"alarmMetricName\":\"CPUUtilization\",
         \"monitor\":true
      }
   ],
   \"logs\":[
      {
         \"logGroupName\":\"<UNIQUE_LOG_GROUP_NAME>\",
         \"logPath\":\"/var/lib/pgsql/data/log/\",
         \"logType\":\"POSTGRESQL\",
         \"monitor\":true,
         \"encoding\":\"utf-8\"
      }
   ]
}"
```

Mengonfigurasi pemantauan untuk Oracle RDS

1. Membuat sebuah aplikasi untuk grup sumber daya dengan instans basis data Oracle RDS.

```
aws application-insights create-application --region <REGION> --resource-group-name
  <RESOURCE_GROUP_NAME>
```

2. Menerbitkan log Oracle ke tidak CloudWatch diaktifkan secara default. Untuk mengaktifkan pemantauan, buka konsol RDS kemudian pilih basis data yang akan dipantau. Pilih Modifikasi yang ada di sudut kanan atas, kemudian pilih kotak centang berlabel Peringatan dan log Pendengar. Pilih Lanjutkan untuk menyimpan pengaturan ini.

3. Log Oracle Anda diekspor ke. CloudWatch
4. Mengonfigurasi komponen Oracle RDS.

```
aws application-insights update-component-configuration --region <REGION> --resource-
group-name <RESOURCE_GROUP_NAME> --component-name <DB_COMPONENT_NAME> --monitor --
tier DEFAULT --component-configuration
"{
   \"alarmMetrics\":[
      {
         \"alarmMetricName\": \"CPUUtilization\",
         \"monitor\": true
      }
   ],
   \"logs\":[
      {
         \"logType\": \"ORACLE_ALERT\",
         \"monitor\": true
      },
      {
         \"logType\": \"ORACLE_LISTENER\",
         \"monitor\": true
      }
   ]
}"
```

Mengonfigurasi pemantauan untuk Oracle EC2

1. Membuat sebuah aplikasi untuk grup sumber daya dengan instans Oracle EC2

```
aws application-insights create-application --region <REGION> --resource-group-name
  <RESOURCE_GROUP_NAME>
```

2. Mengonfigurasi komponen Oracle EC2.

```
aws application-insights update-component-configuration --region <REGION> --resource-
group-name <RESOURCE_GROUP_NAME> --component-name <DB_COMPONENT_NAME> --monitor --
tier ORACLE --component-configuration
"{
   \"alarmMetrics\":[
      {
         \"alarmMetricName\":\"CPUUtilization\",
```

```
         \"monitor\":true
      }
   ],
   \"logs\":[
      {
         \"logGroupName\":\"<UNIQUE_LOG_GROUP_NAME>\",
         \"logPath\":\"/opt/oracle/diag/rdbms/*/*/trace\",
         \"logType\":\"ORACLE_ALERT\",
         \"monitor\":true,
      },
      {
         \"logGroupName\":\"<UNIQUE_LOG_GROUP_NAME>\",
         \"logPath\":\"/opt/oracle/diag/tnslsnr/$HOSTNAME/listener/trace/\",
         \"logType\":\"ORACLE_ALERT\",
         \"monitor\":true,
      }
   ]
}"
```

Wawasan Aplikasi CloudWatch Acara dan pemberitahuan untuk masalah yang terdeteksi

Untuk setiap aplikasi yang ditambahkan ke CloudWatch Application Insights, sebuah CloudWatch acara diterbitkan untuk acara berikut dengan upaya terbaik:

- Penciptaan masalah. Dipancarkan saat CloudWatch Application Insights mendeteksi masalah baru.
  - Tipe Detail: "Masalah Wawasan Aplikasi Terdeteksi"
  - Detail:
    - `problemId`: ID masalah yang terdeteksi.
    - `region`: AWS Wilayah tempat masalah dibuat.
    - `resourceGroupName`: Grup Sumber Daya untuk aplikasi terdaftar yang masalahnya terdeteksi.
    - `status`: Status masalah. Kemungkinan status dan definisinya adalah sebagai berikut:
      - `In progress`: Sebuah masalah baru telah diidentifikasi. Masalah tersebut masih sedang diobservasi.
      - `Recovering`: Masalah berangsur stabil. Anda dapat menyelesaikan masalah secara manual saat masalah berada dalam status ini.
      - `Resolved`: Masalah teratasi. Tidak ada observasi baru tentang masalah ini.

    - `Recurring`: Masalah sudah diselesaikan dalam 24 jam terakhir. Masalah ini telah dibuka kembali sebagai hasil dari observasi tambahan.
  - `severity`: Tingkat keseriusan masalah.
  - `problemUrl`: URL konsol untuk masalah.

- Pembaruan masalah. Dipancarkan ketika masalah diperbarui dengan observasi baru atau ketika observasi yang ada diperbarui dan masalah kemudian diperbarui; pembaruan tersebut mencakup penyelesaian atau penutupan masalah.
  - Tipe Detail: "Masalah Wawasan Aplikasi Diperbarui"
  - Detail:
    - `problemId`: ID masalah yang dibuat.
    - `region`: AWS Wilayah tempat masalah dibuat.
    - `resourceGroupName`: Grup Sumber Daya untuk aplikasi terdaftar yang masalahnya terdeteksi.
    - `status`: Status masalah.
    - `severity`: Tingkat keseriusan masalah.
    - `problemUrl`: URL konsol untuk masalah.

Cara menerima notifikasi untuk peristiwa masalah yang dihasilkan oleh sebuah aplikasi

Dari CloudWatch konsol, pilih Aturan di bawah Acara di panel navigasi kiri. Dari halaman Aturan , pilih Buat aturan. Pilih Amazon CloudWatch Application Insights dari daftar dropdown Nama Layanan dan pilih Jenis Acara. Kemudian, pilih Tambahkan target dan pilih target dan parameter, sebagai contoh, Topik SNS atau Fungsi Lambda.

Tindakan melalui AWS Systems Manager. CloudWatch Application Insights menyediakan integrasi bawaan dengan Systems Manager OpsCenter. Jika Anda memilih untuk menggunakan integrasi ini untuk aplikasi Anda, sebuah OpsItem dibuat di OpsCenter konsol untuk setiap masalah yang terdeteksi dengan aplikasi. Dari OpsCenter konsol, Anda dapat melihat informasi yang diringkas tentang masalah yang terdeteksi oleh CloudWatch Application Insights dan memilih runbook Systems Manager Automation untuk mengambil tindakan perbaikan atau mengidentifikasi lebih lanjut proses Windows yang menyebabkan masalah sumber daya dalam aplikasi Anda.

Observabilitas lintas akun Wawasan Aplikasi

Dengan observabilitas lintas akun CloudWatch Application Insights, Anda dapat memantau dan memecahkan masalah aplikasi Anda yang menjangkau beberapa AWS akun dalam satu Wilayah.

Anda dapat menggunakan Amazon CloudWatch Observability Access Manager untuk menyiapkan satu atau beberapa AWS akun Anda sebagai akun pemantauan. Anda akan memberikan akun pemantauan tersebut kemampuan-kemampuan untuk melihat data yang ada di akun sumber Anda dengan membuat sebuah sink di akun pemantauan Anda. Anda menggunakan sink tersebut untuk membuat sebuah tautan dari akun sumber Anda ke akun pemantauan Anda. Untuk informasi selengkapnya, lihat CloudWatch observabilitas lintas akun.

Sumber daya yang dibutuhkan

Untuk fungsionalitas yang tepat dari observabilitas lintas akun CloudWatch Application Insights, pastikan bahwa jenis telemetri berikut dibagikan melalui Observability Access Manager. CloudWatch

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

Cara menggunakan konfigurasi komponen

Sebuah konfigurasi komponen adalah file teks dalam format JSON yang menjelaskan pengaturan konfigurasi komponen. Bagian ini akan memberikan untuk Anda contoh-contoh fragmen template, deskripsi bagian konfigurasi komponen, dan contoh konfigurasi komponen.

Topik

- Fragmen template konfigurasi komponen
- Bagian konfigurasi komponen
- Contoh-contoh konfigurasi komponen

Fragmen template konfigurasi komponen

Contoh berikut menunjukkan sebuah fragmen template dalam format JSON.

```
{
  "alarmMetrics" : [
    list of alarm metrics
  ],
  "logs" : [
    list of logs
  ],
  "processes" : [
   list of processes
  ],
  "windowsEvents" : [
    list of windows events channels configurations
  ],
  "alarms" : [
    list of CloudWatch alarms
  ],
  "jmxPrometheusExporter": {
    JMX Prometheus Exporter configuration
  },
  "hanaPrometheusExporter": {
      SAP HANA Prometheus Exporter configuration
  },
  "haClusterPrometheusExporter": {
      HA Cluster Prometheus Exporter configuration
  },
  "netWeaverPrometheusExporter": {
      SAP NetWeaver Prometheus Exporter configuration
  },
  "subComponents" : [
    {
      "subComponentType" : "AWS::EC2::Instance" ...
      component nested instances configuration
    },
    {
      "subComponentType" : "AWS::EC2::Volume" ...
      component nested volumes configuration
    }
  ]
}
```

Bagian konfigurasi komponen

Sebuah konfigurasi komponen mencakup beberapa bagian utama. Bagian-bagian dalam sebuah konfigurasi komponen dapat di buat daftarnya dengan urutan apa pun.

- alarmMetrics (opsional)

  Sebuah daftar metrik yang akan dipantau untuk komponen. Semua jenis komponen dapat memiliki sebuah bagian alarmMetrics.

- log (opsional)

  Sebuah daftar log yang akan dipantau untuk komponen. Hanya instans EC2 yang dapat memiliki bagian log.

- proses (opsional)

  Sebuah daftar proses yang akan dipantau untuk komponen. Hanya instans EC2 yang dapat memiliki bagian proses.

- subComponents (opsional)

  Instans tersarang dan konfigurasi subComponent volume untuk komponen. Jenis komponen berikut dapat memiliki bagian instans tersarang dan subComponents: ELB, ASG, instans EC2 yang dikelompokkan secara khusus, dan instans EC2.

- alarm (opsional)

  Sebuah daftar alarm yang akan dipantau untuk komponen. Semua jenis komponen dapat memiliki sebuah bagian alarm.

- windowsEvents (opsional)

  Sebuah daftar peristiwa windows yang akan dipantau untuk komponen. Hanya Windows pada instans EC2 yang memiliki bagian `windowsEvents`.

- JMX PrometheusExporter (opsional)

  Konfigurasi Pengekspor JMXPrometheus.

- hanaPrometheusExporter (opsional)

  Konfigurasi Pengekspor SAP HANA Prometheus.

- haClusterPrometheusEksportir (opsional)

Konfigurasi Pengekspor Klaster HA Prometheus.

- netWeaverPrometheusEksportir (opsional)

  Konfigurasi SAP NetWeaver Prometheus Exporter.

- sapAsePrometheusEksportir (opsional)

  Konfigurasi Pengekspor SAP ASE Prometheus.

Contoh berikut menunjukkan sintaks untuk fragmen bagian subComponents dalam format JSON.

```
[
  {
    "subComponentType" : "AWS::EC2::Instance",
    "alarmMetrics" : [
      list of alarm metrics
    ],
    "logs" : [
      list of logs
    ],
    "processes": [
      list of processes
    ],
    "windowsEvents" : [
      list of windows events channels configurations
    ]
  },
  {
    "subComponentType" : "AWS::EC2::Volume",
    "alarmMetrics" : [
      list of alarm metrics
    ]
  }
]
```

Sifat-sifat dari bagian konfigurasi komponen

Bagian ini menjelaskan sifat-sifat dari setiap bagian konfigurasi komponen.

Bagian-bagian

- Metrik

- Log
- Proses
- Pengekspor Prometheus JMX
- Pengekspor Prometheus HANA
- Pengekspor Prometheus Klaster HA
- NetWeaver Prometheus Eksportir
- Pengekspor SAP ASE Prometheus
- Peristiwa-peristiwa Windows
- Alarm

Metrik

Mendefinisikan sebuah metrik yang akan dipantau untuk komponen.

JSON

```
{
  "alarmMetricName" : "monitoredMetricName",
  "monitor" : true/false
}
```

Sifat-sifat

- alarmMetricName (diperlukan)

  Nama metrik yang akan dipantau untuk komponen. Untuk metrik-metrik yang didukung oleh Wawasan Aplikasi, silakan lihat Log dan metrik yang didukung oleh Amazon CloudWatch Application Insights.
- monitor (opsional)

  Boolean untuk menunjukkan apakah akan memantau metrik, atau tidak. Nilai default adalah `true`.

Log

Menentukan sebuah log yang akan dipantau untuk komponen.

JSON

```
{
  "logGroupName" : "logGroupName",
  "logPath" : "logPath",
  "logType" : "logType",
  "encoding" : "encodingType",
  "monitor" : true/false
}
```

Sifat-sifat

- logGroupName (diperlukan)

  Nama grup CloudWatch log yang akan dikaitkan dengan log yang dipantau. Untuk kendala nama grup log, lihat CreateLogGroup.

- LogPath (diperlukan untuk komponen instans EC2; tidak diperlukan untuk komponen yang tidak menggunakan CloudWatch Agen, seperti) AWS Lambda

  Jalur log yang akan dipantau. Jalur log harus berupa sebuah jalur file sistem Windows absolut. Untuk informasi selengkapnya, lihat File Konfigurasi CloudWatch Agen: Bagian Log.

- logType (wajib)

  Jenis log menentukan pola log yang dibandingkan dengan Wawasan Aplikasi yang menganalisis log. Jenis log dipilih dari pilihan-pilihan berikut:

  - SQL_SERVER
  - MYSQL
  - MYSQL_SLOW_QUERY
  - POSTGRESQL
  - ORACLE_ALERT
  - ORACLE_LISTENER
  - IIS
  - APPLICATION
  - WINDOWS_EVENTS
  - WINDOWS_EVENTS_ACTIVE_DIRECTORY
  - WINDOWS_EVENTS_DNS
  - WINDOWS_EVENTS_IIS

  - WINDOWS_EVENTS_SHAREPOINT
  - SQL_SERVER_ALWAYSON_AVAILABILITY_GROUP
  - SQL_SERVER_FAILOVER_CLUSTER_INSTANCE
  - DEFAULT
  - CUSTOM
  - STEP_FUNCTION
  - API_GATEWAY_ACCESS
  - API_GATEWAY_EXECUTION
  - SAP_HANA_LOGS
  - SAP_HANA_TRACE
  - SAP_HANA_HIGH_AVAILABILITY
  - SAP_NETWEAVER_DEV_TRACE_LOGS
  - PACEMAKER_HIGH_AVAILABILITY
- encoding (opsional)

  Jenis pengkodean log yang akan dipantau. Penyandian yang ditentukan harus dimasukkan dalam daftar CloudWatch pengkodean yang didukung agen. Jika tidak disediakan, CloudWatch Application Insights menggunakan pengkodean default tipe utf-8, kecuali untuk:

  - SQL_SERVER: pengkodean utf-16
  - IIS: pengkodean ascii
- monitor (opsional)

  Boolean yang menunjukkan apakah akan memantau log. Nilai bawaannya adalah `true`.

Proses

Menentukan sebuah proses yang akan dipantau untuk komponen.

JSON

```
{
  "processName" : "monitoredProcessName",
  "alarmMetrics" : [
      list of alarm metrics
  ]
```

```
}
```

Sifat-sifat

- processName (wajib)

  Nama proses yang akan dipantau untuk komponen. Nama proses tidak boleh memuat batang proses, seperti `sqlservr` atau `sqlservr.exe`.

- alarmMetrics (wajib)

  Sebuah daftar metrik untuk memantau proses ini. Untuk melihat metrik proses yang didukung oleh CloudWatch Application Insights, lihat. Amazon Elastic Compute Cloud (EC2)

Pengekspor Prometheus JMX

Menentukan pengaturan Pengekspor Prometheus JMX.

JSON

```
"JMXPrometheusExporter": {
  "jmxURL" : "JMX URL",
  "hostPort" : "The host and port",
  "prometheusPort" : "Target port to emit Prometheus metrics"
}
```

Sifat-sifat

- jmxURL (opsional)

  Sebuah URL JMX lengkap yang akan dihubungkan.

- hostPort (opsional)

  Host dan port yang akan dihubungkan melalui JMX jarak jauh. Hanya satu dari `jmxURL` dan `hostPort` yang dapat ditentukan.

- prometheusPort (opsional)

  Port target untuk yang menjadi tujuan untuk mengirim metrik-metrik Prometheus. Jika tidak ditentukan, port 9404 bawaan akan digunakan.

Pengekspor Prometheus HANA

Menentukan pengaturan Pengekspor Prometheus HANA.

JSON

```
"hanaPrometheusExporter": {
    "hanaSid": "SAP HANA  SID",
    "hanaPort": "HANA database port",
    "hanaSecretName": "HANA secret name",
    "prometheusPort": "Target port to emit Prometheus metrics"
}
```

Sifat-sifat

- hanaSid

  ID sistem SAP (SID) tiga karakter dari sistem SAP HANA.

- hanaPort

  Port basis data HANA dimana pengekspor akan menjalankan kueri terhadap metrik HANA.

- hanaSecretName

  AWS Secrets Manager Rahasia yang menyimpan HANA memantau kredensil pengguna. Pengekspor Prometheus HANA menggunakan kredensial ini untuk terhubung ke basis data dan menjalankan kueri terhadap metrik HANA.

- prometheusPort (opsional)

  Port target yang menjadi tujuan pengiriman metrik-metrik Prometheus. Jika tidak ditentukan, maka port bawaan 9668 akan digunakan.

Pengekspor Prometheus Klaster HA

Menentukan pengaturan Pengekspor Prometheus Klaster HA.

JSON

```
"haClusterPrometheusExporter": {
    "prometheusPort": "Target port to emit Prometheus metrics"
```

```
}
```

Sifat-sifat

- prometheusPort (opsional)

  Port target yang menjadi tujuan pengiriman metrik-metrik Prometheus. Jika tidak ditentukan, maka port bawaan 9664 akan digunakan.

NetWeaver Prometheus Eksportir

Mendefinisikan pengaturan Eksportir NetWeaver Prometheus.

JSON

```
"netWeaverPrometheusExporter": {
    "sapSid": "SAP NetWeaver  SID",
    "instanceNumbers": [ "Array of instance Numbers of SAP NetWeaver system "],
"prometheusPort": "Target port to emit Prometheus metrics"
}
```

Sifat-sifat

- sapSid

  3 karakter SAP system ID (SID) dari sistem SAP NetWeaver .
- instanceNumbers

  Array contoh Numbers dari NetWeaver sistem SAP.

  Contoh: `"instanceNumbers": [ "00", "01"]`
- prometheusPort (opsional)

  Port target yang menjadi tujuan untuk mengirim metrik-metrik Prometheus. Jika tidak ditentukan, maka port bawaan 9680 akan digunakan.

Pengekspor SAP ASE Prometheus

Menentukan pengaturan Pengekspor SAP ASE Prometheus.

JSON

```
"sapASEPrometheusExporter": {
    "sapAseSid": "SAP ASE SID",
    "sapAsePort": "SAP ASE database port",
    "sapAseSecretName": "SAP ASE secret name",
    "prometheusPort": "Target port to emit Prometheus metrics",
    "agreeToEnableASEMonitoring": true
}
```

Sifat-sifat

- sapAseSid

  ID sistem SAP (SID) tiga karakter dari sistem SAP ASE.

- sapAsePort

  Port basis data SAP ASE dimana pengekspor akan menjalankan kueri terhadap metrik-metrik ASE.

- sapAseSecretNama

  AWS Secrets Manager Rahasia yang menyimpan ASE memantau kredensil pengguna. Pengekspor SAP ASE Prometheus menggunakan kredensial ini untuk terhubung ke basis data dan menjalankan kueri terhadap metrik ASE.

- prometheusPort (opsional)

  Port target yang menjadi tujuan pengiriman metrik-metrik Prometheus. Jika tidak ditentukan, maka port bawaan 9399 akan digunakan. Jika ada ASE DB lain yang menggunakan port default, maka kita bisa menggunakan port 9499.

Peristiwa-peristiwa Windows

Menentukan Peristiwa-Peristiwa Windows yang akan dicatat log-nya.

JSON

```
{
  "logGroupName" : "logGroupName",
  "eventName" : "eventName",
  "eventLevels" : ["ERROR","WARNING","CRITICAL","INFORMATION","VERBOSE"],
  "monitor" : true/false
}
```

Sifat-sifat

- logGroupName (diperlukan)

  Nama grup CloudWatch log yang akan dikaitkan dengan log yang dipantau. Untuk kendala nama grup log, lihat CreateLogGroup.

- eventName (wajib)

  Jenis Peristiwa Windows yang akan dicatat log-nya. Ini setara dengan nama saluran log Peristiwa Windows. Misalnya, Sistem, Keamanan, CustomEventName, dll. Bidang ini diperlukan untuk setiap jenis peristiwa Windows yang akan dicatat log-nya.

- eventLevels (wajib)

  Tingkat peristiwa yang dicatat log-nya. Anda harus menentukan setiap level yang akan dicatat log-nya. Nilai yang mungkin termasuk adalah `INFORMATION`, `WARNING`, `ERROR`, `CRITICAL`, dan `VERBOSE`. Bidang ini diperlukan untuk setiap jenis Peristiwa Windows yang akan dicatat log-nya.

- monitor (opsional)

  Boolean yang menunjukkan apakah akan memantau log. Nilai bawaannya adalah `true`.

Alarm

Mendefinisikan CloudWatch alarm yang akan dipantau untuk komponen.

JSON

```
{
  "alarmName" : "monitoredAlarmName",
  "severity" : HIGH/MEDIUM/LOW
}
```

Sifat-sifat

- alarmName (wajib)

  Nama CloudWatch alarm yang akan dipantau untuk komponen.

- severity (opsional)

  Menunjukkan tingkat penghentian saat alarm berbunyi.

Contoh-contoh konfigurasi komponen

Contoh-contoh berikut menunjukkan konfigurasi komponen dalam format JSON untuk layanan-layanan yang relevan.

Contoh konfigurasi komponen

- Tabel Amazon DynamoDB
- Amazon EC2 Auto Scaling (ASG)
- Klaster Amazon EKS
- Instans Amazon Elastic Compute Cloud (EC2)
- Amazon Elastic Container Service (Amazon ECS)
- Layanan-layanan Amazon ECS
- Tugas-tugas Amazon ECS
- Amazon Elastic File System (Amazon EFS)
- Amazon FSx
- Amazon Relational Database Service (RDS) Aurora MySQL
- Instans Amazon Relational Database Service (RDS)
- Pemeriksaan kondisi kesehatan Amazon Route 53
- Zona yang di-hosting Amazon Route 53
- Amazon Route 53 Resolver titik akhir
- Amazon Route 53 Resolver konfigurasi pencatatan kueri
- Bucket Amazon S3
- Amazon Simple Queue Service (SQS)
- Topik Amazon SNS
- Amazon Virtual Private Cloud (Amazon VPC)
- Gateway Network Address Translation (NAT) Amazon VPC
- Tahapan-tahapan API REST API Gateway
- Penyeimbang Beban Elastis Aplikasi
- FungsiAWS Lambda
- AWS Network Firewall kelompok aturan
- AWS Network Firewall asosiasi kelompok aturan
- AWS Step Functions

- Instans-instans Amazon EC2 yang dikelompokkan pelanggan
- Penyeimbang Beban Elastis
- Java
- Kubernetes di Amazon EC2
- RDS MariaDB dan RDS MySQL
- RDS Oracle
- RDS PostgreSQL
- SAP ASE di Amazon EC2
- SAP ASE Ketersediaan Tinggi di Amazon EC2
- SAP HANA di Amazon EC2
- SAP HANA High Availability di Amazon EC2
- SAP NetWeaver di Amazon EC2
- Ketersediaan NetWeaver Tinggi SAP di Amazon EC2
- SQL Always On Availability Group
- Contoh instans klaster failover

Tabel Amazon DynamoDB

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk tabel Amazon DynamoDB.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "SystemErrors",
      "monitor": false
    },
    {
      "alarmMetricName": "UserErrors",
      "monitor": false
    },
    {
      "alarmMetricName": "ConsumedReadCapacityUnits",
      "monitor": false
    },
    {
      "alarmMetricName": "ConsumedWriteCapacityUnits",
```

```
      "monitor": false
    },
    {
      "alarmMetricName": "ReadThrottleEvents",
      "monitor": false
    },
    {
      "alarmMetricName": "WriteThrottleEvents",
      "monitor": false
    },
    {
      "alarmMetricName": "ConditionalCheckFailedRequests",
      "monitor": false
    },
    {
      "alarmMetricName": "TransactionConflict",
      "monitor": false
    }
  ],
  "logs": []
}
```

Amazon EC2 Auto Scaling (ASG)

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk Amazon EC2 Auto Scaling (ASG).

```
{
    "alarmMetrics" : [
      {
        "alarmMetricName" : "CPUCreditBalance"
      }, {
        "alarmMetricName" : "EBSIOBalance%"
      }
    ],
    "subComponents" : [
      {
        "subComponentType" : "AWS::EC2::Instance",
        "alarmMetrics" : [
          {
            "alarmMetricName" : "CPUUtilization"
          }, {
            "alarmMetricName" : "StatusCheckFailed"
```

```
          }
        ],
        "logs" : [
          {
            "logGroupName" : "my_log_group",
            "logPath" : "C:\\LogFolder\\*",
            "logType" : "APPLICATION"
          }
        ],
        "processes" : [
          {
            "processName" : "my_process",
            "alarmMetrics" : [
              {
                  "alarmMetricName" : "procstat cpu_usage",
                  "monitor" : true
              }, {
                  "alarmMetricName" : "procstat memory_rss",
                  "monitor" : true
              }
            ]
      }
  ],
        "windowsEvents" : [
          {
            "logGroupName" : "my_log_group_2",
            "eventName" : "Application",
            "eventLevels" : [ "ERROR", "WARNING", "CRITICAL" ]
          }
        ]
      }, {
        "subComponentType" : "AWS::EC2::Volume",
        "alarmMetrics" : [
          {
            "alarmMetricName" : "VolumeQueueLength"
          }, {
            "alarmMetricName" : "BurstBalance"
          }
        ]
      }
    ],
    "alarms" : [
      {
        "alarmName" : "my_asg_alarm",
```

```
        "severity" : "LOW"
      }
    ]
  }
```

Klaster Amazon EKS

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk klaster Amazon EKS.

```
{
    "alarmMetrics":[
       {
          "alarmMetricName": "cluster_failed_node_count",
          "monitor":true
       },
       {
          "alarmMetricName": "node_cpu_reserved_capacity",
          "monitor":true
       },
       {
          "alarmMetricName": "node_cpu_utilization",
          "monitor":true
       },
       {
          "alarmMetricName": "node_filesystem_utilization",
          "monitor":true
       },
       {
          "alarmMetricName": "node_memory_reserved_capacity",
          "monitor":true
       },
       {
          "alarmMetricName": "node_memory_utilization",
          "monitor":true
       },
       {
          "alarmMetricName": "node_network_total_bytes",
          "monitor":true
       },
       {
          "alarmMetricName": "pod_cpu_reserved_capacity",
          "monitor":true
```

```
        },
        {
           "alarmMetricName": "pod_cpu_utilization",
           "monitor":true
        },
        {
           "alarmMetricName": "pod_cpu_utilization_over_pod_limit",
           "monitor":true
        },
        {
           "alarmMetricName": "pod_memory_reserved_capacity",
           "monitor":true
        },
        {
           "alarmMetricName": "pod_memory_utilization",
           "monitor":true
        },
        {
           "alarmMetricName": "pod_memory_utilization_over_pod_limit",
           "monitor":true
        },
        {
           "alarmMetricName": "pod_network_rx_bytes",
           "monitor":true
        },
        {
           "alarmMetricName": "pod_network_tx_bytes",
           "monitor":true
        }
     ],
     "logs":[
        {
           "logGroupName": "/aws/containerinsights/kubernetes/application",
           "logType":"APPLICATION",
           "monitor":true,
           "encoding":"utf-8"
        }
     ],
     "subComponents":[
        {
           "subComponentType":"AWS::EC2::Instance",
           "alarmMetrics":[
              {
                 "alarmMetricName":"CPUUtilization",
```

```
                "monitor":true
            },
            {
                "alarmMetricName":"StatusCheckFailed",
                "monitor":true
            },
            {
                "alarmMetricName":"disk_used_percent",
                "monitor":true
            },
            {
                "alarmMetricName":"mem_used_percent",
                "monitor":true
            }
        ],
        "logs":[
            {
                "logGroupName":"APPLICATION-KubernetesClusterOnEC2-IAD",
                "logPath":"",
                "logType":"APPLICATION",
                "monitor":true,
                "encoding":"utf-8"
            }
        ],
        "processes" : [
          {
              "processName" : "my_process",
              "alarmMetrics" : [
                  {
                      "alarmMetricName" : "procstat cpu_usage",
                      "monitor" : true
                  }, {
                      "alarmMetricName" : "procstat memory_rss",
                      "monitor" : true
                  }
              ]
          }
      ],
        "windowsEvents":[
            {
                "logGroupName":"my_log_group_2",
                "eventName":"Application",
                "eventLevels":[
                   "ERROR",
```

```
                "WARNING",
                "CRITICAL"
             ],
             "monitor":true
          }
       ]
    },
    {
       "subComponentType":"AWS::AutoScaling::AutoScalingGroup",
       "alarmMetrics":[
          {
             "alarmMetricName":"CPUCreditBalance",
             "monitor":true
          },
          {
             "alarmMetricName":"EBSIOBalance%",
             "monitor":true
          }
       ]
    },
    {
       "subComponentType":"AWS::EC2::Volume",
       "alarmMetrics":[
          {
             "alarmMetricName":"VolumeReadBytes",
             "monitor":true
          },
          {
             "alarmMetricName":"VolumeWriteBytes",
             "monitor":true
          },
          {
             "alarmMetricName":"VolumeReadOps",
             "monitor":true
          },
          {
             "alarmMetricName":"VolumeWriteOps",
             "monitor":true
          },
          {
             "alarmMetricName":"VolumeQueueLength",
             "monitor":true
          },
          {
```

```
                "alarmMetricName":"BurstBalance",
                "monitor":true
            }
         ]
      }
   ]
}
```

> **Note**
>
> - Bagian `subComponents` dari `AWS::EC2::Instance`, `AWS::EC2::Volume`, dan `AWS::AutoScaling::AutoScalingGroup` hanya berlaku untuk klaster Amazon EKS yang berjalan pada tipe peluncuran EC2.
> - Bagian `windowsEvents` dari `AWS::EC2::Instance` di `subComponents` hanya berlaku untuk Windows yang berjalan di instans Amazon EC2.

Instans Amazon Elastic Compute Cloud (EC2)

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk sebuah instans Amazon EC2.

> **Important**
>
> Saat sebuah instans Amazon EC2 berada dalam status `stopped`, instans tersebut akan dihapus dari pemantauan. Ketika kembali ke `running` keadaan, itu ditambahkan ke daftar komponen yang tidak dipantau pada halaman Detail aplikasi konsol Application Insights. CloudWatch Jika pemantauan otomatis ats sumber daya baru diaktifkan untuk aplikasi, maka instans akan ditambahkan ke daftar Komponen yang dipantau. Namun demikian, log dan metrik tersebut disetel menjadi default untuk beban kerja. Konfigurasi log dan metrik sebelumnya tidak akan disimpan.

```
{
    "alarmMetrics" : [
      {
        "alarmMetricName" : "CPUUtilization",
        "monitor" : true
      }, {
```

```
        "alarmMetricName" : "StatusCheckFailed"
      }
    ],
    "logs" : [
      {
        "logGroupName" : "my_log_group",
        "logPath" : "C:\\LogFolder\\*",
        "logType" : "APPLICATION",
        "monitor" : true
      },
      {
        "logGroupName" : "my_log_group_2",
        "logPath" : "C:\\LogFolder2\\*",
        "logType" : "IIS",
        "encoding" : "utf-8"
      }
    ],
    "processes" : [
        {
            "processName" : "my_process",
            "alarmMetrics" : [
                {
                    "alarmMetricName" : "procstat cpu_usage",
                    "monitor" : true
                }, {
                    "alarmMetricName" : "procstat memory_rss",
                    "monitor" : true
                }
            ]
        }
    ],
    "windowsEvents" : [
      {
        "logGroupName" : "my_log_group_3",
        "eventName" : "Application",
        "eventLevels" : [ "ERROR", "WARNING", "CRITICAL" ],
        "monitor" : true
      }, {
        "logGroupName" : "my_log_group_4",
        "eventName" : "System",
        "eventLevels" : [ "ERROR", "WARNING", "CRITICAL" ],
        "monitor" : true
    }],
     "alarms" : [
```

```
      {
        "alarmName" : "my_instance_alarm_1",
        "severity" : "HIGH"
      },
      {
        "alarmName" : "my_instance_alarm_2",
        "severity" : "LOW"
      }
    ],
     "subComponents" : [
     {
       "subComponentType" : "AWS::EC2::Volume",
       "alarmMetrics" : [
       {
         "alarmMetricName" : "VolumeQueueLength",
         "monitor" : "true"
       },
       {
         "alarmMetricName" : "VolumeThroughputPercentage",
         "monitor" : "true"
       },
       {
         "alarmMetricName" : "BurstBalance",
         "monitor" : "true"
       }
    }]
  }
```

Amazon Elastic Container Service (Amazon ECS)

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk Amazon Elastic Container Service (Amazon ECS).

```
{
    "alarmMetrics":[
       {
          "alarmMetricName":"CpuUtilized",
          "monitor":true
       },
       {
          "alarmMetricName":"MemoryUtilized",
          "monitor":true
       },
```

```
        {
           "alarmMetricName":"NetworkRxBytes",
           "monitor":true
        },
        {
           "alarmMetricName":"NetworkTxBytes",
           "monitor":true
        },
        {
           "alarmMetricName":"RunningTaskCount",
           "monitor":true
        },
        {
           "alarmMetricName":"PendingTaskCount",
           "monitor":true
        },
        {
           "alarmMetricName":"StorageReadBytes",
           "monitor":true
        },
        {
           "alarmMetricName":"StorageWriteBytes",
           "monitor":true
        }
     ],
     "logs":[
        {
           "logGroupName":"/ecs/my-task-definition",
           "logType":"APPLICATION",
           "monitor":true
        }
     ],
     "subComponents":[
        {
           "subComponentType":"AWS::ElasticLoadBalancing::LoadBalancer",
           "alarmMetrics":[
              {
                 "alarmMetricName":"HTTPCode_Backend_4XX",
                 "monitor":true
              },
              {
                 "alarmMetricName":"HTTPCode_Backend_5XX",
                 "monitor":true
              },
```

```
            {
               "alarmMetricName":"Latency",
               "monitor":true
            },
            {
               "alarmMetricName":"SurgeQueueLength",
               "monitor":true
            },
            {
               "alarmMetricName":"UnHealthyHostCount",
               "monitor":true
            }
         ]
      },
      {
         "subComponentType":"AWS::ElasticLoadBalancingV2::LoadBalancer",
         "alarmMetrics":[
            {
               "alarmMetricName":"HTTPCode_Target_4XX_Count",
               "monitor":true
            },
            {
               "alarmMetricName":"HTTPCode_Target_5XX_Count",
               "monitor":true
            },
            {
               "alarmMetricName":"TargetResponseTime",
               "monitor":true
            },
            {
               "alarmMetricName":"UnHealthyHostCount",
               "monitor":true
            }
         ]
      },
      {
         "subComponentType":"AWS::EC2::Instance",
         "alarmMetrics":[
            {
               "alarmMetricName":"CPUUtilization",
               "monitor":true
            },
            {
               "alarmMetricName":"StatusCheckFailed",
```

```
                "monitor":true
            },
            {
                "alarmMetricName":"disk_used_percent",
                "monitor":true
            },
            {
                "alarmMetricName":"mem_used_percent",
                "monitor":true
            }
        ],
        "logs":[
            {
                "logGroupName":"my_log_group",
                "logPath":"/mylog/path",
                "logType":"APPLICATION",
                "monitor":true
            }
        ],
        "processes" : [
            {
                "processName" : "my_process",
                "alarmMetrics" : [
                    {
                        "alarmMetricName" : "procstat cpu_usage",
                        "monitor" : true
                    }, {
                        "alarmMetricName" : "procstat memory_rss",
                        "monitor" : true
                    }
                ]
            }
        ],
        "windowsEvents":[
            {
                "logGroupName":"my_log_group_2",
                "eventName":"Application",
                "eventLevels":[
                    "ERROR",
                    "WARNING",
                    "CRITICAL"
                ],
                "monitor":true
            }
```

```
            ]
         },
         {
            "subComponentType":"AWS::EC2::Volume",
            "alarmMetrics":[
               {
                  "alarmMetricName":"VolumeQueueLength",
                  "monitor":"true"
               },
               {
                  "alarmMetricName":"VolumeThroughputPercentage",
                  "monitor":"true"
               },
               {
                  "alarmMetricName":"BurstBalance",
                  "monitor":"true"
               }
            ]
         }
      ]
   }
```

> **Note**
>
> - Bagian `subComponents` dari `AWS::EC2::Instance` dan `AWS::EC2::Volume` hanya berlaku untuk klaster Amazon ECS dengan layanan ECS atau tugas ECS yang berjalan pada tipe peluncuran EC2.
> - Bagian `windowsEvents` dari `AWS::EC2::Instance` di `subComponents` hanya berlaku untuk Windows yang berjalan di instans Amazon EC2.

Layanan-layanan Amazon ECS

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk sebuah layanan Amazon ECS.

```
{
    "alarmMetrics":[
       {
          "alarmMetricName":"CPUUtilization",
          "monitor":true
```

```
        },
        {
           "alarmMetricName":"MemoryUtilization",
           "monitor":true
        },
        {
           "alarmMetricName":"CpuUtilized",
           "monitor":true
        },
        {
           "alarmMetricName":"MemoryUtilized",
           "monitor":true
        },
        {
           "alarmMetricName":"NetworkRxBytes",
           "monitor":true
        },
        {
           "alarmMetricName":"NetworkTxBytes",
           "monitor":true
        },
        {
           "alarmMetricName":"RunningTaskCount",
           "monitor":true
        },
        {
           "alarmMetricName":"PendingTaskCount",
           "monitor":true
        },
        {
           "alarmMetricName":"StorageReadBytes",
           "monitor":true
        },
        {
           "alarmMetricName":"StorageWriteBytes",
           "monitor":true
        }
     ],
     "logs":[
        {
           "logGroupName":"/ecs/my-task-definition",
           "logType":"APPLICATION",
           "monitor":true
        }
```

```
    ],
    "subComponents":[
       {
          "subComponentType":"AWS::ElasticLoadBalancing::LoadBalancer",
          "alarmMetrics":[
             {
                "alarmMetricName":"HTTPCode_Backend_4XX",
                "monitor":true
             },
             {
                "alarmMetricName":"HTTPCode_Backend_5XX",
                "monitor":true
             },
             {
                "alarmMetricName":"Latency",
                "monitor":true
             },
             {
                "alarmMetricName":"SurgeQueueLength",
                "monitor":true
             },
             {
                "alarmMetricName":"UnHealthyHostCount",
                "monitor":true
             }
          ]
       },
       {
          "subComponentType":"AWS::ElasticLoadBalancingV2::LoadBalancer",
          "alarmMetrics":[
             {
                "alarmMetricName":"HTTPCode_Target_4XX_Count",
                "monitor":true
             },
             {
                "alarmMetricName":"HTTPCode_Target_5XX_Count",
                "monitor":true
             },
             {
                "alarmMetricName":"TargetResponseTime",
                "monitor":true
             },
             {
                "alarmMetricName":"UnHealthyHostCount",
```

```
                "monitor":true
            }
        ]
    },
    {
        "subComponentType":"AWS::EC2::Instance",
        "alarmMetrics":[
            {
                "alarmMetricName":"CPUUtilization",
                "monitor":true
            },
            {
                "alarmMetricName":"StatusCheckFailed",
                "monitor":true
            },
            {
                "alarmMetricName":"disk_used_percent",
                "monitor":true
            },
            {
                "alarmMetricName":"mem_used_percent",
                "monitor":true
            }
        ],
        "logs":[
            {
                "logGroupName":"my_log_group",
                "logPath":"/mylog/path",
                "logType":"APPLICATION",
                "monitor":true
            }
        ],
        "processes" : [
            {
                "processName" : "my_process",
                "alarmMetrics" : [
                {
                    "alarmMetricName" : "procstat cpu_usage",
                    "monitor" : true
                }, {
                    "alarmMetricName" : "procstat memory_rss",
                    "monitor" : true
                }
          ]
```

```
        }
    ],
          "windowsEvents":[
             {
                "logGroupName":"my_log_group_2",
                "eventName":"Application",
                "eventLevels":[
                   "ERROR",
                   "WARNING",
                   "CRITICAL"
                ],
                "monitor":true
             }
          ]
       },
       {
          "subComponentType":"AWS::EC2::Volume",
          "alarmMetrics":[
             {
                "alarmMetricName":"VolumeQueueLength",
                "monitor":"true"
             },
             {
                "alarmMetricName":"VolumeThroughputPercentage",
                "monitor":"true"
             },
             {
                "alarmMetricName":"BurstBalance",
                "monitor":"true"
             }
          ]
       }
    ]
}
```

> **Note**
>
> - Bagian `subComponents` dari `AWS::EC2::Instance` dan `AWS::EC2::Volume` hanya berlaku untuk Amazon ECS yang berjalan pada tipe peluncuran EC2.
> - Bagian `windowsEvents` dari `AWS::EC2::Instance` di `subComponents` hanya berlaku untuk Windows yang berjalan di instans Amazon EC2.

Tugas-tugas Amazon ECS

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk sebuah tugas Amazon ECS.

```
{
    "logs":[
       {
          "logGroupName":"/ecs/my-task-definition",
          "logType":"APPLICATION",
          "monitor":true
       }
    ],
    "processes" : [
        {
            "processName" : "my_process",
            "alarmMetrics" : [
                {
                    "alarmMetricName" : "procstat cpu_usage",
                    "monitor" : true
                }, {
                    "alarmMetricName" : "procstat memory_rss",
                    "monitor" : true
                }
            ]
        }
    ]
 }
```

Amazon Elastic File System (Amazon EFS)

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk Amazon EFS.

```
{
   "alarmMetrics": [
     {
       "alarmMetricName": "BurstCreditBalance",
       "monitor": true
     },
     {
       "alarmMetricName": "PercentIOLimit",
       "monitor": true
     },
```

```
    {
      "alarmMetricName": "PermittedThroughput",
      "monitor": true
    },
    {
      "alarmMetricName": "MeteredIOBytes",
      "monitor": true
    },
    {
      "alarmMetricName": "TotalIOBytes",
      "monitor": true
    },
    {
      "alarmMetricName": "DataWriteIOBytes",
      "monitor": true
    },
    {
      "alarmMetricName": "DataReadIOBytes",
      "monitor": true
    },
    {
      "alarmMetricName": "MetadataIOBytes",
      "monitor": true
    },
    {
      "alarmMetricName": "ClientConnections",
      "monitor": true
    },
    {
      "alarmMetricName": "TimeSinceLastSync",
      "monitor": true
    },
    {
      "alarmMetricName": "Throughput",
      "monitor": true
    },
    {
      "alarmMetricName": "PercentageOfPermittedThroughputUtilization",
      "monitor": true
    },
    {
      "alarmMetricName": "ThroughputIOPS",
      "monitor": true
    },
```

```
    {
      "alarmMetricName": "PercentThroughputDataReadIOBytes",
      "monitor": true
    },
    {
      "alarmMetricName": "PercentThroughputDataWriteIOBytes",
      "monitor": true
    },
    {
      "alarmMetricName": "PercentageOfIOPSDataReadIOBytes",
      "monitor": true
    },
    {
      "alarmMetricName": "PercentageOfIOPSDataWriteIOBytes",
      "monitor": true
    },
    {
      "alarmMetricName": "AverageDataReadIOBytesSize",
      "monitor": true
    },
    {
      "alarmMetricName": "AverageDataWriteIOBytesSize",
      "monitor": true
    }
  ],
  "logs": [
   {
   "logGroupName": "/aws/efs/utils",
   "logType": "EFS_MOUNT_STATUS",
   "monitor": true,
   }
  ]
}
```

Amazon FSx

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk Amazon FSx.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "DataReadBytes",
      "monitor": true
    },
```

```
    {
      "alarmMetricName": "DataWriteBytes",
      "monitor": true
    },
    {
      "alarmMetricName": "DataReadOperations",
      "monitor": true
    },
    {
      "alarmMetricName": "DataWriteOperations",
      "monitor": true
    },
    {
      "alarmMetricName": "MetadataOperations",
      "monitor": true
    },
    {
      "alarmMetricName": "FreeStorageCapacity",
      "monitor": true
    }
  ]
}
```

Amazon Relational Database Service (RDS) Aurora MySQL

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk Amazon RDS Aurora MySQL.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "CPUUtilization",
      "monitor": true
    },
    {
      "alarmMetricName": "CommitLatency",
      "monitor": true
    }
  ],
  "logs": [
    {
      "logType": "MYSQL",
      "monitor": true,
```

```
    },
    {
      "logType": "MYSQL_SLOW_QUERY",
      "monitor": false
    }
  ]
}
```

Instans Amazon Relational Database Service (RDS)

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk sebuah instans Amazon RDS.

```
{
  "alarmMetrics" : [
    {
      "alarmMetricName" : "BurstBalance",
      "monitor" : true
    }, {
      "alarmMetricName" : "WriteThroughput",
      "monitor" : false
    }
  ],

  "alarms" : [
    {
      "alarmName" : "my_rds_instance_alarm",
      "severity" : "MEDIUM"
    }
  ]
}
```

Pemeriksaan kondisi kesehatan Amazon Route 53

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk pemeriksaan kondisi kesehatan Amazon Route 53.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "ChildHealthCheckHealthyCount",
      "monitor": true
    },
```

```
    {
      "alarmMetricName": "ConnectionTime",
      "monitor": true
    },
    {
      "alarmMetricName": "HealthCheckPercentageHealthy",
      "monitor": true
    },
    {
      "alarmMetricName": "HealthCheckStatus",
      "monitor": true
    },
    {
      "alarmMetricName": "SSLHandshakeTime",
      "monitor": true
    },
    {
      "alarmMetricName": "TimeToFirstByte",
      "monitor": true
    }
  ]
}
```

Zona yang di-hosting Amazon Route 53

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk zona yang di-hosting Amazon Route 53.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "DNSQueries",
      "monitor": true
    },
    {
      "alarmMetricName": "DNSSECInternalFailure",
      "monitor": true
    },
    {
      "alarmMetricName": "DNSSECKeySigningKeysNeedingAction",
      "monitor": true
    },
    {
```

```
      "alarmMetricName": "DNSSECKeySigningKeyMaxNeedingActionAge",
      "monitor": true
    },
    {
      "alarmMetricName": "DNSSECKeySigningKeyAge",
      "monitor": true
    }
  ],
  "logs": [
    {
      "logGroupName":"/hosted-zone/logs",
      "logType": "ROUTE53_DNS_PUBLIC_QUERY_LOGS",
      "monitor": true
    }
  ]
}
```

Amazon Route 53 Resolver titik akhir

Contoh berikut menunjukkan konfigurasi komponen dalam format JSON untuk Amazon Route 53 Resolver endpoint.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "EndpointHealthyENICount",
      "monitor": true
    },
    {
      "alarmMetricName": "EndpointUnHealthyENICount",
      "monitor": true
    },
    {
      "alarmMetricName": "InboundQueryVolume",
      "monitor": true
    },
    {
      "alarmMetricName": "OutboundQueryVolume",
      "monitor": true
    },
    {
      "alarmMetricName": "OutboundQueryAggregateVolume",
      "monitor": true
```

```
    }
  ]
}
```

Amazon Route 53 Resolver konfigurasi pencatatan kueri

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk konfigurasi pencatatan log kueri Amazon Route 53 Resolver .

```
{
  "logs": [
    {
      "logGroupName": "/resolver-query-log-config/logs",
      "logType": "ROUTE53_RESOLVER_QUERY_LOGS",
      "monitor": true
    }
  ]
}
```

Bucket Amazon S3

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON bucket Amazon S3.

```
{
    "alarmMetrics" : [
        {
            "alarmMetricName" : "ReplicationLatency",
            "monitor" : true
        }, {
            "alarmMetricName" : "5xxErrors",
            "monitor" : true
        }, {
            "alarmMetricName" : "BytesDownloaded"
            "monitor" : true
        }
    ]
}
```

Amazon Simple Queue Service (SQS)

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk Amazon Simple Queue Service.

```
{
  "alarmMetrics" : [
    {
      "alarmMetricName" : "ApproximateAgeOfOldestMessage"
    }, {
      "alarmMetricName" : "NumberOfEmptyReceives"
    }
  ],
  "alarms" : [
    {
      "alarmName" : "my_sqs_alarm",
      "severity" : "MEDIUM"
    }
  ]
}
```

Topik Amazon SNS

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk topik Amazon SNS.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "NumberOfNotificationsFailed",
      "monitor": true
    },
    {
      "alarmMetricName": "NumberOfNotificationsFilteredOut-InvalidAttributes",
      "monitor": true
    },
    {
      "alarmMetricName": "NumberOfNotificationsFilteredOut-NoMessageAttributes",
      "monitor": true
    },
    {
      "alarmMetricName": "NumberOfNotificationsFailedToRedriveToDlq",
      "monitor": true
    }
  ]
}
```

Amazon Virtual Private Cloud (Amazon VPC)

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk Amazon VPC.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "NetworkAddressUsage",
      "monitor": true
    },
    {
      "alarmMetricName": "NetworkAddressUsagePeered",
      "monitor": true
    },
    {
      "alarmMetricName": "VPCFirewallQueryVolume",
      "monitor": true
    }
  ]
}
```

Gateway Network Address Translation (NAT) Amazon VPC

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk NAT gateway.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "ErrorPortAllocation",
      "monitor": true
    },
    {
      "alarmMetricName": "IdleTimeoutCount",
      "monitor": true
    }
  ]
}
```

Tahapan-tahapan API REST API Gateway

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk tahapan API Gateway API REST.

```
{
     "alarmMetrics" : [
         {
             "alarmMetricName" : "4XXError",
             "monitor" : true
         },
         {
             "alarmMetricName" : "5XXError",
             "monitor" : true
         }
     ],
    "logs" : [
        {
            "logType" : "API_GATEWAY_EXECUTION",
            "monitor" : true
        },
        {
            "logType" : "API_GATEWAY_ACCESS",
            "monitor" : true
        }
    ]
}
```

Penyeimbang Beban Elastis Aplikasi

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk Penyeimbang Beban Elastis Aplikasi.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "ActiveConnectionCount",
    }, {
      "alarmMetricName": "TargetResponseTime"
    }
  ],
  "subComponents": [
    {
      "subComponentType": "AWS::EC2::Instance",
      "alarmMetrics": [
        {
          "alarmMetricName": "CPUUtilization",
        }, {
```

```
          "alarmMetricName": "StatusCheckFailed"
        }
      ],
      "logs": [
        {
          "logGroupName": "my_log_group",
          "logPath": "C:\\LogFolder\\*",
          "logType": "APPLICATION",
        }
      ],
      "windowsEvents": [
        {
          "logGroupName": "my_log_group_2",
          "eventName": "Application",
          "eventLevels": [ "ERROR", "WARNING", "CRITICAL" ]
        }
      ]
    }, {
      "subComponentType": "AWS::EC2::Volume",
      "alarmMetrics": [
        {
          "alarmMetricName": "VolumeQueueLength",
        }, {
          "alarmMetricName": "BurstBalance"
        }
      ]
    }
  ],

  "alarms": [
    {
      "alarmName": "my_alb_alarm",
      "severity": "LOW"
    }
  ]
}
```

FungsiAWS Lambda

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk Fungsi AWS Lambda .

```
{
```

```
  "alarmMetrics": [
    {
      "alarmMetricName": "Errors",
      "monitor": true
    },
    {
      "alarmMetricName": "Throttles",
      "monitor": true
    },
    {
      "alarmMetricName": "IteratorAge",
      "monitor": true
    },
    {
      "alarmMetricName": "Duration",
      "monitor": true
    }
  ],
  "logs": [
    {
      "logType": "DEFAULT",
      "monitor": true
    }
  ]
}
```

AWS Network Firewall kelompok aturan

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk grup aturan AWS Network Firewall .

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "FirewallRuleGroupQueryVolume",
      "monitor": true
    }
  ]
}
```

AWS Network Firewall asosiasi kelompok aturan

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk asosiasi grup aturan AWS Network Firewall .

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "FirewallRuleGroupQueryVolume",
      "monitor": true
    }
  ]
}
```

AWS Step Functions

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk AWS Step Functions.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "ExecutionsFailed",
      "monitor": true
    },
    {
      "alarmMetricName": "LambdaFunctionsFailed",
      "monitor": true
    },
    {
      "alarmMetricName": "ProvisionedRefillRate",
      "monitor": true
    }
  ],
  "logs": [
    {
     "logGroupName": "/aws/states/HelloWorld-Logs",
      "logType": "STEP_FUNCTION",
      "monitor": true,
    }
  ]
}
```

Instans-instans Amazon EC2 yang dikelompokkan pelanggan

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk instans Amazon EC2 yang dikelompokkan pelanggan.

```
{
    "subComponents": [
      {
        "subComponentType": "AWS::EC2::Instance",
        "alarmMetrics": [
          {
            "alarmMetricName": "CPUUtilization",
          },
          {
            "alarmMetricName": "StatusCheckFailed"
          }
        ],
        "logs": [
          {
            "logGroupName": "my_log_group",
            "logPath": "C:\\LogFolder\\*",
            "logType": "APPLICATION",
          }
        ],
        "processes": [
            {
                "processName": "my_process",
                "alarmMetrics": [
                    {
                        "alarmMetricName": "procstat cpu_usage",
                        "monitor": true
                    }, {
                        "alarmMetricName": "procstat memory_rss",
                        "monitor": true
                    }
                ]
            }
        ],
        "windowsEvents": [
          {
            "logGroupName": "my_log_group_2",
            "eventName": "Application",
            "eventLevels": [ "ERROR", "WARNING", "CRITICAL" ]
          }
```

```
        ]
      }, {
        "subComponentType": "AWS::EC2::Volume",
        "alarmMetrics": [
          {
            "alarmMetricName": "VolumeQueueLength",
          }, {
            "alarmMetricName": "BurstBalance"
          }
        ]
      }
    ],
    "alarms": [
      {
        "alarmName": "my_alarm",
        "severity": "MEDIUM"
      }
    ]
  }
```

Penyeimbang Beban Elastis

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk Penyeimbang Beban Elastis.

```
{
    "alarmMetrics": [
      {
        "alarmMetricName": "EstimatedALBActiveConnectionCount"
      }, {
        "alarmMetricName": "HTTPCode_Backend_5XX"
      }
    ],
    "subComponents": [
      {
        "subComponentType": "AWS::EC2::Instance",
        "alarmMetrics": [
          {
            "alarmMetricName": "CPUUtilization"
          }, {
            "alarmMetricName": "StatusCheckFailed"
          }
        ],
```

```
        "logs": [
          {
            "logGroupName": "my_log_group",
            "logPath": "C:\\LogFolder\\*",
            "logType": "APPLICATION"
          }
        ],
        "processes": [
          {
            "processName": "my_process",
            "alarmMetrics": [
                {
                  "alarmMetricName": "procstat cpu_usage",
                  "monitor": true
                }, {
                  "alarmMetricName": "procstat memory_rss",
                  "monitor": true
                }
             ]
          }
        ],
        "windowsEvents": [
          {
            "logGroupName": "my_log_group_2",
            "eventName": "Application",
            "eventLevels": [ "ERROR", "WARNING", "CRITICAL" ],
            "monitor": true
          }
        ]
      }, {
        "subComponentType": "AWS::EC2::Volume",
        "alarmMetrics": [
          {
            "alarmMetricName": "VolumeQueueLength"
          }, {
            "alarmMetricName": "BurstBalance"
          }
        ]
      }
    ],

    "alarms": [
      {
        "alarmName": "my_elb_alarm",
```

```
        "severity": "HIGH"
      }
    ]
  }
```

Java

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk Java.

```
{
  "alarmMetrics": [ {
    "alarmMetricName": "java_lang_threading_threadcount",
    "monitor": true
  },
  {
    "alarmMetricName": "java_lang_memory_heapmemoryusage_used",
    "monitor": true
  },
  {
    "alarmMetricName": "java_lang_memory_heapmemoryusage_committed",
    "monitor": true
  }],
  "logs": [ ],
  "JMXPrometheusExporter": {
      "hostPort": "8686",
      "prometheusPort": "9404"
  }
}
```

> **Note**
>
> Wawasan Aplikasi tidak mendukung untuk digunakan mengonfigurasi autentikasi untuk pengekspor Prometheus JMX. Untuk informasi tentang cara menyiapkan autentikasi, silakan lihat Konfigurasi contoh pengekspor Prometheus JMX.

Kubernetes di Amazon EC2

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk Kubernetes di Amazon EC2.

```
{
```

```
    "alarmMetrics":[
       {
          "alarmMetricName":"cluster_failed_node_count",
          "monitor":true
       },
       {
          "alarmMetricName":"node_cpu_reserved_capacity",
          "monitor":true
       },
       {
          "alarmMetricName":"node_cpu_utilization",
          "monitor":true
       },
       {
          "alarmMetricName":"node_filesystem_utilization",
          "monitor":true
       },
       {
          "alarmMetricName":"node_memory_reserved_capacity",
          "monitor":true
       },
       {
          "alarmMetricName":"node_memory_utilization",
          "monitor":true
       },
       {
          "alarmMetricName":"node_network_total_bytes",
          "monitor":true
       },
       {
          "alarmMetricName":"pod_cpu_reserved_capacity",
          "monitor":true
       },
       {
          "alarmMetricName":"pod_cpu_utilization",
          "monitor":true
       },
       {
          "alarmMetricName":"pod_cpu_utilization_over_pod_limit",
          "monitor":true
       },
       {
          "alarmMetricName":"pod_memory_reserved_capacity",
          "monitor":true
```

```
      },
      {
         "alarmMetricName":"pod_memory_utilization",
         "monitor":true
      },
      {
         "alarmMetricName":"pod_memory_utilization_over_pod_limit",
         "monitor":true
      },
      {
         "alarmMetricName":"pod_network_rx_bytes",
         "monitor":true
      },
      {
         "alarmMetricName":"pod_network_tx_bytes",
         "monitor":true
      }
   ],
   "logs":[
      {
         "logGroupName":"/aws/containerinsights/kubernetes/application",
         "logType":"APPLICATION",
         "monitor":true,
         "encoding":"utf-8"
      }
   ],
   "subComponents":[
      {
         "subComponentType":"AWS::EC2::Instance",
         "alarmMetrics":[
            {
               "alarmMetricName":"CPUUtilization",
               "monitor":true
            },
            {
               "alarmMetricName":"StatusCheckFailed",
               "monitor":true
            },
            {
               "alarmMetricName":"disk_used_percent",
               "monitor":true
            },
            {
               "alarmMetricName":"mem_used_percent",
```

```
                "monitor":true
            }
         ],
         "logs":[
            {
               "logGroupName":"APPLICATION-KubernetesClusterOnEC2-IAD",
               "logPath":"",
               "logType":"APPLICATION",
               "monitor":true,
               "encoding":"utf-8"
            }
         ],
         "processes" : [
           {
               "processName" : "my_process",
               "alarmMetrics" : [
                   {
                       "alarmMetricName" : "procstat cpu_usage",
                       "monitor" : true
                   }, {
                       "alarmMetricName" : "procstat memory_rss",
                       "monitor" : true
                   }
               ]
           }
       ]
      },
      {
         "subComponentType":"AWS::EC2::Volume",
         "alarmMetrics":[
            {
               "alarmMetricName":"VolumeReadBytes",
               "monitor":true
            },
            {
               "alarmMetricName":"VolumeWriteBytes",
               "monitor":true
            },
            {
               "alarmMetricName":"VolumeReadOps",
               "monitor":true
            },
            {
               "alarmMetricName":"VolumeWriteOps",
```

```
                "monitor":true
             },
             {
                "alarmMetricName":"VolumeQueueLength",
                "monitor":true
             },
             {
                "alarmMetricName":"BurstBalance",
                "monitor":true
             }
          ]
       }
    ]
 }
```

RDS MariaDB dan RDS MySQL

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk RDS MariaDB dan RDS MySQL.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "CPUUtilization",
      "monitor": true
    }
  ],
  "logs": [
    {
      "logType": "MYSQL",
      "monitor": true,
    },
    {
      "logType": "MYSQL_SLOW_QUERY",
      "monitor": false
    }
  ]
}
```

RDS Oracle

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk RDS Oracle.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "CPUUtilization",
      "monitor": true
    }
  ],
  "logs": [
    {
      "logType": "ORACLE_ALERT",
      "monitor": true,
    },
    {
      "logType": "ORACLE_LISTENER",
      "monitor": false
    }
  ]
}
```

RDS PostgreSQL

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk RDS PostgreSQL.

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "CPUUtilization",
      "monitor": true
    }
  ],
  "logs": [
    {
      "logType": "POSTGRESQL",
      "monitor": true
    }
  ]
}
```

SAP ASE di Amazon EC2

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk SAP ASE di Amazon EC2.

```
{
  "subComponents": [
    {
      "subComponentType": "AWS::EC2::Instance",
      "alarmMetrics": [
        {
          "alarmMetricName": "asedb_database_availability",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_trunc_log_on_chkpt_enabled",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_last_db_backup_age_in_days",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_last_transaction_log_backup_age_in_hours",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_suspected_database",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_db_space_usage_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_db_log_space_usage_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_locked_login",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_data_cache_hit_ratio",
```

```
          "monitor": true
        }
      ],
      "logs": [
        {
          "logGroupName": "SAP_ASE_SERVER_LOGS-my-resource-group",
          "logPath": "/sybase/SY2/ASE-*/install/SY2.log",
          "logType": "SAP_ASE_SERVER_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        },
        {
          "logGroupName": "SAP_ASE_BACKUP_SERVER_LOGS-my-resource-group",
          "logPath": "/sybase/SY2/ASE-*/install/SY2_BS.log",
          "logType": "SAP_ASE_BACKUP_SERVER_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        }
  ],
  "sapAsePrometheusExporter": {
    "sapAseSid": "ASE",
    "sapAsePort": "4901",
    "sapAseSecretName": "ASE_DB_CREDS",
    "prometheusPort": "9399",
    "agreeToEnableASEMonitoring": true
}
```

SAP ASE Ketersediaan Tinggi di Amazon EC2

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk SAP ASE High Availability di Amazon EC2.

```
{
  "subComponents": [
    {
      "subComponentType": "AWS::EC2::Instance",
      "alarmMetrics": [
        {
          "alarmMetricName": "asedb_database_availability",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_trunc_log_on_chkpt_enabled",
```

```
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_last_db_backup_age_in_days",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_last_transaction_log_backup_age_in_hours",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_suspected_database",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_db_space_usage_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_ha_replication_state",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_ha_replication_mode",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_ha_replication_latency_in_minutes",
          "monitor": true
        }
      ],
      "logs": [
        {
          "logGroupName": "SAP_ASE_SERVER_LOGS-my-resource-group",
          "logPath": "/sybase/SY2/ASE-*/install/SY2.log",
          "logType": "SAP_ASE_SERVER_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        },
        {
          "logGroupName": "SAP_ASE_BACKUP_SERVER_LOGS-my-resource-group",
          "logPath": "/sybase/SY2/ASE-*/install/SY2_BS.log",
          "logType": "SAP_ASE_BACKUP_SERVER_LOGS",
          "monitor": true,
```

```
          "encoding": "utf-8"
        },
        {
          "logGroupName": "SAP_ASE_REP_SERVER_LOGS-my-resource-group",
          "logPath": "/sybase/SY2/DM/repservername/repservername.log",
          "logType": "SAP_ASE_REP_SERVER_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        },
        {
          "logGroupName": "SAP_ASE_RMA_AGENT_LOGS-my-resource-group",
          "logPath": "/sybase/SY2/DM/RMA-*/instances/AgentContainer/logs/",
          "logType": "SAP_ASE_RMA_AGENT_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        },
        {
          "logGroupName": "SAP_ASE_FAULT_MANAGER_LOGS-my-resource-group",
          "logPath": "/opt/sap/FaultManager/dev_sybdbfm",
          "logType": "SAP_ASE_FAULT_MANAGER_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        }
  ],
  "sapAsePrometheusExporter": {
    "sapAseSid": "ASE",
    "sapAsePort": "4901",
    "sapAseSecretName": "ASE_DB_CREDS",
    "prometheusPort": "9399",
    "agreeToEnableASEMonitoring": true
}
```

SAP HANA di Amazon EC2

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk SAP HANA di Amazon EC2.

```
{
  "subComponents": [
    {
      "subComponentType": "AWS::EC2::Instance",
      "alarmMetrics": [
        {
```

```
          "alarmMetricName": "hanadb_server_startup_time_variations_seconds",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_level_5_alerts_count",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_level_4_alerts_count",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_out_of_memory_events_count",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_max_trigger_read_ratio_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_table_allocation_limit_used_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_cpu_usage_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_plan_cache_hit_ratio_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_last_data_backup_age_days",
          "monitor": true
        }
      ],
      "logs": [
        {
          "logGroupName": "SAP_HANA_TRACE-my-resourge-group",
          "logPath": "/usr/sap/HDB/HDB00/*/trace/*.trc",
          "logType": "SAP_HANA_TRACE",
          "monitor": true,
          "encoding": "utf-8"
        },
```

```
        {
          "logGroupName": "SAP_HANA_LOGS-my-resource-group",
          "logPath": "/usr/sap/HDB/HDB00/*/trace/*.log",
          "logType": "SAP_HANA_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        }
      ]
    }
  ],
  "hanaPrometheusExporter": {
    "hanaSid": "HDB",
    "hanaPort": "30013",
    "hanaSecretName": "HANA_DB_CREDS",
    "prometheusPort": "9668"
  }
}
```

SAP HANA High Availability di Amazon EC2

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk SAP HANA High Availability di Amazon EC2.

```
{
  "subComponents": [
    {
      "subComponentType": "AWS::EC2::Instance",
      "alarmMetrics": [
        {
          "alarmMetricName": "hanadb_server_startup_time_variations_seconds",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_level_5_alerts_count",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_level_4_alerts_count",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_out_of_memory_events_count",
          "monitor": true
```

```
        },
        {
          "alarmMetricName": "ha_cluster_pacemaker_stonith_enabled",
          "monitor": true
        }
      ],
      "logs": [
        {
          "logGroupName": "SAP_HANA_TRACE-my-resourge-group",
          "logPath": "/usr/sap/HDB/HDB00/*/trace/*.trc",
          "logType": "SAP_HANA_TRACE",
          "monitor": true,
          "encoding": "utf-8"
        },
        {
          "logGroupName": "SAP_HANA_HIGH_AVAILABILITY-my-resource-group",
          "logPath": "/var/log/pacemaker/pacemaker.log",
          "logType": "SAP_HANA_HIGH_AVAILABILITY",
          "monitor": true,
          "encoding": "utf-8"
        }
      ]
    }
  ],
  "hanaPrometheusExporter": {
    "hanaSid": "HDB",
    "hanaPort": "30013",
    "hanaSecretName": "HANA_DB_CREDS",
    "prometheusPort": "9668"
  },
  "haClusterPrometheusExporter": {
    "prometheusPort": "9664"
  }
}
```

SAP NetWeaver di Amazon EC2

Contoh berikut menunjukkan konfigurasi komponen dalam format JSON untuk SAP NetWeaver di Amazon EC2.

```
{
  "subComponents": [
    {
```

```
      "subComponentType": "AWS::EC2::Instance",
      "alarmMetrics": [
        {
          "alarmMetricName": "CPUUtilization",
          "monitor": true
        },
        {
          "alarmMetricName": "StatusCheckFailed",
          "monitor": true
        },
        {
          "alarmMetricName": "disk_used_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "mem_used_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_ResponseTime",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_ResponseTimeDialog",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_ResponseTimeDialogRFC",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_DBRequestTime",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_LongRunners",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_AbortedJobs",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_BasisSystem",
```

```
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_Database",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_Security",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_System",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_QueueTime",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_Availability",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_start_service_processes",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_dispatcher_queue_now",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_dispatcher_queue_max",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_enqueue_server_locks_max",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_enqueue_server_locks_now",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_enqueue_server_locks_state",
```

```
          "monitor": true
        },
        {
          "alarmMetricName": "sap_enqueue_server_replication_state",
          "monitor": true
        }
      ],
      "logs": [
        {
          "logGroupName": "SAP_NETWEAVER_DEV_TRACE_LOGS-NetWeaver-ML4",
          "logPath": "/usr/sap/ML4/*/work/dev_w*",
          "logType": "SAP_NETWEAVER_DEV_TRACE_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        }
      ]
    }
  ],
  "netWeaverPrometheusExporter": {
    "sapSid": "ML4",
    "instanceNumbers": [
      "00",
      "11"
    ],
    "prometheusPort": "9680"
  }
}
```

Ketersediaan NetWeaver Tinggi SAP di Amazon EC2

Contoh berikut menunjukkan konfigurasi komponen dalam format JSON untuk SAP NetWeaver High Availability di Amazon EC2.

```
{
  "subComponents": [
    {
      "subComponentType": "AWS::EC2::Instance",
      "alarmMetrics": [
        {
          "alarmMetricName": "ha_cluster_corosync_ring_errors",
          "monitor": true
        },
        {
```

```
            "alarmMetricName": "ha_cluster_pacemaker_fail_count",
            "monitor": true
          },
          {
            "alarmMetricName": "sap_HA_check_failover_config_state",
            "monitor": true
          },
          {
            "alarmMetricName": "sap_HA_get_failover_config_HAActive",
            "monitor": true
          },
          {
            "alarmMetricName": "sap_alerts_AbortedJobs",
            "monitor": true
          },
          {
            "alarmMetricName": "sap_alerts_Availability",
            "monitor": true
          },
          {
            "alarmMetricName": "sap_alerts_BasisSystem",
            "monitor": true
          },
          {
            "alarmMetricName": "sap_alerts_DBRequestTime",
            "monitor": true
          },
          {
            "alarmMetricName": "sap_alerts_Database",
            "monitor": true
          },
          {
            "alarmMetricName": "sap_alerts_FrontendResponseTime",
            "monitor": true
          },
          {
            "alarmMetricName": "sap_alerts_LongRunners",
            "monitor": true
          },
          {
            "alarmMetricName": "sap_alerts_QueueTime",
            "monitor": true
          },
          {
```

```
          "alarmMetricName": "sap_alerts_ResponseTime",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_ResponseTimeDialog",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_ResponseTimeDialogRFC",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_Security",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_Shortdumps",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_SqlError",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_System",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_enqueue_server_replication_state",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_start_service_processes",
          "monitor": true
        }
      ],
      "logs": [
        {
          "logGroupName": "SAP_NETWEAVER_DEV_TRACE_LOGS-NetWeaver-PR1",
          "logPath": "/usr/sap/<SID>/D*/work/dev_w*",
          "logType": "SAP_NETWEAVER_DEV_TRACE_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        }
```

```
      ]
    }
  ],
  "haClusterPrometheusExporter": {
    "prometheusPort": "9664"
  },
  "netWeaverPrometheusExporter": {
    "sapSid": "PR1",
    "instanceNumbers": [
      "11",
      "12"
    ],
    "prometheusPort": "9680"
  }
}
```

SQL Always On Availability Group

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk SQL Always On Availability Group.

```
{
  "subComponents" : [ {
    "subComponentType" : "AWS::EC2::Instance",
    "alarmMetrics" : [ {
      "alarmMetricName" : "CPUUtilization",
      "monitor" : true
    }, {
      "alarmMetricName" : "StatusCheckFailed",
      "monitor" : true
    }, {
      "alarmMetricName" : "Processor % Processor Time",
      "monitor" : true
    }, {
      "alarmMetricName" : "Memory % Committed Bytes In Use",
      "monitor" : true
    }, {
      "alarmMetricName" : "Memory Available Mbytes",
      "monitor" : true
    }, {
      "alarmMetricName" : "Paging File % Usage",
      "monitor" : true
    }, {
```

```
      "alarmMetricName" : "System Processor Queue Length",
      "monitor" : true
    }, {
      "alarmMetricName" : "Network Interface Bytes Total/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "PhysicalDisk % Disk Time",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Buffer Manager Buffer cache hit ratio",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Buffer Manager Page life expectancy",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:General Statistics Processes blocked",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:General Statistics User Connections",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Locks Number of Deadlocks/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:SQL Statistics Batch Requests/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica File Bytes Received/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Log Bytes Received/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Log remaining for undo",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Log Send Queue",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Mirrored Write Transaction/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Recovery Queue",
      "monitor" : true
```

```
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Redo Bytes Remaining",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Redone Bytes/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Total Log requiring undo",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Transaction Delay",
      "monitor" : true
    } ],
    "windowsEvents" : [ {
      "logGroupName" : "WINDOWS_EVENTS-Application-<RESOURCE_GROUP_NAME>",
      "eventName" : "Application",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL", "INFORMATION" ],
      "monitor" : true
    }, {
      "logGroupName" : "WINDOWS_EVENTS-System-<RESOURCE_GROUP_NAME>",
      "eventName" : "System",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL" ],
      "monitor" : true
    }, {
      "logGroupName" : "WINDOWS_EVENTS-Security-<RESOURCE_GROUP_NAME>",
      "eventName" : "Security",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL" ],
      "monitor" : true
    } ],
    "logs" : [ {
      "logGroupName" : "SQL_SERVER_ALWAYSON_AVAILABILITY_GROUP-<RESOURCE_GROUP_NAME>",
      "logPath" : "C:\\Program Files\\Microsoft SQL Server\\MSSQL**.MSSQLSERVER\\MSSQL\
\Log\\ERRORLOG",
      "logType" : "SQL_SERVER",
      "monitor" : true,
      "encoding" : "utf-8"
    } ]
  }, {
    "subComponentType" : "AWS::EC2::Volume",
    "alarmMetrics" : [ {
      "alarmMetricName" : "VolumeReadBytes",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeWriteBytes",
```

```
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeReadOps",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeWriteOps",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeQueueLength",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeThroughputPercentage",
      "monitor" : true
    }, {
    "alarmMetricName" : "BurstBalance",
      "monitor" : true
    } ]
  } ]
}
```

Contoh instans klaster failover

Contoh berikut menunjukkan sebuah konfigurasi komponen dalam format JSON untuk instans klaster failover.

```
{
  "subComponents" : [ {
    "subComponentType" : "AWS::EC2::Instance",
    "alarmMetrics" : [ {
      "alarmMetricName" : "CPUUtilization",
      "monitor" : true
    }, {
      "alarmMetricName" : "StatusCheckFailed",
      "monitor" : true
    }, {
      "alarmMetricName" : "Processor % Processor Time",
      "monitor" : true
    }, {
      "alarmMetricName" : "Memory % Committed Bytes In Use",
      "monitor" : true
    }, {
      "alarmMetricName" : "Memory Available Mbytes",
      "monitor" : true
```

```
    }, {
      "alarmMetricName" : "Paging File % Usage",
      "monitor" : true
    }, {
      "alarmMetricName" : "System Processor Queue Length",
      "monitor" : true
    }, {
      "alarmMetricName" : "Network Interface Bytes Total/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "PhysicalDisk % Disk Time",
      "monitor" : true
    }, {
      "alarmMetricName" : "Bytes Received/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Normal Messages Queue Length/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Urgent Message Queue Length/se",
      "monitor" : true
    }, {
      "alarmMetricName" : "Reconnect Count",
      "monitor" : true
    }, {
      "alarmMetricName" : "Unacknowledged Message Queue Length/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Messages Outstanding",
      "monitor" : true
    }, {
      "alarmMetricName" : "Messages Sent/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Database Update Messages/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Update Messages/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Flushes/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Crypto Checkpoints Saved/sec",
```

```
      "monitor" : true
    }, {
      "alarmMetricName" : "Crypto Checkpoints Restored/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Registry Checkpoints Restored/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Registry Checkpoints Saved/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Cluster API Calls/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Resource API Calls/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Cluster Handles/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Resource Handles/sec",
      "monitor" : true
    } ],
    "windowsEvents" : [ {
      "logGroupName" : "WINDOWS_EVENTS-Application-<RESOURCE_GROUP_NAME>",
      "eventName" : "Application",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL"],
      "monitor" : true
    }, {
      "logGroupName" : "WINDOWS_EVENTS-System-<RESOURCE_GROUP_NAME>",
      "eventName" : "System",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL", "INFORMATION" ],
      "monitor" : true
    }, {
      "logGroupName" : "WINDOWS_EVENTS-Security-<RESOURCE_GROUP_NAME>",
      "eventName" : "Security",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL" ],
      "monitor" : true
    } ],
    "logs" : [ {
      "logGroupName" : "SQL_SERVER_FAILOVER_CLUSTER_INSTANCE-<RESOURCE_GROUP_NAME>",
      "logPath" : "\\\\amznfsxjmzbykwn.mydomain.aws\\SQLDB\\MSSQL**.MSSQLSERVER\\MSSQL\
\Log\\ERRORLOG",
      "logType" : "SQL_SERVER",
```

```
      "monitor" : true,
      "encoding" : "utf-8"
    } ]
  }, {
    "subComponentType" : "AWS::EC2::Volume",
    "alarmMetrics" : [ {
      "alarmMetricName" : "VolumeReadBytes",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeWriteBytes",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeReadOps",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeWriteOps",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeQueueLength",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeThroughputPercentage",
      "monitor" : true
    }, {
    "alarmMetricName" : "BurstBalance",
      "monitor" : true
    } ]
  } ]
}
```

Membuat dan mengonfigurasi pemantauan Wawasan CloudWatch Aplikasi menggunakan templat CloudFormation

Anda dapat menambahkan pemantauan Application Insights, termasuk metrik utama dan telemetri, ke aplikasi, database, dan server web Anda, langsung dari template. AWS CloudFormation

Bagian ini menyediakan contoh AWS CloudFormation templat dalam format JSON dan YAMM untuk membantu Anda membuat dan mengonfigurasi pemantauan Wawasan Aplikasi.

Untuk melihat referensi sumber daya dan properti Wawasan Aplikasi di PanduanAWS CloudFormation Pengguna, lihat referensi jenis ApplicationInsights sumber daya.

Sampel template

- Buat aplikasi Application Insights untuk seluruh tumpukan AWS CloudFormation
- Membuat sebuah aplikasi Wawasan Aplikasi dengan pengaturan terperinci
- Membuat sebuah aplikasi Wawasan Aplikasi dengan konfigurasi komponen mode CUSTOM
- Membuat sebuah aplikasi Wawasan Aplikasi dengan konfigurasi komponen mode DEFAULT
- Membuat sebuah aplikasi Wawasan Aplikasi dengan konfigurasi komponen mode DEFAULT_WITH_OVERWRITE

Buat aplikasi Application Insights untuk seluruh tumpukan AWS CloudFormation

Untuk menerapkan template berikut, Anda harus membuat AWS sumber daya dan satu atau beberapa grup sumber daya untuk membuat aplikasi Application Insights untuk memantau sumber daya tersebut. Untuk informasi selengkapnya, silakan lihat Memulai dengan Grup Sumber Daya AWS.

Dua bagian pertama dari template berikut menentukan sebuah sumber daya dan sebuah grup sumber daya. Bagian terakhir dari template tersebut membuat sebuah aplikasi Wawasan Aplikasi untuk grup sumber daya, tetapi tidak mengonfigurasi aplikasi atau menerapkan pemantauan. Untuk informasi selengkapnya, lihat detail CreateApplicationperintah di Referensi API Amazon CloudWatch Application Insights.

Template dalam format JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Test Resource Group stack",
    "Resources": {
        "EC2Instance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "ImageId" : "ami-abcd1234efgh5678i",
                "SecurityGroupIds" : ["sg-abcd1234"]
            }
        },
        ...
        "ResourceGroup": {
            "Type": "AWS::ResourceGroups::Group",
            "Properties": {
```

```
                "Name": "my_resource_group"
            }
        },
        "AppInsightsApp": {
            "Type": "AWS::ApplicationInsights::Application",
            "Properties": {
                "ResourceGroupName": "my_resource_group"
            },
            "DependsOn" : "ResourceGroup"
        }
    }
}
```

Template dalam format YAML

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: Test Resource Group stack
Resources:
  EC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-abcd1234efgh5678i
      SecurityGroupIds:
      - sg-abcd1234
  ...
  ResourceGroup:
    Type: AWS::ResourceGroups::Group
    Properties:
      Name: my_resource_group
  AppInsightsApp:
    Type: AWS::ApplicationInsights::Application
    Properties:
      ResourceGroupName: my_resource_group
    DependsOn: ResourceGroup
```

Bagian template berikut menerapkan konfigurasi pemantauan bawaan ke aplikasi Wawasan Aplikasi. Untuk informasi selengkapnya, lihat detail CreateApplicationperintah di Referensi API Amazon CloudWatch Application Insights.

Saat `AutoConfigurationEnabled` diatur menjadi `true`, semua komponen aplikasi dikonfigurasi dengan pengaturan pemantauan yang disarankan untuk tingkat aplikasi

DEFAULT. Untuk informasi selengkapnya tentang pengaturan dan tingkatan ini, lihat DescribeComponentConfigurationRecommendationdan UpdateComponentConfigurationdi Referensi API Amazon CloudWatch Application Insights.

Template dalam format JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Test Application Insights Application stack",
    "Resources": {
        "AppInsightsApp": {
            "Type": "AWS::ApplicationInsights::Application",
            "Properties": {
                "ResourceGroupName": "my_resource_group",
                "AutoConfigurationEnabled": true
            }
        }
    }
}
```

Template dalam format YAML

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: Test Application Insights Application stack
Resources:
  AppInsightsApp:
    Type: AWS::ApplicationInsights::Application
    Properties:
      ResourceGroupName: my_resource_group
      AutoConfigurationEnabled: true
```

Membuat sebuah aplikasi Wawasan Aplikasi dengan pengaturan terperinci

Template berikut melakukan tindakan ini:

- Membuat aplikasi Application Insights dengan pemberitahuan CloudWatch Acara dan OpsCenter diaktifkan. Untuk informasi selengkapnya, lihat detail CreateApplicationperintah di Referensi API Amazon CloudWatch Application Insights.
- Menandai aplikasi dengan dua tag, salah satu tanda tidak memiliki nilai tag. Untuk informasi selengkapnya, lihat TagResourcedi Referensi API Amazon CloudWatch Application Insights.

- Membuat dua komponen grup instans kustom. Untuk informasi selengkapnya, lihat CreateComponentdi Referensi API Amazon CloudWatch Application Insights.
- Membuat dua set pola log. Untuk informasi selengkapnya, lihat CreateLogPatterndi Referensi API Amazon CloudWatch Application Insights.
- Atur `AutoConfigurationEnabled` menjadi `true`, yang akan mengonfigurasi semua komponen aplikasi dengan pengaturan pemantauan yang disarankan untuk tingkat `DEFAULT`. Untuk informasi selengkapnya, lihat DescribeComponentConfigurationRecommendationdi Referensi API Amazon CloudWatch Application Insights.

Template dalam format JSON

```
{
    "Type": "AWS::ApplicationInsights::Application",
    "Properties": {
        "ResourceGroupName": "my_resource_group",
        "CWEMonitorEnabled": true,
        "OpsCenterEnabled": true,
        "OpsItemSNSTopicArn": "arn:aws:sns:us-east-1:123456789012:my_topic",
        "AutoConfigurationEnabled": true,
        "Tags": [
            {
                "Key": "key1",
                "Value": "value1"
            },
            {
                "Key": "key2",
                "Value": ""
            }
        ],
        "CustomComponents": [
            {
                "ComponentName": "test_component_1",
                "ResourceList": [
                    "arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1234efgh5678i"
                ]
            },
            {
                "ComponentName": "test_component_2",
                "ResourceList": [
                    "arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1234efgh5678i",
                    "arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1234efgh5678i"
```

```
                ]
            }
        ],
        "LogPatternSets": [
            {
                "PatternSetName": "pattern_set_1",
                "LogPatterns": [
                    {
                        "PatternName": "deadlock_pattern",
                        "Pattern": ".*\\sDeadlocked\\sSchedulers(([^\\w].*)|($))",
                        "Rank": 1
                    }
                ]
            },
            {
                "PatternSetName": "pattern_set_2",
                "LogPatterns": [
                    {
                        "PatternName": "error_pattern",
                        "Pattern": ".*[\\s\\[]ERROR[\\s\\]].*",
                        "Rank": 1
                    },
                    {
                        "PatternName": "warning_pattern",
                        "Pattern": ".*[\\s\\[]WARN(ING)?[\\s\\]].*",
                        "Rank": 10
                    }
                ]
            }
        ]
    }
}
```

Template dalam format YAML

```
---
Type: AWS::ApplicationInsights::Application
Properties:
  ResourceGroupName: my_resource_group
  CWEMonitorEnabled: true
  OpsCenterEnabled: true
  OpsItemSNSTopicArn: arn:aws:sns:us-east-1:123456789012:my_topic
  AutoConfigurationEnabled: true
```

```
  Tags:
  - Key: key1
    Value: value1
  - Key: key2
    Value: ''
  CustomComponents:
  - ComponentName: test_component_1
    ResourceList:
    - arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1234efgh5678i
  - ComponentName: test_component_2
    ResourceList:
    - arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1234efgh5678i
    - arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1234efgh5678i
  LogPatternSets:
  - PatternSetName: pattern_set_1
    LogPatterns:
    - PatternName: deadlock_pattern
      Pattern: ".*\\sDeadlocked\\sSchedulers(([^\\w].*)|($))"
      Rank: 1
  - PatternSetName: pattern_set_2
    LogPatterns:
    - PatternName: error_pattern
      Pattern: ".*[\\s\\[]ERROR[\\s\\]].*"
      Rank: 1
    - PatternName: warning_pattern
      Pattern: ".*[\\s\\[]WARN(ING)?[\\s\\]].*"
      Rank: 10
```

Membuat sebuah aplikasi Wawasan Aplikasi dengan konfigurasi komponen mode **CUSTOM**

Template berikut akan melakukan tindakan-tindakan ini:

- Membuat sebuah aplikasi Wawasan Aplikasi. Untuk informasi selengkapnya, lihat CreateApplicationdi Referensi API Amazon CloudWatch Application Insights.
- Komponen `my_component` diatur `ComponentConfigurationMode` ke `CUSTOM`, yang akan menyebabkan komponen ini dikonfigurasi dengan konfigurasi yang ditentukan dalam `CustomComponentConfiguration`. Untuk informasi selengkapnya, lihat UpdateComponentConfigurationdi Referensi API Amazon CloudWatch Application Insights.

Template dalam format JSON

```
{
    "Type": "AWS::ApplicationInsights::Application",
    "Properties": {
        "ResourceGroupName": "my_resource_group,
        "ComponentMonitoringSettings": [
            {
                "ComponentARN": "my_component",
                "Tier": "SQL_SERVER",
                "ComponentConfigurationMode": "CUSTOM",
                "CustomComponentConfiguration": {
                    "ConfigurationDetails": {
                        "AlarmMetrics": [
                            {
                                "AlarmMetricName": "StatusCheckFailed"
                            },
                            ...
                        ],
                        "Logs": [
                            {
                                "LogGroupName": "my_log_group_1",
                                "LogPath": "C:\\LogFolder_1\\*",
                                "LogType": "DOT_NET_CORE",
                                "Encoding": "utf-8",
                                "PatternSet": "my_pattern_set_1"
                            },
                            ...
                        ],
                        "WindowsEvents": [
                            {
                                "LogGroupName": "my_windows_event_log_group_1",
                                "EventName": "Application",
                                "EventLevels": [
                                    "ERROR",
                                    "WARNING",
                                    ...
                                ],
                                "Encoding": "utf-8",
                                "PatternSet": "my_pattern_set_2"
                            },
                            ...
                        ],
                        "Alarms": [
                            {
```

```
                        "AlarmName": "my_alarm_name",
                        "Severity": "HIGH"
                    },
                    ...
                ]
            },
            "SubComponentTypeConfigurations": [
                {
                    "SubComponentType": "EC2_INSTANCE",
                    "SubComponentConfigurationDetails": {
                        "AlarmMetrics": [
                            {
                                "AlarmMetricName": "DiskReadOps"
                            },
                            ...
                        ],
                        "Logs": [
                            {
                                "LogGroupName": "my_log_group_2",
                                "LogPath": "C:\\LogFolder_2\\*",
                                "LogType": "IIS",
                                "Encoding": "utf-8",
                                "PatternSet": "my_pattern_set_3"
                            },
                            ...
                        ],
                        "processes" : [
                            {
                                "processName" : "my_process",
                                "alarmMetrics" : [
                            {
                                "alarmMetricName" : "procstat cpu_usage",
                                "monitor" : true
                            }, {
                                "alarmMetricName" : "procstat memory_rss",
                                "monitor" : true
                            }
                        ]
                    }
                ],
                        "WindowsEvents": [
                            {
                                "LogGroupName": "my_windows_event_log_group_2",
                                "EventName": "Application",
```

```
                                    "EventLevels": [
                                        "ERROR",
                                        "WARNING",
                                        ...
                                    ],
                                    "Encoding": "utf-8",
                                    "PatternSet": "my_pattern_set_4"
                                },
                                ...
                            ]
                        }
                    }
                ]
            }
        }
    ]
  }
}
```

Template dalam format YAML

```
---
Type: AWS::ApplicationInsights::Application
Properties:
  ResourceGroupName: my_resource_group
  ComponentMonitoringSettings:
  - ComponentARN: my_component
    Tier: SQL_SERVER
    ComponentConfigurationMode: CUSTOM
    CustomComponentConfiguration:
      ConfigurationDetails:
        AlarmMetrics:
        - AlarmMetricName: StatusCheckFailed
        ...
        Logs:
        - LogGroupName: my_log_group_1
          LogPath: C:\LogFolder_1\*
          LogType: DOT_NET_CORE
          Encoding: utf-8
          PatternSet: my_pattern_set_1
        ...
        WindowsEvents:
        - LogGroupName: my_windows_event_log_group_1
```

```
          EventName: Application
          EventLevels:
          - ERROR
          - WARNING
          ...
          Encoding: utf-8
          PatternSet: my_pattern_set_2
        ...
        Alarms:
        - AlarmName: my_alarm_name
          Severity: HIGH
        ...
      SubComponentTypeConfigurations:
      - SubComponentType: EC2_INSTANCE
        SubComponentConfigurationDetails:
          AlarmMetrics:
          - AlarmMetricName: DiskReadOps
          ...
          Logs:
          - LogGroupName: my_log_group_2
            LogPath: C:\LogFolder_2\*
            LogType: IIS
            Encoding: utf-8
            PatternSet: my_pattern_set_3
          ...
          Processes:
          - ProcessName: my_process
            AlarmMetrics:
            - AlarmMetricName: procstat cpu_usage
              ...
            ...
          WindowsEvents:
          - LogGroupName: my_windows_event_log_group_2
            EventName: Application
            EventLevels:
            - ERROR
            - WARNING
            ...
            Encoding: utf-8
            PatternSet: my_pattern_set_4
          ...
```

Membuat sebuah aplikasi Wawasan Aplikasi dengan konfigurasi komponen mode `DEFAULT`

Template berikut akan melakukan tindakan-tindakan ini:

- Membuat sebuah aplikasi Wawasan Aplikasi. Untuk informasi selengkapnya, lihat CreateApplicationdi Referensi API Amazon CloudWatch Application Insights.
- Komponen `my_component` mengatur `ComponentConfigurationMode` dengan `DEFAULT` dan `Tier` dengan `SQL_SERVER`, yang akan menyebabkan komponen ini dikonfigurasi dengan pengaturan konfigurasi yang direkomendasikan oleh Wawasan Aplikasi untuk tingkat `SQL_Server`. Untuk informasi selengkapnya, lihat DescribeComponentConfigurationdan UpdateComponentConfigurationdi Referensi API Amazon CloudWatch Application Insights.

Template dalam format JSON

```
{
    "Type": "AWS::ApplicationInsights::Application",
    "Properties": {
        "ResourceGroupName": "my_resource_group",
        "ComponentMonitoringSettings": [
            {
                "ComponentARN": "my_component",
                "Tier": "SQL_SERVER",
                "ComponentConfigurationMode": "DEFAULT"
            }
        ]
    }
}
```

Template dalam format YAML

```
---
Type: AWS::ApplicationInsights::Application
Properties:
  ResourceGroupName: my_resource_group
  ComponentMonitoringSettings:
  - ComponentARN: my_component
    Tier: SQL_SERVER
    ComponentConfigurationMode: DEFAULT
```

Membuat sebuah aplikasi Wawasan Aplikasi dengan konfigurasi komponen mode `DEFAULT_WITH_OVERWRITE`

Template berikut akan melakukan tindakan-tindakan ini:

- Membuat sebuah aplikasi Wawasan Aplikasi. Untuk informasi selengkapnya, lihat CreateApplicationdi Referensi API Amazon CloudWatch Application Insights.
- Komponen `my_component` mengatur `ComponentConfigurationMode` dengan `DEFAULT_WITH_OVERWRITE` dan `tier` dengan `DOT_NET_CORE`, yang akan menyebabkan komponen ini dikonfigurasi dengan pengaturan konfigurasi yang direkomendasikan oleh Wawasan Aplikasi untuk tingkat `DOT_NET_CORE`. Pengaturan konfigurasi yang ditimpa ditentukan di `DefaultOverwriteComponentConfiguration`:
  - Pada tingkat komponen, pengaturan `AlarmMetrics` ditimpa.
  - Pada tingkat sub-komponen, untuk sub-komponen jenis `EC2_Instance`, pengaturan `Logs` ditimpa.

  Untuk informasi selengkapnya, lihat UpdateComponentConfigurationdi Referensi API Amazon CloudWatch Application Insights.

Template dalam format JSON

```
{
    "Type": "AWS::ApplicationInsights::Application",
    "Properties": {
        "ResourceGroupName": "my_resource_group",
        "ComponentMonitoringSettings": [
            {
                "ComponentName": "my_component",
                "Tier": "DOT_NET_CORE",
                "ComponentConfigurationMode": "DEFAULT_WITH_OVERWRITE",
                "DefaultOverwriteComponentConfiguration": {
                    "ConfigurationDetails": {
                        "AlarmMetrics": [
                            {
                                "AlarmMetricName": "StatusCheckFailed"
                            }
                        ]
                    },
                    "SubComponentTypeConfigurations": [
                        {
```

```
                            "SubComponentType": "EC2_INSTANCE",
                            "SubComponentConfigurationDetails": {
                                "Logs": [
                                    {
                                        "LogGroupName": "my_log_group",
                                        "LogPath": "C:\\LogFolder\\*",
                                        "LogType": "IIS",
                                        "Encoding": "utf-8",
                                        "PatternSet": "my_pattern_set"
                                    }
                                ]
                            }
                        }
                    ]
                }
            }
        ]
    }
}
```

Template dalam format YAML

```
---
Type: AWS::ApplicationInsights::Application
Properties:
  ResourceGroupName: my_resource_group
  ComponentMonitoringSettings:
  - ComponentName: my_component
    Tier: DOT_NET_CORE
    ComponentConfigurationMode: DEFAULT_WITH_OVERWRITE
    DefaultOverwriteComponentConfiguration:
      ConfigurationDetails:
        AlarmMetrics:
        - AlarmMetricName: StatusCheckFailed
      SubComponentTypeConfigurations:
      - SubComponentType: EC2_INSTANCE
        SubComponentConfigurationDetails:
          Logs:
          - LogGroupName: my_log_group
            LogPath: C:\LogFolder\*
            LogType: IIS
            Encoding: utf-8
            PatternSet: my_pattern_set
```

Tutorial: Cara menyiapkan pemantauan untuk SAP ASE

Tutorial ini akan menjelaskan kepada Anda cara mengonfigurasi Wawasan Aplikasi CloudWatch untuk menyiapkan pemantauan untuk basis data SAP ASE. Anda dapat menggunakan dasbor otomatis Wawasan Aplikasi CloudWatch untuk menampilkan detail masalah, mempercepat pemecahan masalah, dan memfasilitasi mean time to resolution (MTTR) untuk basis data SAP ASE Anda.

Topik-topik Wawasan Aplikasi untuk SAP ASE

- Lingkungan yang didukung
- Sistem operasi yang didukung
- Fitur-fitur
- Prasyarat
- Menyiapkan pemantauan pada basis data SAP ASE Anda
- Mengelola pemantauan basis data SAP ASE Anda
- Mengonfigurasi ambang batas alarm
- Menampilkan dan memecahkan masalah-masalah SAP ASE yang terdeteksi oleh Wawasan Aplikasi
- Penyelesaian Masalah Wawasan Aplikasi untuk SAP ASE

Lingkungan yang didukung

Wawasan Aplikasi CloudWatch mendukung deployment sumber daya AWS untuk sistem dan pola berikut ini. Anda menyediakan dan melakukan instalasi perangkat lunak basis data SAP ASE dan perangkat lunak aplikasi SAP yang didukung.

- Satu atau beberapa basis data SAP ASE pada satu instans Amazon EC2 – SAP ASE dalam arsitektur simpul–tunggal dan dinaikkan skalanya.
- Pengaturan ketersediaan tinggi basis data SAP ASE lintas AZ – SAP ASE dengan ketersediaan tinggi yang dikonfigurasi di dua Zona Ketersediaan dengan menggunakan pengklasteran SUSE/RHEL.

> **Note**
>
> Wawasan Aplikasi CloudWatch hanya mendukung lingkungan ASE HA ID sistem SAP (SID) tunggal. Jika ada beberapa ASE HA SID yang dilampirkan, maka pemantauan akan diatur hanya untuk SID pertama yang terdeteksi.

Sistem operasi yang didukung

Wawasan Aplikasi CloudWatch untuk SAP ASE mendukung arsitektur x86-64 pada sistem operasi berikut ini:

- SUSE Linux 12 SP4
- SUSE Linux 12 SP5
- SUSE Linux 15
- SUSE Linux 15 SP1
- SUSE Linux 15 SP2
- SUSE Linux 15 SP3
- SUSE Linux 15 SP4
- SuSE Linux 15 SP1 Untuk SAP
- SuSE Linux 15 SP2 Untuk SAP
- SuSE Linux 15 SP3 Untuk SAP
- SuSE Linux 15 SP4 Untuk SAP
- SuSE Linux 12 SP4 Untuk SAP
- SuSE Linux 12 SP5 Untuk SAP
- RedHat Linux 7.6
- RedHat Linux 7.7
- RedHat Linux 7.9
- RedHat Linux 8.1
- RedHat Linux 8.4
- RedHat Linux 8.6

Fitur-fitur

Wawasan Aplikasi CloudWatch untuk SAP ASE menyediakan fitur-fitur berikut ini:

- Deteksi beban kerja SAP ASE secara otomatis
- Pembuatan alarm SAP ASE secara otomatis berdasarkan ambang batas statis
- Pembuatan alarm SAP ASE berdasarkan pada deteksi anomali
- Pengenalan pola log SAP ASE secara otomatis
- Dasbor kesehatan untuk SAP ASE
- Dasbor masalah untuk SAP ASE

Prasyarat

Anda harus memenuhi prasyarat berikut untuk mengonfigurasi basis data SAP ASE dengan Wawasan Aplikasi CloudWatch:

- Parameter konfigurasi SAP ASE – Parameter konfigurasi berikut harus diaktifkan pada ASE DB Anda: `"enable monitoring"`, `"sql text pipe max messages"`, `"sql text pipe active"`. Hal ini akan memungkinkan Wawasan Aplikasi CloudWatch untuk menyediakan kemampuan pemantauan penuh untuk DB Anda. Jika pengaturan ini tidak diaktifkan pada basis data ASE Anda, maka Wawasan Aplikasi akan secara otomatis mengaktifkannya untuk mengumpulkan metrik-metrik yang diperlukan untuk memungkinkan pelaksanaan pemantauan.
- Pengguna basis data SAP ASE – Pengguna basis data yang disediakan selama orientasi Wawasan Aplikasi harus memiliki izin untuk mengakses hal–hal berikut:
  - Tabel sistem dalam basis data master dan basis data pengguna (penyewa)
  - Memantau tabel
- SAPHostCtrl – Menginstal dan mengatur SAPHostCtrl pada instans Amazon EC2 Anda.
- Agen Amazon CloudWatch – Memastikan Anda tidak menjalankan agen CloudWatch yang sudah ada sebelumnya di instans Amazon EC2 Anda. Jika Anda telah melakukan instalasi agen CloudWatch, maka Anda harus memastikan untuk menghapus konfigurasi sumber daya yang Anda gunakan di Wawasan Aplikasi CloudWatch dari file konfigurasi agen CloudWatch yang ada untuk menghindari konflik gabungan. Untuk informasi selengkapnya, lihat Buat atau edit file konfigurasi CloudWatch agen secara manual.

- Pengaktifan AWS Systems Manager – Menginstal Agen SSM di instans Anda, dan mengaktifkan instans yang diaktifkan untuk SSM. Untuk informasi tentang cara melakukan instalasi agen SSM, silakan lihat Bekerja dengan Agen SSM di Panduan Pengguna AWS Systems Manager.
- Peran instans Amazon EC2 Anda harus mengaitkan peran instans Amazon EC2 berikut untuk mengonfigurasi basis data Anda.
  - Anda harus melampirkan peran `AmazonSSMManagedInstanceCore` untuk mengaktifkan Systems Manager. Untuk informasi selengkapnya, silakan lihat contoh kebijakan berbasis identitas AWS Systems Manager.
  - Anda harus mengaitkan `CloudWatchAgentServerPolicy` untuk mengaktifkan metrik dan log instans agar dapat dipancarkan melalui CloudWatch. Untuk informasi selengkapnya, silakan lihat Membuat peran IAM dan pengguna untuk digunakan dengan agen Amazon CloudWatch.
  - Anda harus mengaitkan kebijakan selaras IAM berikut ini terhadap peran instans Amazon EC2 agar Anda bisa membaca kata sandi yang disimpan di AWS Secrets Manager. Untuk informasi selengkapnya tentang kebijakan-kebijakan inline, silakan lihat Kebijakan Inline di Panduan Pengguna AWS Identity and Access Management.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:ApplicationInsights-*"
        }
    ]
}
```

- AWS Resource Groups – Anda harus membuat sebuah grup sumber daya yang mencakup semua sumber AWS daya terkait yang digunakan oleh tumpukan aplikasi Anda untuk memasukkan aplikasi Anda ke Wawasan Aplikasi CloudWatch. Ini termasuk instans Amazon EC2 dan volume Amazon EBS yang menjalankan basis data SAP ASE Anda. Jika ada beberapa basis data per akun, kami sarankan Anda untuk membuat satu grup sumber daya yang mencakup sumber daya AWS untuk setiap sistem basis data SAP ASE.
- Izin IAM – Untuk pengguna non–admin:

- Anda harus membuat sebuah kebijakan AWS Identity and Access Management (IAM) yang memungkinkan Wawasan Aplikasi membuat sebuah peran tertaut layanan, dan melampirkannya ke identitas pengguna Anda. Untuk mengetahui langkah-langkah untuk melampirkan kebijakan tersebut, silakan lihat Kebijakan IAM.
- Pengguna harus memiliki izin untuk membuat rahasia di AWS Secrets Manager untuk menyimpan kredensial pengguna basis data. Untuk informasi selengkapnya, silakan lihat Contoh: Izin untuk membuat rahasia.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:CreateSecret"
      ],
      "Resource": "arn:aws:secretsmanager:*:*:secret:ApplicationInsights-*"
    }
  ]
}
```

- Peran tertaut layanan – Wawasan Aplikasi menggunakan peran-peran tertaut layanan AWS Identity and Access Management (IAM). Sebuah peran tertaut layanan dibuat untuk Anda saat Anda membuat aplikasi Wawasan Aplikasi pertama Anda di konsol Wawasan Aplikasi. Untuk informasi selengkapnya, lihat Menggunakan peran tertaut layanan untuk Wawasan Aplikasi CloudWatch.

Menyiapkan pemantauan pada basis data SAP ASE Anda

Gunakan langkah-langkah berikut untuk menyiapkan pemantauan untuk basis data SAP ASE

1. Buka Konsol CloudWatch.
2. Dari panel navigasi yang ada di bagian kiri, pada Pemantauan infrastruktur, pilih Wawasan Aplikasi.
3. Halaman Wawasan Aplikasi akan menampilkan daftar aplikasi yang dipantau dengan Wawasan Aplikasi, dan status pemantauan untuk masing-masing aplikasi tersebut. Di bagian pojok kanan atas, pilih Tambahkan aplikasi.
4. Pada halaman Tentukan detail aplikasi, dari daftar geser-turun pada Grup sumber daya, pilih grup sumber daya AWS yang berisi sumber daya basis data SAP ASE Anda. Jika Anda belum

membuat sebuah grup sumber daya untuk aplikasi Anda, maka Anda dapat membuatnya dengan memilih Buat grup sumber daya baru pada daftar geser-turun Grup sumber daya. Untuk informasi selengkapnya tentang cara membuat grup sumber daya, silakan lihat Panduan Pengguna AWS Resource Groups.

5. Pada Monitor CloudWatch Events, pilih kotak centang untuk mengintegrasikan pemantauan Wawasan Aplikasi dengan CloudWatch Events untuk mendapatkan wawasan dari Amazon EBS, Amazon EC2, AWS CodeDeploy, Amazon ECS, API dan notifikasi AWS Health, Amazon RDS, Amazon S3, dan AWS Step Functions.
6. Pada Integrasikan dengan AWS Systems Manager OpsCenter, pilih kotak centang yang ada di samping Buat AWS Systems Manager OpsCenter Opsitems untuk tindakan perbaikan untuk menampilkan dan mendapatkan notifikasi saat ada masalah terdeteksi untuk aplikasi yang telah Anda pilih. Untuk melacak operasi yang dilakukan untuk menyelesaikan item-item pekerjaan operasional, yang bernama OppItems, yang terkait dengan sumber daya AWS Anda, silakan berikan ARN topik SNS.
7. Anda dapat memasukkan tag secara opsional untuk membantu Anda melakukan identifikasi dan pengaturan sumber daya. Wawasan Aplikasi CloudWatch mendukung grup sumber daya berbasis tag dan AWS CloudFormation berbasis tumpukan, kecuali grup Application Auto Scaling. Untuk informasi selengkapnya, silakan lihat Editor Tag di Panduan Pengguna AWS Resource Groups dan Tag.
8. Pilih Berikutnya untuk melanjutkan melakukan pengaturan pemantauan.
9. Pada halaman Tinjau komponen terdeteksi, komponen-komponen yang dipantau dan beban kerjanya yang terdeteksi secara otomatis oleh Wawasan Aplikasi akan dicantumkan.

> **Note**
>
> Komponen-komponen yang berisi beban kerja SAP ASE High Availability yang terdeteksi hanya mendukung satu beban kerja pada suatu komponen. Komponen-komponen yang berisi beban kerja simpul tunggal SAP ASE yang terdeteksi mendukung beberapa beban kerja, tetapi Anda tidak akan dapat menambah atau menghapus beban kerja. Semua beban kerja yang terdeteksi secara otomatis akan dipantau.

10. Pilih Berikutnya.
11. Pada halaman Tentukan detail komponen, masukkan nama pengguna dan kata sandi basis data SAP ASE Anda.
12. Tinjau konfigurasi pemantauan aplikasi Anda, dan kemudian pilih Kirim.

13. Halaman detail aplikasi kemudian terbuka, di mana Anda dapat melihat Ringkasan aplikasi, daftar Komponen dan beban kerja yang dipantau, serta Komponen dan beban kerja yang tidak dipantau. Jika Anda memilih tombol radio yang ada di samping komponen atau beban kerja, maka Anda juga dapat melihat Riwayat konfigurasi, Pola Log, dan Tag apa pun yang telah Anda buat sebelumnya. Ketika Anda mengirimkan konfigurasi Anda, akun Anda akan menyebarkan semua metrik dan alarm untuk sistem SAP ASE Anda, hal ini dapat memakan waktu hingga 2 jam.

Mengelola pemantauan basis data SAP ASE Anda

Anda dapat mengelola kredensial pengguna, metrik, dan jalur log untuk basis data SAP ASE Anda dengan melakukan langkah-langkah berikut:

1. Buka Konsol CloudWatch.
2. Dari panel navigasi yang ada di bagian kiri, pada Pemantauan infrastruktur, pilih Wawasan Aplikasi.
3. Halaman Wawasan Aplikasi akan menampilkan daftar aplikasi yang dipantau dengan Wawasan Aplikasi, dan status pemantauan untuk masing-masing aplikasi tersebut.
4. Pada Komponen yang dipantau, pilih tombol radio yang ada di samping nama komponen. Kemudian, silakan pilih Kelola pemantauan.
5. Pada log grup instans EC2, Anda dapat memperbarui jalur log yang sudah ada, kumpulan pola log, dan nama grup log. Selain itu, Anda juga dapat menambahkan hingga tiga Log aplikasi tambahan.
6. Pada Metrik, Anda dapat memilih metrik SAP ASE sesuai dengan kebutuhan Anda. Nama-nama metrik SAP ASE diawali dengan `asedb`. Anda dapat menambahkan hingga 60 metrik untuk masing-masing komponen.
7. Pada Konfigurasi ASE, silakan masukkan nama pengguna dan kata sandi untuk basis data SAP ASE. Ini adalah nama pengguna dan kata sandi yang digunakan oleh agen Amazon CloudWatch untuk terhubung ke basis data SAP ASE.
8. Pada Alarm kustom, Anda dapat menambahkan alarm-alarm tambahan yang akan dipantau oleh Wawasan Aplikasi CloudWatch.
9. Tinjau konfigurasi pemantauan aplikasi Anda dan kemudian pilih Kirim. Ketika Anda mengirimkan konfigurasi Anda, akun Anda akan memperbarui semua metrik dan alarm untuk sistem SAP HANA Anda, hal ini dapat memakan waktu hingga 2 jam.

Mengonfigurasi ambang batas alarm

Wawasan Aplikasi CloudWatch secara otomatis akan membuat metrik Amazon CloudWatch untuk alarm yang akan diawasi, bersama dengan ambang batas untuk metrik tersebut. Alarm tersebut statusnya akan beralih menjadi status `ALARM` saat metrik melampaui ambang batas untuk jumlah periode evaluasi yang ditentukan. Perlu diperhatikan bahwa pengaturan ini tidak dipertahankan oleh Wawasan Aplikasi.

Untuk mengedit sebuah alarm untuk satu metrik, Anda bisa melakukan langkah-langkah berikut:

1. Buka Konsol CloudWatch.
2. Pada panel navigasi yang ada di bagian kiri, pilih Alarm>Semua alarm.
3. Pilih tombol radio yang ada di samping alarm yang sudah dibuat secara otomatis oleh Wawasan Aplikasi CloudWatch. Kemudian pilih Tindakan, dan pilih Edit dari menu geser-turun.
4. Edit parameter berikut ini pada Metrik.
   a. Pada Statistik, pilih salah satu statistik atau persentil yang telah ditentukan sebelumnya, atau tentukan persentil yang Anda kehendaki. Sebagai contoh, `p95.45`.
   b. Pada Periode, silakan pilih periode evaluasi untuk alarm. Saat Anda melakukan evaluasi alarm, masing-masing periode digabungkan menjadi satu titik data.
5. Edit parameter-parameter berikut ini pada Kondisi.
   a. Pilih apakah metriknya harus lebih besar dari, kurang dari, atau sama dengan ambang batas.
   b. Tentukan nilai ambang batasnya.
6. Pada Konfigurasi tambahan, silakan edit parameter-parameter berikut.
   a. Pada Titik data ke alarm, silakan tentukan jumlah titik data, atau periode evaluasi, yang harus berada dalam status `ALARM` untuk memulai alarm. Ketika dua nilai tersebut cocok, sebuah alarm akan tercipta dan berada dalam status `ALARM` jika jumlah periode berturut-turut yang ditentukan terlampaui. Untuk membuat `m` dari alarm `n`, tentukan nilai yang lebih rendah untuk titik data pertama, bukan untuk titik data yang kedua. Untuk informasi selengkapnya tentang cara melakukan evaluasi alarm, silakan lihat Mengevaluasi alarm.
   b. Pada Perlakuan data yang hilang, silakan pilih perilaku alarm saat ada beberapa titik data yang tidak ditemukan. Untuk informasi selengkapnya tentang perlakuan data yang hilang, silakan lihat Mengonfigurasi cara kerja alarm CloudWatch dalam menangani data hilang.

c. Jika alarm menggunakan sebuah persentil sebagai statistik yang dipantau, maka kotak Persentil dengan sampel rendah akan ditampilkan. Pilih apakah Anda akan mengevaluasi atau mengabaikan kasus yang memiliki tingkat sampel yang kecil. Jika Anda memilih abaikan (status alarm tidak berubah), maka status alarm saat ini akan tetap dipertahankan ketika ukuran sampel terlalu kecil. Untuk informasi selengkapnya tentang persentil dengan sampel yang rendah, silakan lihat CloudWatch Alarm berbasis persentil dan sampel data rendah.

7. Pilih Berikutnya.
8. Pada Notifikasi, silakan pilih topik SNS yang akan diberitahukan sebagai notifikasi ketika alarm berada dalam status `ALARM`, status `OK`, atau status `INSUFFICIENT_DATA`.
9. Pilih Perbarui alarm.

Menampilkan dan memecahkan masalah-masalah SAP ASE yang terdeteksi oleh Wawasan Aplikasi

Bagian ini akan membantu Anda dalam mengatasi persoalan pemecahan masalah umum yang terjadi saat Anda mengonfigurasi pemantauan untuk SAP ASE pada Wawasan Aplikasi.

Kesalahan Server Cadangan SAP ASE

Anda dapat mengidentifikasi pesan kesalahan dengan memeriksa dasbor yang dibuat secara dinamis. Dasbor tersebut akan menunjukkan kepada Anda pesan kesalahan yang dilaporkan di Server Cadangan SAP ASE. Untuk detail selengkapnya tentang log Server Cadangan SAP ASE, silakan lihat Pencatatan Kesalahan Server Cadangan Dokumentasi SAP.

Transaksi SAP ASE yang berjalan lama

Mengidentifikasi transaksi yang berjalan lama dan mengonfirmasikan apakah transaksi itu dapat dihentikan atau apakah waktu yang berjalan lama itu adalah hal yang disengaja. Untuk detail selengkapnya, silakan lihat 2180410 — Bagaimana cara menampilkan catatan log transaksi untuk transaksi-transaksi yang berjalan lama? — SAP ASE.

Koneksi Pengguna SAP ASE

Meninjau apakah basis data SAP ASE Anda memiliki ukuran yang sesuai untuk beban kerja yang ingin Anda jalankan di basis data tersebut. Untuk detail selengkapnya, silakan lihat Mengonfigurasi Koneksi Pengguna di dokumentasi SAP.

Ruang disk SAP ASE

Anda dapat mengidentifikasi lapisan basis data yang menyebabkan terjadinya masalah dengan memeriksa dasbor yang dibuat secara dinamis. Dasbor tersebut akan menunjukkan metrik-metrik terkait dan potongan file log. Anda harus harus memahami penyebab terjadinya pertumbuhan disk, dan bila hal itu terjadi, Anda harus menambah ukuran disk fisik, ruang disk yang dialokasikan, atau keduanya. Untuk detail selengkapnya, silakan lihat Mengubah ukuran disk Dokumentasi SAP dalam dokumentasi SAP.

Penyelesaian Masalah Wawasan Aplikasi untuk SAP ASE

Bagian ini akan menjelaskan kepada Anda tentang langkah-langkah yang akan membantu Anda dalam mengatasi kesalahan-kesalahan umum yang ditampilkan dalam dasbor Wawasan Aplikasi.

Kesalahan	Kesalahan yang ditampilkan	Penyebab galat	Resolusi
Tidak dapat menambahkan metrik monitor lebih dari 60 metrik monitor.	`Component cannot have more than 60 monitored metric`	Batas metrik saat ini adalah 60 metrik yang dipantau untuk setiap komponen.	Menghapus metrik yang tidak perlu untuk mematuhi batas metrik yang ditentukan.
Tidak ada metrik atau alarm SAP yang muncul setelah proses onboarding	Perintah `run` pada `AWS-ConfigureAWSPackage` gagal di AWS Systems Manager. Outputnya menunjukkan ada kesalahan : `CT-LIBRARY error:ct_connect(): protocol specific layer: external error: The attempt to connect to the server failed`	Nama pengguna dan kata sandi mungkin salah.	Verifikasi bahwa nama pengguna dan kata sandi sudah benar, kemudian jalankan kembali proses onboarding.

Tutorial: Cara menyiapkan pemantauan untuk SAP HANA

Tutorial ini menunjukkan cara mengkonfigurasi CloudWatch Application Insights untuk mengatur pemantauan untuk database SAP HANA Anda. Anda dapat menggunakan dasbor otomatis CloudWatch Application Insights untuk memvisualisasikan detail masalah, mempercepat pemecahan masalah, dan memfasilitasi mean time to resolution (MTTR) untuk database SAP HANA Anda.

Wawasan Aplikasi untuk topik-topik SAP HANA

- Lingkungan yang didukung
- Sistem operasi yang didukung
- Fitur
- Prasyarat
- Menyiapkan basis data SAP HANA Anda untuk pemantauan
- Mengelola pemantauan basis data SAP HANA Anda
- Mengonfigurasi ambang batas alarm
- Melihat dan memecahkan masalah SAP HANA yang terdeteksi oleh Application Insights CloudWatch
- Deteksi anomali untuk SAP HANA
- Penyelesaian Masalah Wawasan Aplikasi untuk SAP HANA

Lingkungan yang didukung

CloudWatch Application Insights mendukung penyebaran sumber AWS daya untuk sistem dan pola berikut. Anda menyediakan dan melakukan instalasi perangkat lunak basis data SAP HANA dan perangkat lunak aplikasi SAP yang didukung.

- Basis data SAP HANA pada satu instans Amazon EC2 — SAP HANA dalam arsitektur simpul-tunggal dan telah dinaikkan skalanya dengan memori hingga 24TB.
- Basis data SAP HANA pada beberapa instans Amazon EC2 — SAP HANA dalam arsitektur multi-simpul dan diskalakan ke luar.
- Pengaturan ketersediaan tinggi basis data SAP ASE lintas AZ — SAP HANA dengan ketersediaan tinggi yang dikonfigurasi di dua Zona Ketersediaan dengan menggunakan pengklasteran SUSE/RHEL.

> **Note**
>
> CloudWatch Application Insights hanya mendukung lingkungan SID HANA tunggal. Jika ada beberapa HANA SID yang dilampirkan, maka pemantauan akan diatur hanya untuk SID pertama yang terdeteksi.

Sistem operasi yang didukung

CloudWatch Application Insights untuk SAP HANA mendukung arsitektur x86-64 pada sistem operasi berikut:

- SuSE Linux 12 SP4 Untuk SAP
- SuSE Linux 12 SP5 Untuk SAP
- SUSE Linux 15
- SUSE Linux 15 SP1
- SUSE Linux 15 SP2
- SuSE Linux 15 Untuk SAP
- SuSE Linux 15 SP1 Untuk SAP
- SuSE Linux 15 SP2 Untuk SAP
- SuSE Linux 15 SP3 Untuk SAP
- SuSE Linux 15 SP4 Untuk SAP
- SuSE Linux 15 SP5 Untuk SAP
- RedHat Linux 8.6 Untuk SAP Dengan Ketersediaan Tinggi dan Layanan Pembaruan
- RedHat Linux 8.5 Untuk SAP Dengan Ketersediaan Tinggi dan Layanan Pembaruan
- RedHat Linux 8.4 Untuk SAP Dengan Ketersediaan Tinggi dan Layanan Pembaruan
- RedHat Linux 8.3 Untuk SAP Dengan Ketersediaan Tinggi dan Layanan Pembaruan
- RedHat Linux 8.2 Untuk SAP Dengan Ketersediaan Tinggi dan Layanan Pembaruan
- RedHat Linux 8.1 Untuk SAP Dengan Ketersediaan Tinggi dan Layanan Pembaruan
- RedHat Linux 7.9 Untuk SAP Dengan Ketersediaan Tinggi dan Layanan Pembaruan

Fitur

CloudWatch Application Insights untuk SAP HANA menyediakan fitur-fitur berikut:

- Deteksi beban kerja SAP HANA secara otomatis
- Pembuatan alarm SAP HANA secara otomatis berdasarkan ambang batas statis
- Pembuatan alarm SAP HANA secara otomatis berdasarkan pada deteksi anomali
- Pengenalan pola log SAP HANA secara otomatis
- Dasbor Kesehatan untuk SAP HANA
- Dasbor masalah untuk SAP HANA

Prasyarat

Anda harus melakukan prasyarat berikut untuk mengkonfigurasi database SAP HANA dengan Application Insights: CloudWatch

- SAP HANA — Menginstal basis data SAP HANA 2.0 SPS05 yang berjalan dan dapat dijangkau pada sebuah instans Amazon EC2.
- Pengguna basis data SAP HANA — Seorang pengguna basis data dengan peran-peran pemantauan harus dibuat dalam basis data SYSTEM dan semua penyewa.

  Contoh

  Perintah-perintah SQL berikut bisa digunakan untuk membuat pengguna dengan peran-peran pemantauannya.

  ```
  su - <sid>adm
  hdbsql -u SYSTEM -p <SYSTEMDB password> -d SYSTEMDB
  CREATE USER CW_HANADB_EXPORTER_USER PASSWORD <Monitoring user password> NO
   FORCE_FIRST_PASSWORD_CHANGE;
  CREATE ROLE CW_HANADB_EXPORTER_ROLE;
  GRANT MONITORING TO CW_HANADB_EXPORTER_ROLE;
  GRANT CW_HANADB_EXPORTER_ROLE TO CW_HANADB_EXPORTER_USER;
  ```

- Python 3.6 — Melakukan instalasi Python 3.6 atau versi yang lebih baru pada sistem operasi Anda. Menggunakan Python rilis terbaru. Jika Python tidak terdeteksi pada sistem operasi Anda, maka Python 3.8 akan diinstal pada sistem operasi tersebut.

  Lihat installation examples.

- Pip3 — Menginstal program installer, pip3, pada sistem operasi Anda. Jika pip3 tidak terdeteksi pada sistem operasi Anda, maka pip3 akan diinstal pada sistem tersebut.

- CloudWatch Agen Amazon — Pastikan Anda tidak menjalankan agen yang sudah ada sebelumnya di CloudWatch instans Amazon EC2 Anda. Jika Anda telah menginstal CloudWatch agen, pastikan untuk menghapus konfigurasi sumber daya yang Anda gunakan di CloudWatch Application Insights dari file konfigurasi CloudWatch agen yang ada untuk menghindari konflik gabungan. Untuk informasi selengkapnya, lihat Buat atau edit file konfigurasi CloudWatch agen secara manual.
- Pengaktifan AWS Systems Manager — Menginstal Agen SSM pada instans Anda, dan mengaktifkan instans yang harus diaktifkan untuk SSM. Untuk informasi tentang cara melakukan instalasi agen SSM, silakan lihat Bekerja dengan Agen SSM di Panduan Pengguna AWS Systems Manager.
- Peran instans Amazon EC2 — Anda harus mengaitkan peran instans Amazon EC2 berikut untuk mengonfigurasi basis data Anda.
  - Anda harus melampirkan peran `AmazonSSMManagedInstanceCore` untuk mengaktifkan Systems Manager. Untuk informasi selengkapnya, silakan lihat contoh kebijakan berbasis identitas AWS Systems Manager.
  - Anda harus `CloudWatchAgentServerPolicy` melampirkan metrik dan log instance untuk mengaktifkan untuk dipancarkan. CloudWatch Untuk informasi selengkapnya, lihat Membuat peran IAM dan pengguna untuk digunakan dengan CloudWatch agen.
  - Anda harus mengaitkan kebijakan selaras IAM berikut ini terhadap peran instans Amazon EC2 agar Anda bisa membaca kata sandi yang disimpan di AWS Secrets Manager. Untuk informasi selengkapnya tentang kebijakan-kebijakan inline, silakan lihat Kebijakan Inline di Panduan Pengguna AWS Identity and Access Management.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:ApplicationInsights-*"
        }
    ]
}
```

- AWSgrup sumber daya — Anda harus membuat grup sumber daya yang mencakup semua AWS sumber daya terkait yang digunakan oleh tumpukan aplikasi Anda untuk onboard aplikasi Anda ke

Wawasan CloudWatch Aplikasi. Ini termasuk instans Amazon EC2 dan volume Amazon EBS yang menjalankan basis data SAP HANA Anda. Jika ada beberapa basis data per akun, kami sarankan Anda membuat satu grup sumber daya yang mencakup sumber daya AWS untuk setiap sistem basis data SAP HANA.

- Izin IAM — Untuk pengguna non-admin:
  - Anda harus membuat sebuah kebijakan AWS Identity and Access Management (IAM) yang memungkinkan Wawasan Aplikasi membuat sebuah peran terkait layanan, dan melampirkannya ke identitas pengguna Anda. Untuk mengetahui langkah-langkah untuk melampirkan kebijakan tersebut, silakan lihat Kebijakan IAM.
  - Pengguna harus memiliki izin untuk membuat rahasia di AWS Secrets Manager untuk menyimpan kredensial pengguna basis data. Untuk informasi selengkapnya, silakan lihat Contoh: Izin untuk membuat rahasia.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:CreateSecret"
      ],
      "Resource": "arn:aws:secretsmanager:*:*:secret:ApplicationInsights-*"
    }
  ]
}
```

- Peran terkait layanan — Wawasan Aplikasi menggunakan peran terkait layanan AWS Identity and Access Management (IAM). Sebuah peran terkait layanan dibuat untuk Anda saat Anda membuat aplikasi Wawasan Aplikasi pertama Anda di konsol Wawasan Aplikasi. Untuk informasi selengkapnya, lihat Menggunakan peran tertaut layanan untuk Wawasan Aplikasi CloudWatch.

Menyiapkan basis data SAP HANA Anda untuk pemantauan

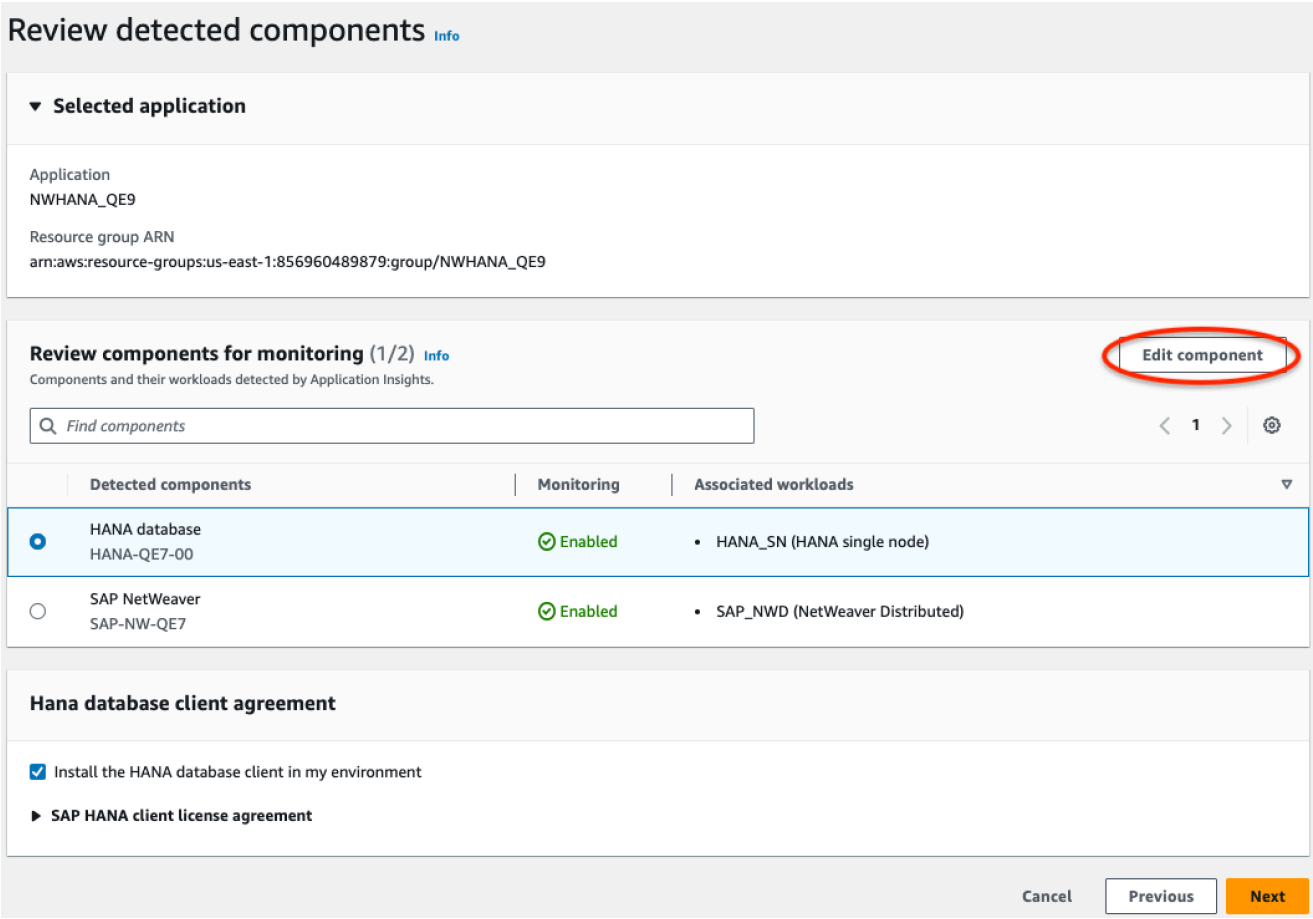
Gunakan langkah-langkah berikut untuk menyiapkan pemantauan untuk basis data SAP HANA Anda

1. Buka konsol CloudWatch.
2. Dari panel navigasi yang ada di bagian kiri, pada Pemantauan infrastruktur, pilih Wawasan Aplikasi.

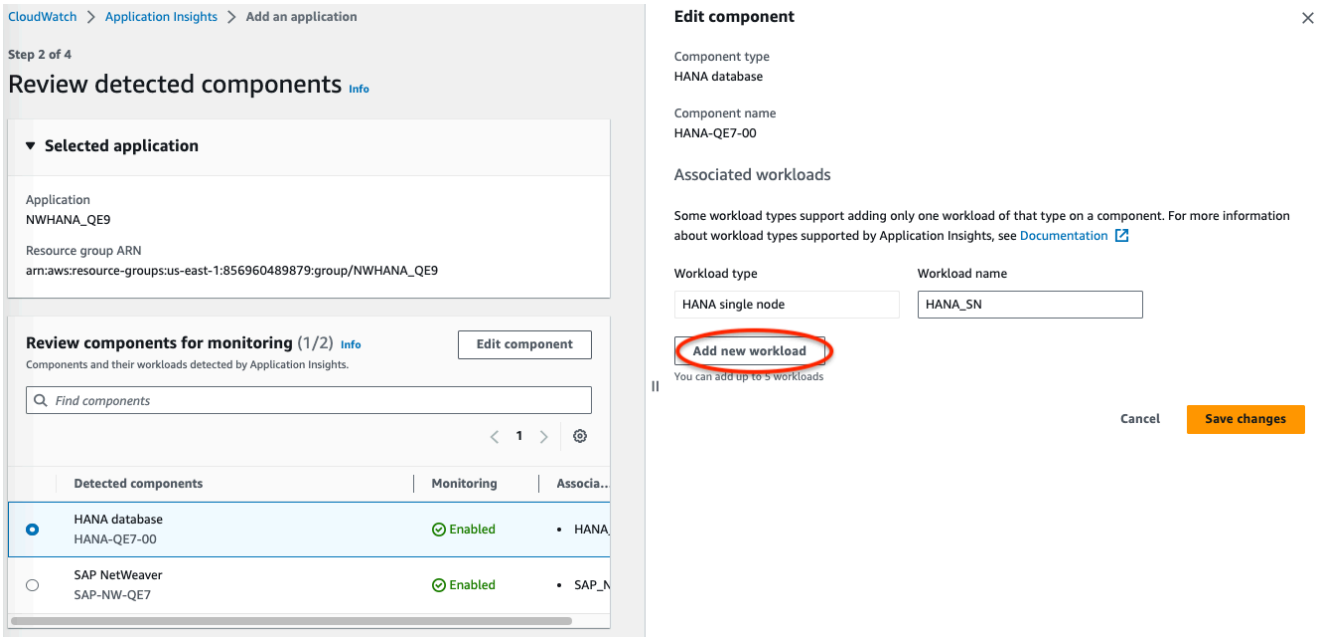
3. Halaman Wawasan Aplikasi akan menampilkan daftar aplikasi yang dipantau dengan Wawasan Aplikasi, dan status pemantauan untuk masing-masing aplikasi tersebut. Di bagian pojok kanan atas, pilih Tambahkan aplikasi.
4. Pada halaman Tentukan detail aplikasi, dari daftar geser-turun pada Grup sumber daya, pilih grup sumber daya AWS yang berisi sumber daya basis data SAP HANA Anda. Jika Anda belum membuat sebuah grup sumber daya untuk aplikasi Anda, maka Anda dapat membuatnya dengan memilih Buat grup sumber daya baru pada daftar geser-turun Grup sumber daya. Untuk informasi selengkapnya tentang cara membuat grup sumber daya, silakan lihat Panduan Pengguna AWS Resource Groups.
5. Di bawah Monitor CloudWatch Acara, pilih kotak centang untuk mengintegrasikan pemantauan Wawasan Aplikasi dengan CloudWatch Acara untuk mendapatkan wawasan dari Amazon EBS, Amazon EC2, Amazon ECSAWS CodeDeploy, AWS Health API dan notifikasi, Amazon RDS, Amazon S3, dan. AWS Step Functions
6. Di bawah Integrasikan dengan AWS Systems Manager OpsCenter, pilih kotak centang di samping Menghasilkan tindakan perbaikan AWS Systems Manager OpsCenter OpsItems untuk melihat dan mendapatkan pemberitahuan saat masalah terdeteksi untuk aplikasi yang dipilih. Untuk melacak operasi yang dilakukan untuk menyelesaikan item pekerjaan operasional, yang disebut OpsItems, yang terkait dengan AWS sumber daya Anda, berikan topik SNS ARN.
7. Anda dapat memasukkan tag secara opsional untuk membantu Anda mengidentifikasi dan mengatur sumber daya Anda. CloudWatch Application Insights mendukung grup sumber daya berbasis tag dan AWS CloudFormation berbasis tumpukan, dengan pengecualian grup. Application Auto Scaling Untuk informasi selengkapnya, silakan lihat Editor Tanda di Panduan Pengguna AWS Resource Groups dan Tag.
8. Pilih Berikutnya untuk melanjutkan melakukan pengaturan pemantauan.
9. Pada halaman Tinjauan komponen yang terdeteksi, komponen yang dipantau dan beban kerjanya yang terdeteksi secara otomatis oleh CloudWatch Application Insights dicantumkan.

    a. Untuk menambahkan beban kerja ke sebuah komponen yang berisi beban kerja simpul tunggal SAP HANA terdeteksi, silakan pilih komponennya, dan kemudian pilih Edit komponen.

> **Note**
>
> Komponen-komponen yang berisi SAP HANA multi simpul terdeteksi atau beban kerja Ketersediaan Tinggi HANA hanya mendukung satu beban kerja pada suatu komponen.



b. Untuk menambahkan sebuah beban kerja baru, silakan pilih Tambahkan beban kerja baru.



c. Setelah Anda selesai mengedit beban kerja, silakan pilih Simpan perubahan.

10. Pilih Berikutnya.
11. Pada halaman Tentukan detail komponen, silakan masukkan nama pengguna dan kata sandi.
12. Tinjau konfigurasi pemantauan aplikasi Anda, dan kemudian pilih Kirim.
13. Halaman detail aplikasi kemudian terbuka, di mana Anda dapat melihat Ringkasan aplikasi, daftar Komponen dan beban kerja yang dipantau, serta Komponen dan beban kerja yang tidak dipantau. Jika Anda memilih tombol radio yang ada di samping komponen atau beban kerja, maka Anda juga dapat melihat Riwayat konfigurasi, Pola Log, dan Tanda apa pun yang telah Anda buat sebelumnya. Ketika Anda mengirimkan konfigurasi Anda, akun Anda menyebarkan semua metrik dan alarm untuk sistem SAP HANA Anda, hal ini dapat memakan waktu hingga 2 jam.

Mengelola pemantauan basis data SAP HANA Anda

Anda dapat mengelola kredensial pengguna, metrik, dan jalur log untuk basis data SAP HANA Anda dengan melakukan langkah-langkah berikut:

1. Buka konsol CloudWatch.
2. Dari panel navigasi yang ada di bagian kiri, pada Pemantauan infrastruktur, pilih Wawasan Aplikasi.
3. Halaman Wawasan Aplikasi akan menampilkan daftar aplikasi yang dipantau dengan Wawasan Aplikasi, dan status pemantauan untuk masing-masing aplikasi tersebut.

4. Pada Komponen yang dipantau, pilih tombol radio yang ada di samping nama komponen. Kemudian, silakan pilih Kelola pemantauan.
5. Pada log grup instans EC2, Anda dapat memperbarui jalur log yang sudah ada, kumpulan pola log, dan nama grup log. Selain itu, Anda juga dapat menambahkan hingga tiga Log aplikasi tambahan.
6. Pada Metrik, Anda dapat memilih metrik SAP HANA sesuai dengan kebutuhan Anda. Nama-nama metrik SAP HANA diawali dengan `hanadb`. Anda dapat menambahkan hingga 40 metrik untuk masing-masing komponen.
7. Pada konfigurasi HANA, silakan masukkan kata sandi dan nama pengguna untuk basis data SAP HANA. Ini adalah nama pengguna dan kata sandi yang digunakan CloudWatch agen Amazon untuk terhubung ke database SAP HANA.
8. Di bawah Alarm khusus, Anda dapat menambahkan alarm tambahan untuk dipantau oleh CloudWatch Application Insights.
9. Tinjau konfigurasi pemantauan aplikasi Anda dan kemudian pilih Kirim. Ketika Anda mengirimkan konfigurasi Anda, akun Anda akan memperbarui semua metrik dan alarm untuk sistem SAP HANA Anda, hal ini dapat memakan waktu hingga 2 jam.

Mengonfigurasi ambang batas alarm

CloudWatch Application Insights secara otomatis membuat CloudWatch metrik Amazon untuk menonton alarm, bersama dengan ambang batas untuk metrik tersebut. Alarm tersebut statusnya akan beralih menjadi status `ALARM` saat metrik melampaui ambang batas untuk jumlah periode evaluasi yang ditentukan. Perlu diperhatikan bahwa pengaturan ini tidak dipertahankan oleh Wawasan Aplikasi.

Untuk mengedit sebuah alarm untuk satu metrik, Anda bisa melakukan langkah-langkah berikut:

1. Buka konsol CloudWatch.
2. Pada panel navigasi yang ada di bagian kiri, pilih Alarm>Semua alarm.
3. Pilih tombol radio di sebelah alarm yang dibuat secara otomatis oleh CloudWatch Application Insights. Kemudian pilih Tindakan, dan pilih Edit dari menu geser-turun.
4. Edit parameter berikut ini pada Metrik.
    a. Pada Statistik, pilih salah satu statistik atau persentil yang telah ditentukan sebelumnya, atau tentukan persentil yang Anda kehendaki. Misalnya, `p95.45`.

b. Pada Periode, silakan pilih periode evaluasi untuk alarm. Saat Anda melakukan evaluasi alarm, masing-masing periode digabungkan menjadi satu titik data.

5. Edit parameter-parameter berikut ini pada Kondisi.

   a. Pilih apakah metriknya harus lebih besar dari, kurang dari, atau sama dengan ambang batas.

   b. Tentukan nilai ambang batasnya.

6. Pada Konfigurasi tambahan, silakan edit parameter-parameter berikut.

   a. Pada Titik data ke alarm, silakan tentukan jumlah titik data, atau periode evaluasi, yang harus berada dalam status `ALARM` untuk memulai alarm. Ketika dua nilai tersebut cocok, sebuah alarm akan tercipta dan berada dalam status `ALARM` jika jumlah periode berturut-turut yang ditentukan terlampaui. Untuk membuat `m` dari alarm `n`, tentukan nilai yang lebih rendah untuk titik data pertama, bukan untuk titik data yang kedua. Untuk informasi selengkapnya tentang cara melakukan evaluasi alarm, silakan lihat Mengevaluasi alarm.

   b. Pada Perlakuan data yang hilang, silakan pilih perilaku alarm saat ada beberapa titik data yang tidak ditemukan. Untuk informasi selengkapnya tentang perlakuan data yang hilang, lihat Mengonfigurasi cara CloudWatch alarm menangani data yang hilang.

   c. Jika alarm menggunakan sebuah persentil sebagai statistik yang dipantau, maka kotak Persentil dengan sampel rendah akan ditampilkan. Pilih apakah Anda akan mengevaluasi atau mengabaikan kasus yang memiliki tingkat sampel yang kecil. Jika Anda memilih abaikan (status alarm tidak berubah), maka status alarm saat ini akan tetap dipertahankan ketika ukuran sampel terlalu kecil. Untuk informasi selengkapnya tentang persentil dengan sampel yang rendah, silakan lihat CloudWatch Alarm berbasis persentil dan sampel data rendah.

7. Pilih Berikutnya.

8. Pada Notifikasi, silakan pilih topik SNS yang akan diberitahukan sebagai notifikasi ketika alarm berada dalam status `ALARM`, status `OK`, atau status `INSUFFICIENT_DATA`.

9. Pilih Perbarui alarm.

Melihat dan memecahkan masalah SAP HANA yang terdeteksi oleh Application Insights CloudWatch

Bagian berikut ini akan menjelaskan kepada Anda langkah-langkah untuk membantu Anda menyelesaikan skenario pemecahan masalah-masalah umum yang terjadi ketika Anda mengonfigurasi pemantauan untuk SAP HANA pada Wawasan Aplikasi.

Topik-topik penyelesaian masalah

- Basis data SAP HANA mencapai batas alokasi memori
- Peristiwa disk penuh
- Pencadangan SAP HANA berhenti berjalan

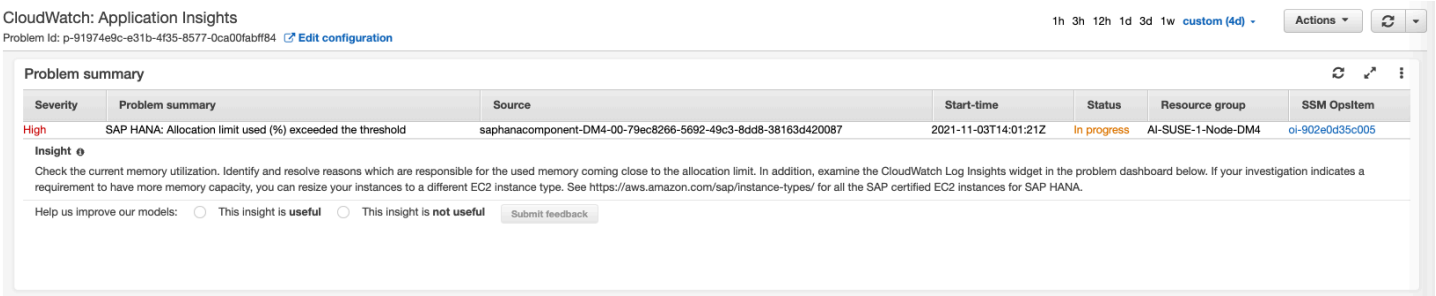
Basis data SAP HANA mencapai batas alokasi memori

Deskripsi

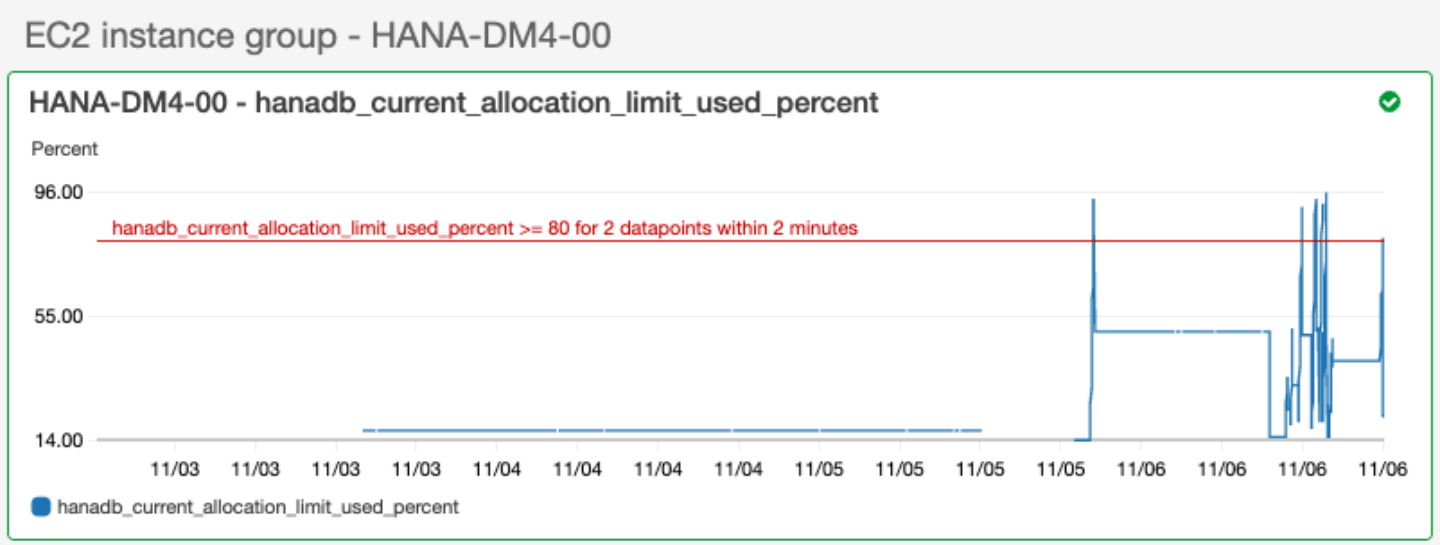
Aplikasi SAP Anda yang didukung oleh basis data SAP HANA mengalami kerusakan karena adanya tekanan memori yang tinggi, yang menyebabkan menurunnya performa aplikasi.

Resolusi

Anda dapat mengidentifikasi lapisan aplikasi yang menyebabkan masalah dengan memeriksa dasbor yang dibuat secara dinamis, yang menunjukkan kepada Anda metrik-metrik dan potongan file log terkait. Dalam contoh berikut, permasalahannya mungkin karena beban data yang besar dalam sistem SAP HANA.



Alokasi memori yang digunakan telah melebihi ambang batas 80 persen dari total batas alokasi memori yang ditentukan.



Grup log menunjukkan skema `BNR-DATA` dan tabel `IMDBMASTER_30003` telah kehabisan memori. Selain itu, grup log juga menunjukkan waktu yang tepat kapan masalah itu terjadi, batas lokasi global saat ini, memori bersama, ukuran kode, dan ukuran alokasi reservasi OOM.

Log Group: SAP_HANA_TRACE-AI-SUSE-1-Node-DM4, Log Type: SAP_HANA_TRACE, AWS::SAPHANA.OutOfMemory

#	@timestamp	@message
1	2021-11-06T13:31:23.317Z	GLOBAL_ALLOCATION_LIMIT (GAL) = 55.78gb (59901001728b), SHARED_MEMORY = 567.77mb (595357696b), CODE_SIZE = 2.94gb (3162550272b), OOM_RESERVATION_ALLOCATOR_SIZE = 96.14mb (100810752b)
2	2021-11-06T13:31:23.316Z	[2867]{311260}[22/963854] 2021-11-06 13:00:44.999570 e OOM_Notification Statement.cc(04580) : oom exception occurred at 'imdbmaster:30003': conn_id=311260, stmt_id=1336853818011966, stmt_hash=17e1ccc2b5f460604ceae8c98690fd01, sql=CAL…
3	2021-11-06T13:31:23.316Z	[3033]{311513}[22/967162] 2021-11-06 13:31:17.163640 e Memory mmReportMemoryProblems.cpp(01805) : OUT OF MEMORY occurred.
4	2021-11-06T13:31:23.316Z	Current callstack: 1: 0x00007f824538dd35 in MemoryManager::PoolAllocator::notifyOOMImpl(unsigned long, unsigned long, bool, ltt::allocation_failure_type, bool)+0x1b1 at mmPoolAllocator.cpp:2284 (libhdbbasis.so) 2: 0x00007f824524a7ad …
5	2021-11-06T13:31:23.316Z	[2822]{-1}[-1/-1] 2021-11-06 13:31:17.175597 e Memory mmReportMemoryProblems.cpp(01805) : OUT OF MEMORY occurred.
6	2021-11-06T13:31:23.316Z	Current callstack: 1: 0x00007f824538dd35 in MemoryManager::PoolAllocator::notifyOOMImpl(unsigned long, unsigned long, bool, ltt::allocation_failure_type, bool)+0x1b1 at mmPoolAllocator.cpp:2284 (libhdbbasis.so) 2: 0x00007f824524a7ad …
7	2021-11-06T13:31:23.316Z	GLOBAL_ALLOCATION_LIMIT (GAL) = 55.78gb (59901001728b), SHARED_MEMORY = 567.77mb (595357696b), CODE_SIZE = 2.94gb (3162550272b), OOM_RESERVATION_ALLOCATOR_SIZE = 96.14mb (100810752b)
8	2021-11-06T13:31:17.318Z	GLOBAL_ALLOCATION_LIMIT (GAL) = 55.78gb (59901001728b), SHARED_MEMORY = 567.77mb (595357696b), CODE_SIZE = 2.94gb (3162550272b), OOM_RESERVATION_ALLOCATOR_SIZE = 96.14mb (100810752b)
9	2021-11-06T13:31:17.317Z	GLOBAL_ALLOCATION_LIMIT (GAL) = 55.78gb (59901001728b), SHARED_MEMORY = 567.77mb (595357696b), CODE_SIZE = 2.94gb (3162550272b), OOM_RESERVATION_ALLOCATOR_SIZE = 96.14mb (100810752b)
10	2021-11-06T13:31:17.317Z	[3033]{311513}[22/967162] 2021-11-06 13:31:17.100223 w Memory mmPoolAllocator.cpp(01212) : Out of memory for Pool/PersistenceManager/PersistentSpace/DefaultLPA/DataPage, size 167772168, alignment=40968, flags 0x0, reason GLOBAL_ALLOC…
11	2021-11-06T13:31:17.317Z	GLOBAL_ALLOCATION_LIMIT (GAL) = 55.78gb (59901001728b), SHARED_MEMORY = 567.77mb (595357696b), CODE_SIZE = 2.94gb (3162550272b), OOM_RESERVATION_ALLOCATOR_SIZE = 96.14mb (100810752b)
12	2021-11-06T13:31:17.317Z	[3033]{311513}[22/967162] 2021-11-06 13:31:17.163640 e Memory mmReportMemoryProblems.cpp(01805) : OUT OF MEMORY occurred.
13	2021-11-06T13:31:17.317Z	Current callstack: 1: 0x00007f824538dd35 in MemoryManager::PoolAllocator::notifyOOMImpl(unsigned long, unsigned long, bool, ltt::allocation_failure_type, bool)+0x1b1 at mmPoolAllocator.cpp:2284 (libhdbbasis.so) 2: 0x00007f824524a7ad …
14	2021-11-06T13:31:17.317Z	[2822]{-1}[-1/-1] 2021-11-06 13:31:17.170707 w Memory mmPoolAllocator.cpp(01212) : Out of memory for Pool/malloc/libhdbbasement.so, size 422808, alignment=88, flags 0x0, reason GLOBAL_ALLOCATION_LIMIT
15	2021-11-06T13:31:17.317Z	[2822]{-1}[-1/-1] 2021-11-06 13:31:17.175597 e Memory mmReportMemoryProblems.cpp(01805) : OUT OF MEMORY occurred.
16	2021-11-06T13:31:17.317Z	Current callstack: 1: 0x00007f824538dd35 in MemoryManager::PoolAllocator::notifyOOMImpl(unsigned long, unsigned long, bool, ltt::allocation_failure_type, bool)+0x1b1 at mmPoolAllocator.cpp:2284 (libhdbbasis.so) 2: 0x00007f824524a7ad …
17	2021-11-06T13:31:17.317Z	GLOBAL_ALLOCATION_LIMIT (GAL) = 55.78gb (59901001728b), SHARED_MEMORY = 567.77mb (595357696b), CODE_SIZE = 2.94gb (3162550272b), OOM_RESERVATION_ALLOCATOR_SIZE = 96.14mb (100810752b)
18	2021-11-06T13:31:16.317Z	GLOBAL_ALLOCATION_LIMIT (GAL) = 55.78gb (59901001728b), SHARED_MEMORY = 567.77mb (595357696b), CODE_SIZE = 2.94gb (3162550272b), OOM_RESERVATION_ALLOCATOR_SIZE = 96.14mb (100810752b)
19	2021-11-06T13:31:16.317Z	GLOBAL_ALLOCATION_LIMIT (GAL) = 55.78gb (59901001728b), SHARED_MEMORY = 567.77mb (595357696b), CODE_SIZE = 2.94gb (3162550272b), OOM_RESERVATION_ALLOCATOR_SIZE = 96.14mb (100810752b)

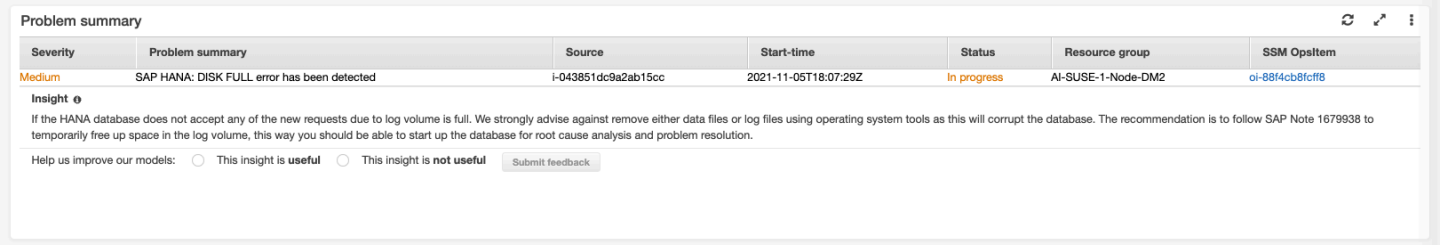
Peristiwa disk penuh

Deskripsi

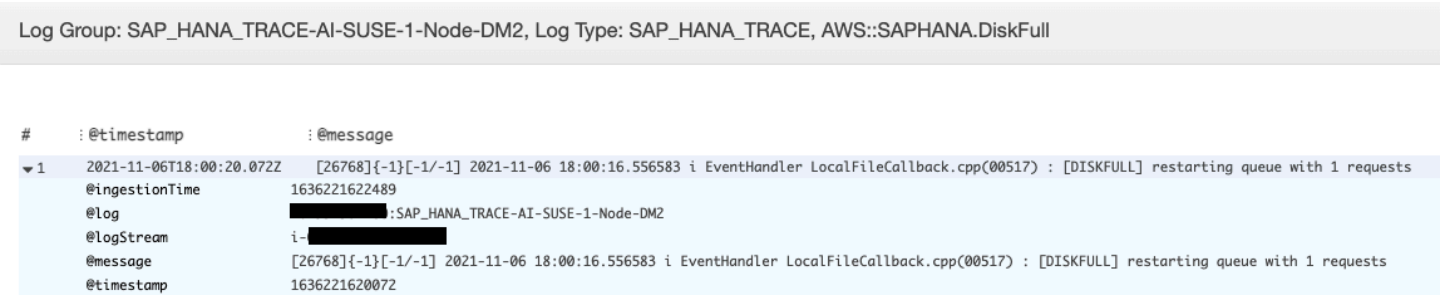
Aplikasi SAP Anda yang didukung oleh basis data SAP HANA berhenti merespons, hal ini menyebabkan aplikasi tidak mampu mengakses basis data.

Resolusi

Anda dapat mengidentifikasi lapisan basis data yang menyebabkan masalah dengan memeriksa dasbor yang dibuat secara dinamis, yang menunjukkan kepada Anda metrik-metrik dan potongan file log terkait. Dalam contoh berikut, permasalahannya mungkin terjadi pada administrator yang gagal mengaktifkan pencadangan log otomatis, hal ini menyebabkan direktori sap/hana/log terisi.



Widget grup log yang ada di dasbor masalah menunjukkan peristiwa `DISKFULL` tersebut.



Pencadangan SAP HANA berhenti berjalan

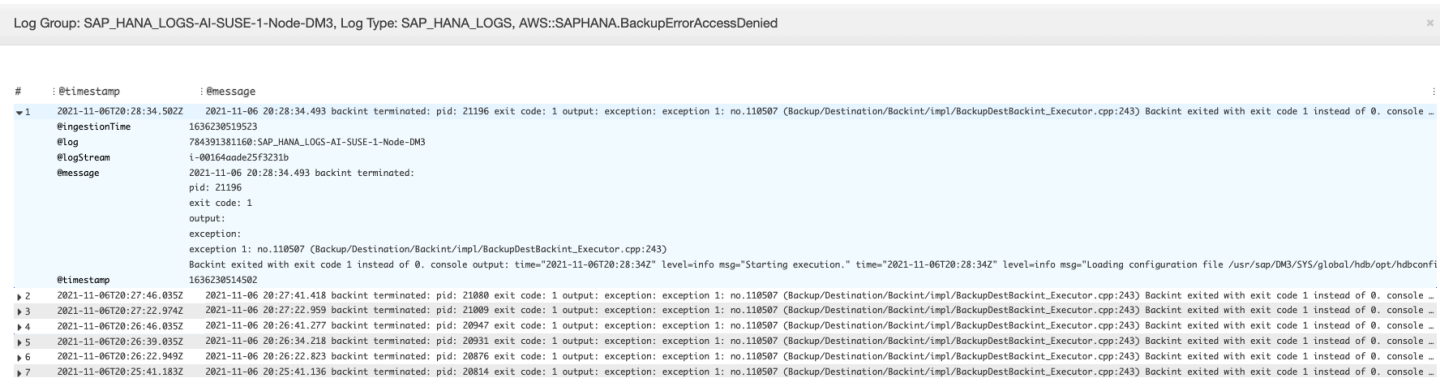
Deskripsi

Aplikasi SAP Anda yang didukung oleh basis data SAP HANA telah berhenti bekerja.

Resolusi

Anda dapat mengidentifikasi lapisan basis data yang menyebabkan masalah dengan memeriksa dasbor yang dibuat secara dinamis, yang menunjukkan kepada Anda metrik-metrik dan potongan file log terkait.

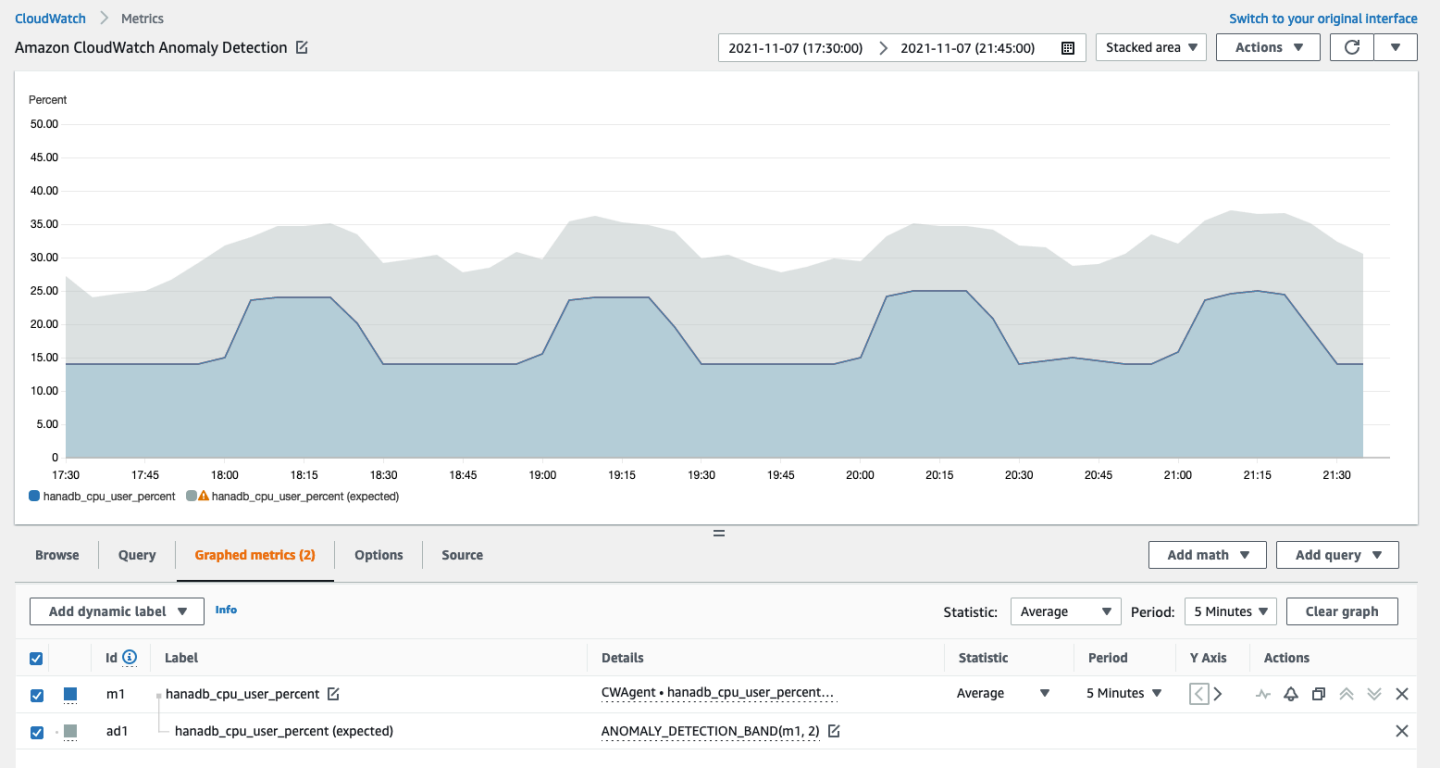
Widget grup log yang ada di dasbor masalah menunjukkan peristiwa `ACCESS DENIED` tersebut. Ini termasuk informasi tambahan, seperti bucket S3, folder bucket S3, dan Wilayah bucket S3.



Deteksi anomali untuk SAP HANA

Untuk metrik SAP HANA tertentu, seperti jumlah jumlah utas, CloudWatch menerapkan algoritma statistik dan pembelajaran mesin untuk menentukan ambang batas. Algoritma ini akan terus menganalisis metrik-metrik basis data SAP HANA, menentukan garis dasar normal, dan anomali permukaan dengan intervensi pengguna minimal. Algoritma tersebut akan menghasilkan model deteksi anomali, yang kemudian akan menghasilkan serangkaian nilai yang diharapkan yang mewakili perilaku metrik normal.

Algoritma deteksi anomali bertanggung jawab atas perubahan-perubahan musiman dan tren metrik. Perubahan-perubahan musiman tersebut dapat dilakukan setiap jam, setiap hari, atau setiap minggu, sebagaimana ditunjukkan pada contoh penggunaan CPU SAP HANA.



Setelah Anda membuat model, deteksi CloudWatch anomali terus mengevaluasi model dan membuat penyesuaian untuk memastikan bahwa itu seakurat mungkin. Ini termasuk melatih ulang model itu untuk menyesuaikan jika nilai-nilai metrik berkembang dari waktu ke waktu atau mengalami perubahan secara tiba-tiba. Hal ini juga mencakup prediktor untuk meningkatkan model untuk metrik-metrik yang bersifat musiman, runcing, atau jarang.

Penyelesaian Masalah Wawasan Aplikasi untuk SAP HANA

Bagian ini akan menjelaskan kepada Anda tentang langkah-langkah yang akan membantu Anda dalam mengatasi kesalahan-kesalahan umum yang ditampilkan dalam dasbor Wawasan Aplikasi.

Tidak dapat menambahkan metrik monitor lebih dari 60 metrik monitor

Kesalahan yang ditampilkan: `Component cannot have more than 60 monitored metrics.`

Penyebab galat: `The current metric limit is 60 monitor metrics per component.`

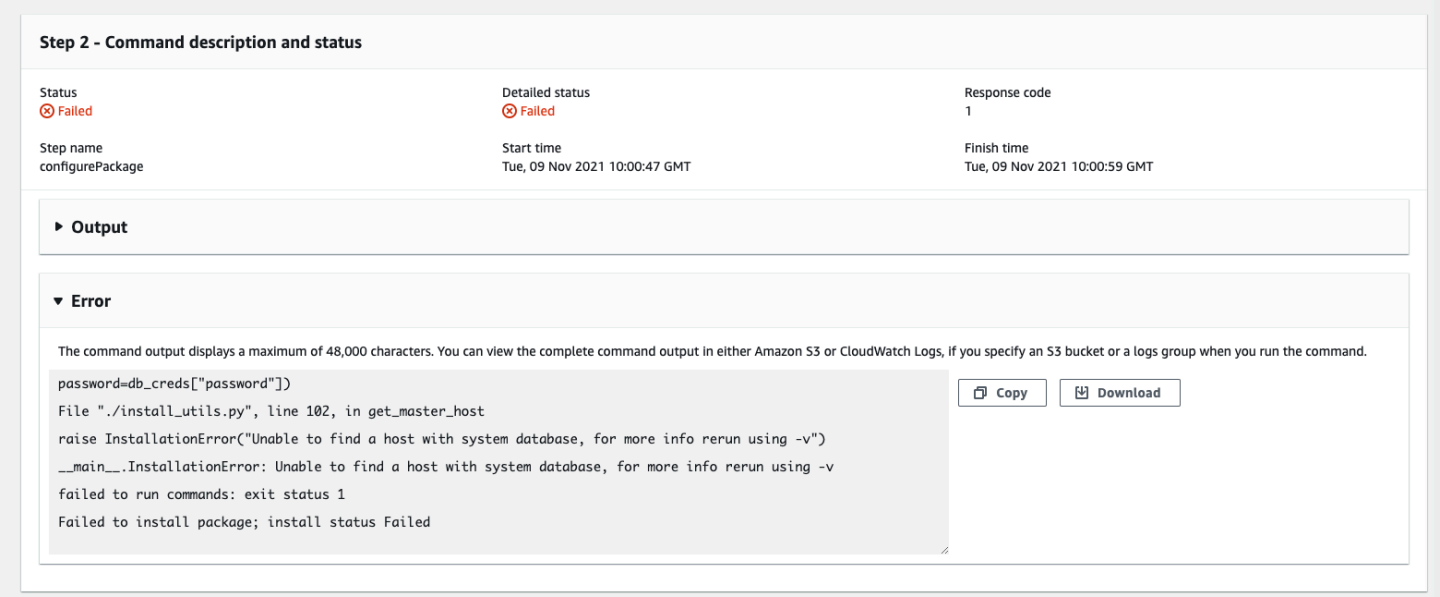
Resolusi: Menghapus metrik yang tidak diperlukan agar bisa mematuhi batas yang ditentukan.

Tidak ada metrik atau alarm SAP yang muncul setelah proses onboarding.

Kesalahan yang ditampilkan: Dalam perintah eksekusi AWS Systems Manager, `AWS-ConfigureAWSPackage` gagal.

Outputnya menunjukkan kesalahan berikut:

```
Unable to find a host with system database, for more info rerun using -v
```



Resolusi: Nama pengguna dan kata sandi mungkin salah. Verifikasi bahwa nama pengguna dan kata sandi sudah valid, dan kemudian jalankan kembali proses onboarding.

Outputnya menunjukkan kesalahan instalasi berikut ini:

```
ERROR: Can not execute `setup.py` since setuptools is not available in the build
 environment.
```

atau

```
[SSL: CERTIFICATE_VERIFY_FAILED]
```

Resolusi: Menginstal Python menggunakan salah satu contoh perintah SUSE Linux berikut:

Contoh 1

```
sudo zypper install -y python36
```

Contoh 2

Menginstal Python 3.8 versi terbaru.

```
wget https://www.python.org/ftp/python/3.8.<LATEST_RELEASE>/
Python-3.8.<LATEST_RELEASE>.tgz
tar xf Python-3.*
cd Python-3.*/
sudo zypper install make gcc-c++ gcc automake autoconf libtool
sudo zypper install zlib-devel
sudo zypper install libopenssl-devel libffi-devel
./configure --with-ensurepip=install
sudo make
sudo make install
sudo su
python3.8 -m pip install --upgrade pip setuptools wheel
```

Tutorial: Mengatur pemantauan untuk SAP NetWeaver

Tutorial ini menunjukkan cara mengkonfigurasi Amazon CloudWatch Application Insights untuk mengatur pemantauan untuk SAP. NetWeaver Anda dapat menggunakan dasbor otomatis CloudWatch Application Insights untuk memvisualisasikan detail masalah, mempercepat pemecahan masalah, dan mengurangi mean time to resolution (MTTR) untuk server aplikasi SAP Anda. NetWeaver

CloudWatch Wawasan Aplikasi untuk topik SAP NetWeaver

- Lingkungan yang didukung

Lingkungan yang didukung

CloudWatch Application Insights mendukung penyebaran sumber AWS daya untuk sistem dan pola berikut.

- Penerapan Sistem NetWeaver Standar SAP.
- Penerapan NetWeaver Terdistribusi SAP di beberapa instans Amazon EC2.
- Pengaturan ketersediaan NetWeaver tinggi SAP lintas-AZ — SAP NetWeaver dengan ketersediaan tinggi yang dikonfigurasi di dua Availability Zone menggunakan pengelompokan SUSE/RHEL.

Sistem operasi yang didukung

CloudWatch Application Insights untuk SAP NetWeaver didukung pada sistem operasi berikut:

- Oracle Linux 8:
- Red Hat Enterprise Linux 7.6
- Red Hat Enterprise Linux 7.7
- Red Hat Enterprise Linux 7.9
- Red Hat Enterprise Linux 8.1
- Red Hat Enterprise Linux 8.2
- Red Hat Enterprise Linux 8.4
- Red Hat Enterprise Linux 8.6
- SUSE Linux Enterprise Server 15 untuk SAP
- SUSE Linux Enterprise Server 15 SP1 untuk SAP

- SUSE Linux Enterprise Server 15 SP2 untuk SAP
- SUSE Linux Enterprise Server 15 SP3 untuk SAP
- SUSE Linux Enterprise Server 15 SP4 untuk SAP
- SUSE Linux Enterprise Server 12 SP4 untuk SAP
- SUSE Linux Enterprise Server 12 SP5 untuk SAP
- SUSE Linux Enterprise Server 15 kecuali pola Ketersediaan Tinggi
- SUSE Linux Enterprise Server 15 SP1 kecuali pola Ketersediaan Tinggi
- SUSE Linux Enterprise Server 15 SP2 kecuali pola Ketersediaan Tinggi
- SUSE Linux Enterprise Server 15 SP3 kecuali pola Ketersediaan Tinggi
- SUSE Linux Enterprise Server 15 SP4 kecuali pola Ketersediaan Tinggi
- SUSE Linux Enterprise Server 12 SP4 kecuali pola Ketersediaan Tinggi
- SUSE Linux Enterprise Server 12 SP5 kecuali pola Ketersediaan Tinggi

Fitur-fitur

CloudWatch Application Insights untuk SAP NetWeaver 7.0x—7.5x (termasuk ABAP Platform) menyediakan fitur-fitur berikut:

- Deteksi beban NetWeaver kerja SAP otomatis
- Pembuatan NetWeaver alarm SAP otomatis berdasarkan ambang batas statis
- Pengenalan pola NetWeaver log SAP otomatis
- Dasbor Kesehatan untuk SAP NetWeaver
- Dasbor masalah untuk SAP NetWeaver

Prasyarat

Anda harus melakukan prasyarat berikut untuk mengonfigurasi SAP NetWeaver dengan Application Insights: CloudWatch

- AWS Pengaktifan Systems Manager — Instal Agen SSM di instans Amazon EC2 Anda, dan aktifkan instans untuk SSM. Untuk informasi tentang cara melakukan instalasi agen SSM, silakan lihat Pengaturan AWS Systems Manager di Panduan Pengguna Systems Manager AWS .
- Peran instans Amazon EC2 — Anda harus melampirkan peran instans Amazon EC2 berikut untuk mengonfigurasi pemantauan SAP Anda. NetWeaver

  - Anda harus melampirkan peran `AmazonSSMManagedInstanceCore` untuk mengaktifkan Systems Manager. Untuk informasi selengkapnya, silakan lihat contoh kebijakan berbasis identitas AWS Systems Manager.
  - Anda harus melampirkan `CloudWatchAgentServerPolicy` kebijakan untuk mengaktifkan metrik instans dan log yang akan dipancarkan. CloudWatch Untuk informasi selengkapnya, lihat Membuat peran IAM dan pengguna untuk digunakan dengan CloudWatch agen.
- AWS grup sumber daya — Anda harus membuat grup sumber daya yang mencakup semua AWS sumber daya terkait yang digunakan oleh tumpukan aplikasi Anda untuk onboard aplikasi Anda ke Wawasan CloudWatch Aplikasi. Ini termasuk instans Amazon EC2, Amazon EFS, dan volume Amazon EBS yang menjalankan server aplikasi SAP Anda. NetWeaver Jika ada beberapa NetWeaver sistem SAP per akun, kami sarankan Anda membuat satu grup sumber daya yang mencakup AWS sumber daya untuk setiap sistem SAP NetWeaver . Untuk informasi selengkapnya tentang cara membuat grup sumber daya, silakan lihat Resource Groups AWS dan Tags User Guide.
- Izin IAM — Untuk pengguna yang tidak memiliki akses administratif, Anda harus membuat kebijakan AWS Identity and Access Management (IAM) yang memungkinkan Wawasan Aplikasi untuk membuat peran terkait layanan dan melampirkannya ke identitas pengguna. Untuk informasi selengkapnya tentang cara membuat kebijakan IAM, silakan lihat kebijakan IAM.
- Peran terkait layanan — Wawasan Aplikasi menggunakan peran terkait layanan AWS Identity and Access Management (IAM). Sebuah peran terkait layanan dibuat untuk Anda saat Anda membuat aplikasi Wawasan Aplikasi pertama Anda di konsol Wawasan Aplikasi. Untuk informasi selengkapnya, lihat Menggunakan peran tertaut layanan untuk Wawasan Aplikasi CloudWatch.
- CloudWatch Agen Amazon - Application Insights menginstal dan mengonfigurasi agen. CloudWatch Jika Anda telah menginstal CloudWatch agen, Application Insights mempertahankan konfigurasi Anda. Untuk menghindari konflik gabungan, hapus konfigurasi sumber daya yang ingin Anda gunakan di Application Insights dari file konfigurasi CloudWatch agen yang ada. Untuk informasi selengkapnya, lihat Buat atau edit file konfigurasi CloudWatch agen secara manual.

Siapkan server NetWeaver aplikasi SAP Anda untuk pemantauan

Gunakan langkah-langkah berikut untuk mengatur pemantauan untuk server NetWeaver aplikasi SAP Anda.

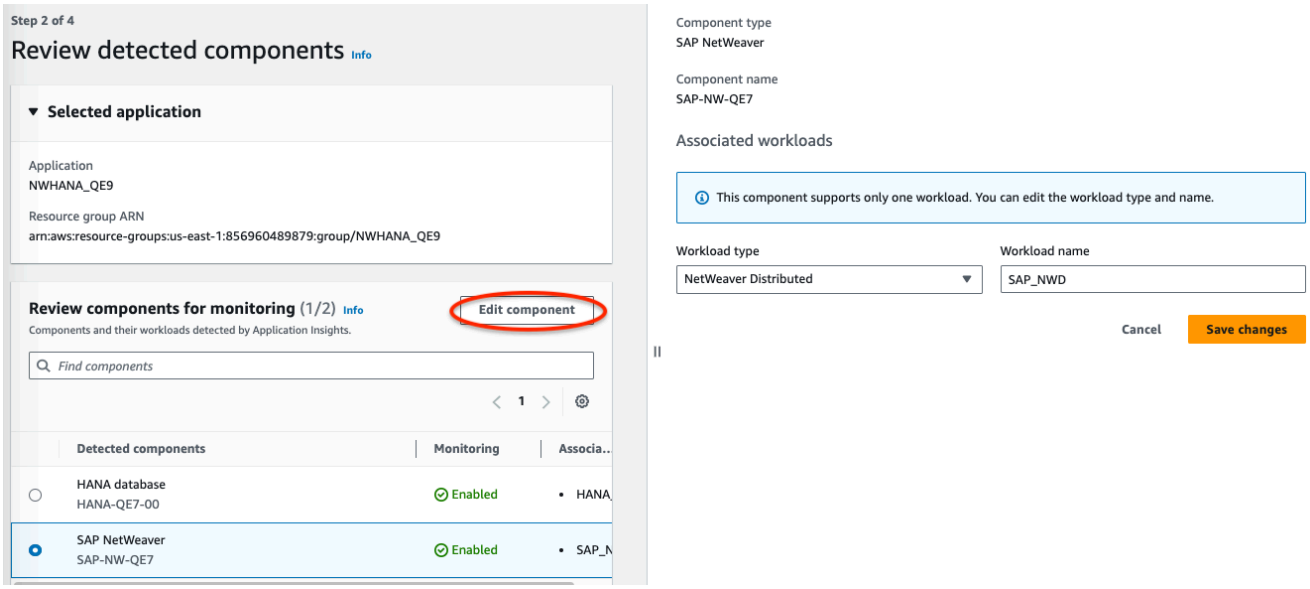
Cara menyiapkan pemantauan

1. Buka konsol CloudWatch .

2. Dari panel navigasi kiri, pada Wawasan, pilih Wawasan Aplikasi.
3. Halaman Wawasan Aplikasi akan menampilkan daftar aplikasi yang dipantau dengan Wawasan Aplikasi, dan status pemantauan untuk masing-masing aplikasi tersebut. Di bagian pojok kanan atas, pilih Tambahkan aplikasi.
4. Pada halaman Tentukan detail aplikasi, dari daftar dropdown di bawah Grup sumber daya, pilih grup AWS sumber daya yang Anda buat yang berisi sumber daya SAP NetWeaver Anda. Jika Anda belum membuat grup sumber daya untuk aplikasi Anda, maka Anda dapat membuatnya dengan memilih Create grup sumber daya baru pada daftar dropdown Grup sumber daya.
5. Di bawah Pemantauan otomatis sumber daya baru, pilih kotak centang agar Wawasan Aplikasi dapat secara otomatis memantau sumber daya yang ditambahkan ke grup sumber daya aplikasi setelah onboarding.
6. Di bawah Memantau EventBridge peristiwa, pilih kotak centang untuk mengintegrasikan pemantauan Wawasan Aplikasi dengan CloudWatch Acara untuk mendapatkan wawasan dari Amazon EBS, Amazon EC2, Amazon ECS AWS CodeDeploy, AWS Health API dan notifikasi, Amazon RDS, Amazon S3, dan. AWS Step Functions
7. Di bawah Integrasikan dengan AWS Systems Manager OpsCenter, pilih kotak centang di samping Menghasilkan tindakan perbaikan AWS Systems Manager OpsCenter OpsItems untuk melihat dan mendapatkan pemberitahuan saat masalah terdeteksi untuk aplikasi yang dipilih. Untuk melacak operasi yang dilakukan untuk menyelesaikan item pekerjaan operasional, yang disebut OpsItems, yang terkait dengan AWS sumber daya Anda, berikan topik SNS ARN.
8. Anda dapat memasukkan tag secara opsional untuk membantu Anda mengidentifikasi dan mengatur sumber daya Anda. CloudWatch Application Insights mendukung grup sumber daya berbasis tag dan AWS CloudFormation berbasis tumpukan, dengan pengecualian grup. Application Auto Scaling Untuk informasi selengkapnya, silakan lihat Editor Tanda di Panduan Pengguna AWS Resource Groups dan Tag.
9. Untuk meninjau komponen yang terdeteksi, pilih Berikutnya.
10. Pada halaman Tinjauan komponen yang terdeteksi, komponen yang dipantau dan beban kerjanya yang terdeteksi secara otomatis oleh CloudWatch Application Insights dicantumkan.

    - Untuk mengedit jenis dan nama beban kerja, pilih Komponen Edit.

> **Note**
>
> Komponen yang berisi beban kerja NetWeaver Terdistribusi atau Ketersediaan NetWeaver Tinggi yang terdeteksi hanya mendukung satu beban kerja pada komponen.



11. Pilih Berikutnya.
12. Pada halaman Tentukan detail komponen, pilih Berikutnya.
13. Tinjau konfigurasi pemantauan aplikasi Anda, kemudian pilih Kirim.
14. Halaman detail aplikasi terbuka, di mana Anda dapat melihat Ringkasan aplikasi, Dasbor, Komponen, dan Beban Kerja. Anda juga dapat melihat Riwayat konfigurasi, Pola log, dan Tanda apa pun yang telah Anda buat. Setelah Anda mengirimkan aplikasi Anda, CloudWatch Application Insights menyebarkan semua metrik dan alarm untuk NetWeaver sistem SAP Anda, yang dapat memakan waktu hingga satu jam.

Kelola pemantauan server NetWeaver aplikasi SAP Anda

Gunakan langkah-langkah berikut untuk mengelola pemantauan server NetWeaver aplikasi SAP Anda.

Cara mengelola pemantauan

1. Buka konsol CloudWatch .
2. Dari panel navigasi kiri, pada Wawasan, pilih Wawasan Aplikasi.
3. Pilih tab Tampilan daftar.
4. Halaman Wawasan Aplikasi akan menampilkan daftar aplikasi yang dipantau dengan Wawasan Aplikasi, dan status pemantauan untuk masing-masing aplikasi tersebut.
5. Pilih aplikasi Anda.
6. Pilih tab Komponen.
7. Pada Komponen yang dipantau, pilih tombol radio yang ada di samping nama komponen. Kemudian, pilih Kelola pemantauan.
8. Pada Log instans, Anda dapat memperbarui jalur log yang ada, kumpulan pola log, dan nama grup log. Selain itu, Anda juga dapat menambahkan hingga tiga Log aplikasi tambahan.
9. Di bawah Metrik, Anda dapat memilih NetWeaver metrik SAP sesuai dengan kebutuhan Anda. Nama NetWeaver metrik SAP diawali dengan. `sap` Anda dapat menambahkan hingga 40 metrik untuk masing-masing komponen.
10. Di bawah Alarm khusus, Anda dapat menambahkan alarm tambahan untuk dipantau oleh CloudWatch Application Insights.
11. Tinjau konfigurasi pemantauan aplikasi Anda dan kemudian pilih Simpan. Ketika Anda mengirimkan konfigurasi Anda, akun Anda memperbarui semua metrik dan alarm untuk sistem SAP NetWeaver Anda.

Melihat dan memecahkan NetWeaver masalah SAP yang terdeteksi oleh Application Insights CloudWatch

Bagian berikut menyediakan langkah-langkah untuk membantu Anda menyelesaikan skenario pemecahan masalah umum yang terjadi saat Anda mengonfigurasi pemantauan untuk SAP NetWeaver pada Application Insights.

Topik-topik penyelesaian masalah

- Masalah konektivitas NetWeaver basis data SAP
- Masalah ketersediaan NetWeaver aplikasi SAP

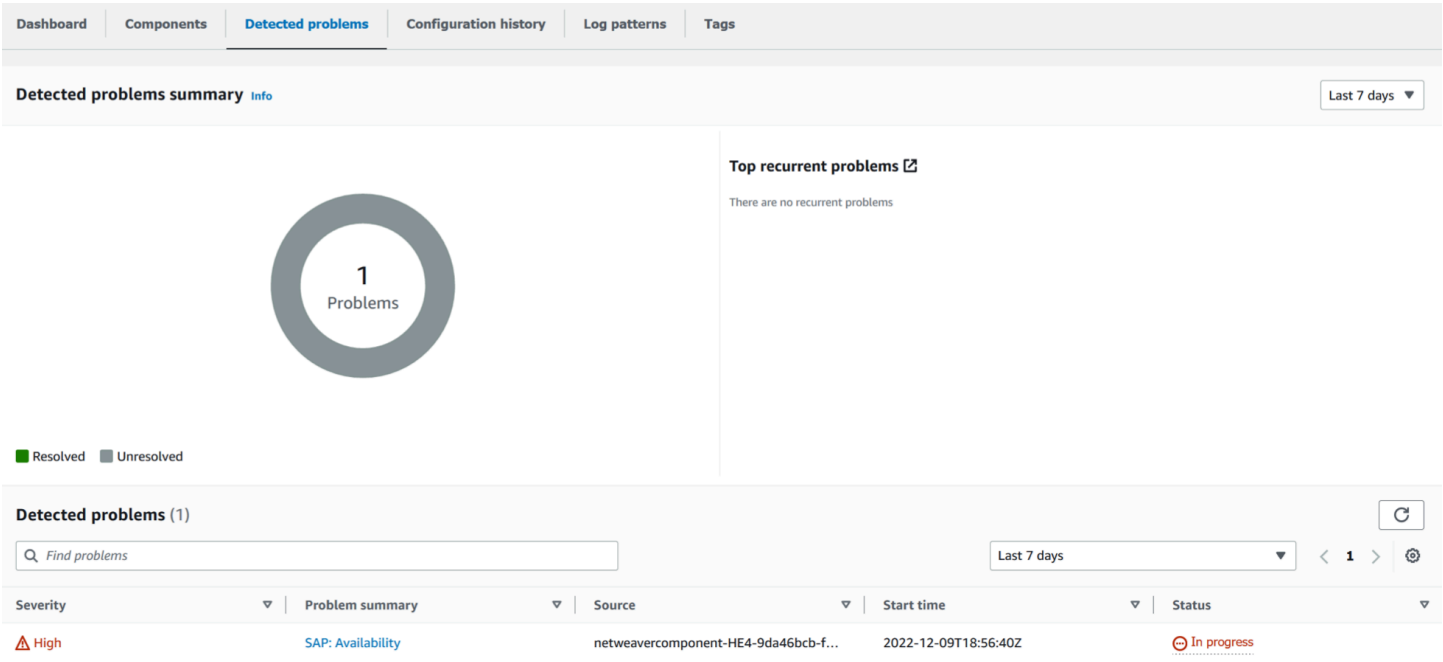
Masalah konektivitas NetWeaver basis data SAP

Deskripsi

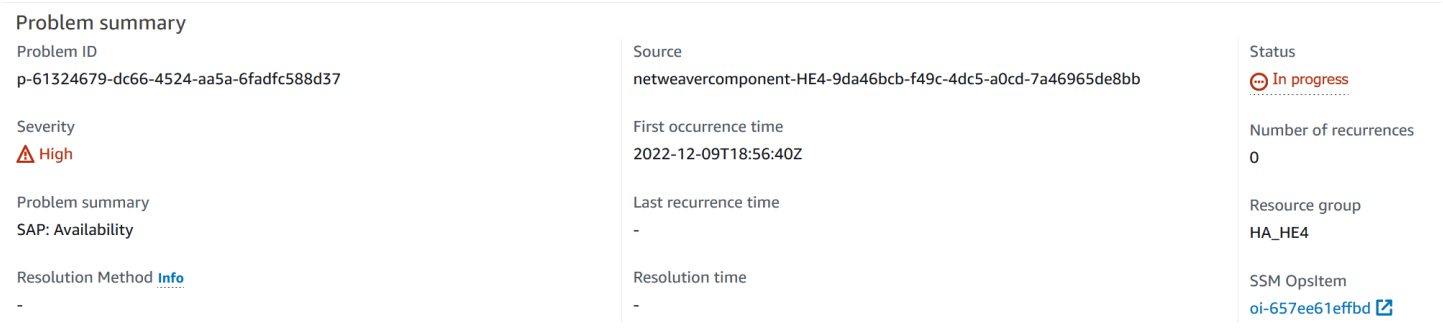
NetWeaver Aplikasi SAP Anda mengalami masalah konektivitas database.

Penyebab

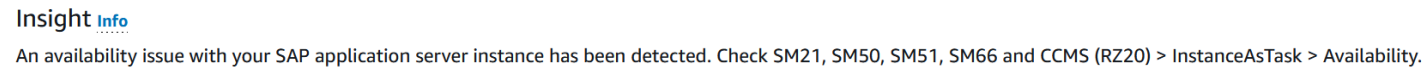
Anda dapat mengidentifikasi masalah konektivitas dengan membuka konsol CloudWatch Application Insights dan memeriksa dasbor masalah SAP NetWeaver Application Insights. Pilih tautan pada Ringkasan masalah untuk melihat masalah spesifik.



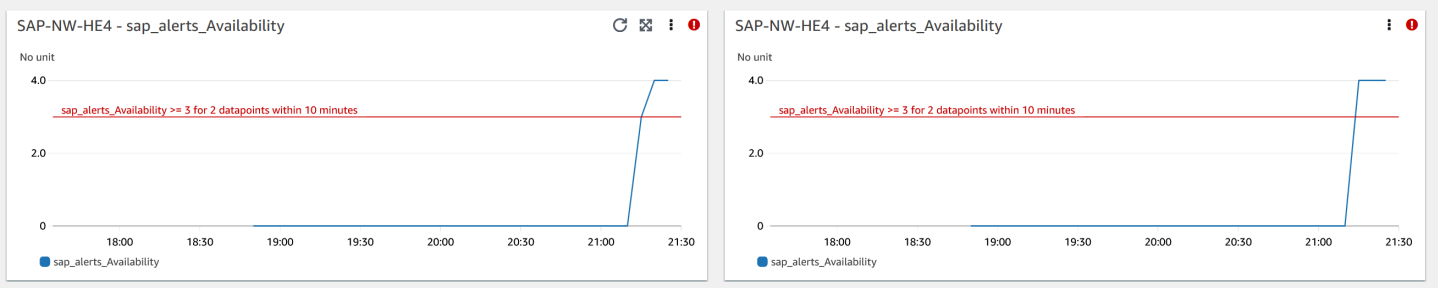
Dalam contoh berikut, pada Ringkasan masalah, SAP: Ketersediaan adalah masalahnya.



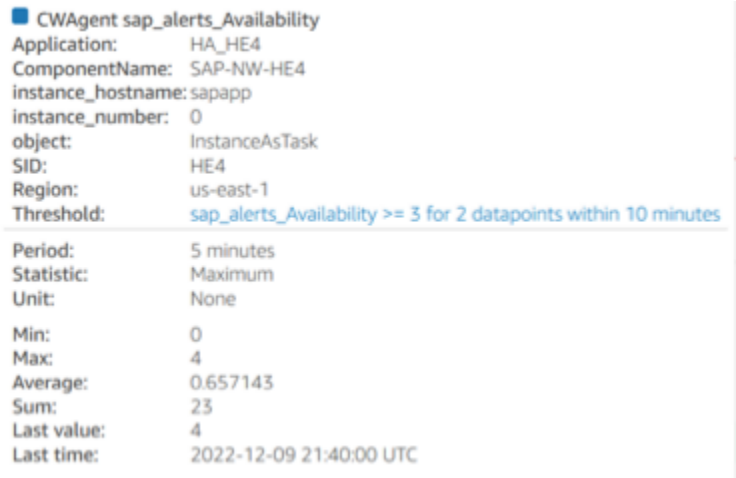
Langsung dilanjutkan dengan Ringkasan masalah, bagian Wawasan memberikan lebih banyak konteks tentang kesalahan dan di mana Anda bisa mendapatkan informasi lebih lanjut tentang penyebab masalah.



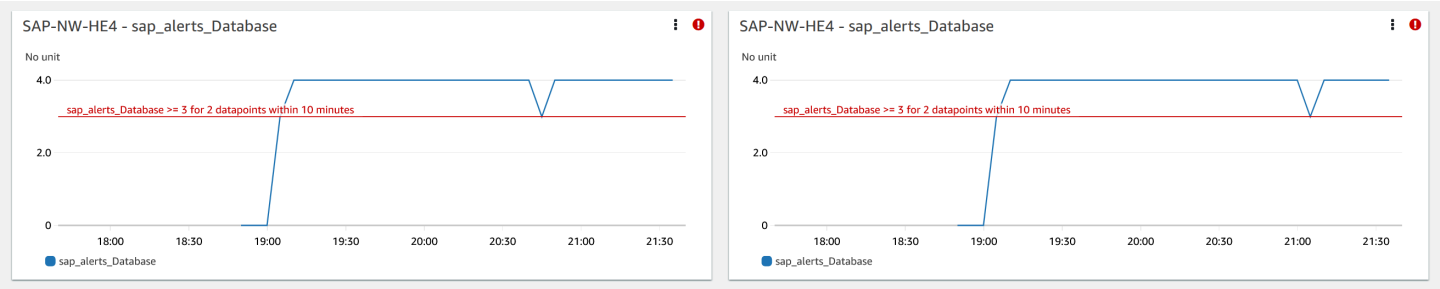
Di dasbor masalah yang sama, Anda dapat melihat log dan metrik terkait yang telah dikelompokkan dalam deteksi masalah untuk membantu Anda mengisolasi penyebab kesalahan. `sap_alerts_Availability`Metrik melacak ketersediaan NetWeaver sistem SAP dari waktu ke waktu. Anda dapat menggunakan pelacakan riawayat untuk mengkorelasikan ketika metrik menginisiasi status kesalahan atau melanggar ambang batas alarm. Dalam contoh berikut, ada masalah ketersediaan dengan NetWeaver sistem SAP. Contoh menunjukkan dua alarm karena ada dua instans server aplikasi SAP dan alarm dibuat untuk setiap instans.



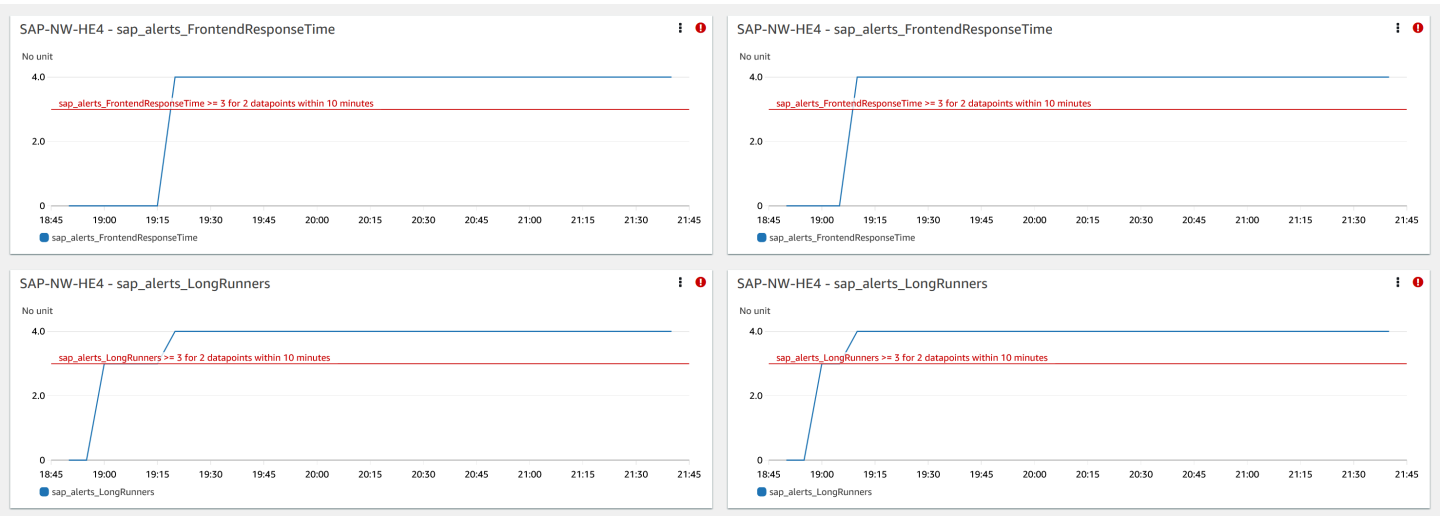
Untuk informasi selengkapnya tentang setiap alarm, arahkan kursor ke nama metrik `sap_alerts_Availability`.



Dalam contoh berikut, `sap_alerts_Database` metrik menunjukkan bahwa lapisan basis data memiliki masalah atau kegagalan. Alarm ini menunjukkan bahwa SAP NetWeaver memiliki masalah menghubungkan atau berkomunikasi dengan database-nya.



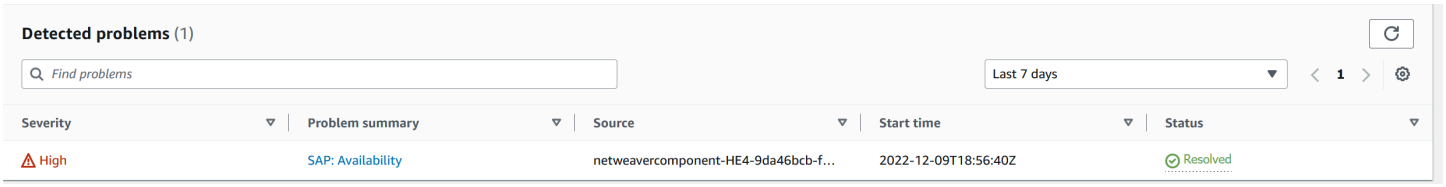
Karena database adalah sumber daya utama untuk SAP NetWeaver, Anda mungkin mendapatkan banyak alarm terkait ketika database memiliki masalah atau kegagalan. Dalam contoh berikut, metrik `sap_alerts_FrontendResponseTime` dan metrik `sap_alerts_LongRunners` dimulai karena basis data tidak tersedia.



Penyelesaian

Wawasan Aplikasi memantau masalah yang terdeteksi setiap jam. Jika tidak ada entri log terkait baru dalam file NetWeaver log SAP Anda, entri log lama akan diperlakukan sebagai diselesaikan. Anda harus memperbaiki kondisi kesalahan apa pun yang terkait dengan CloudWatch alarm. Setelah kondisi kesalahan diperbaiki, alarm teratasi ketika alarm dan log dipulihkan. Ketika semua kesalahan CloudWatch log dan alarm diselesaikan, Application Insights berhenti mendeteksi kesalahan dan masalah secara otomatis diselesaikan dalam waktu satu jam. Kami menyarankan Anda menyelesaikan semua kondisi kesalahan log dan peringatan sehingga Anda hanya memiliki masalah terbaru di dasbor masalah.

Dalam contoh berikut, masalah ketersediaan SAPdiselesaikan.



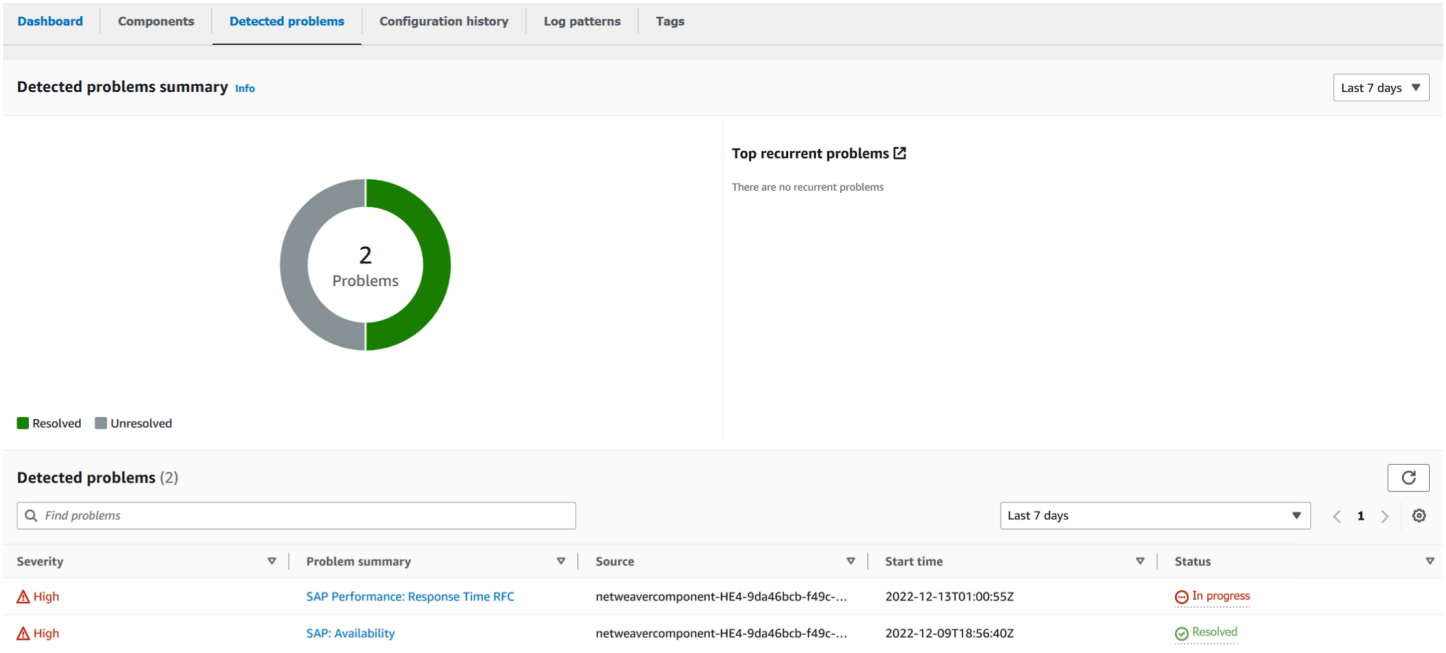
Masalah ketersediaan NetWeaver aplikasi SAP

Deskripsi

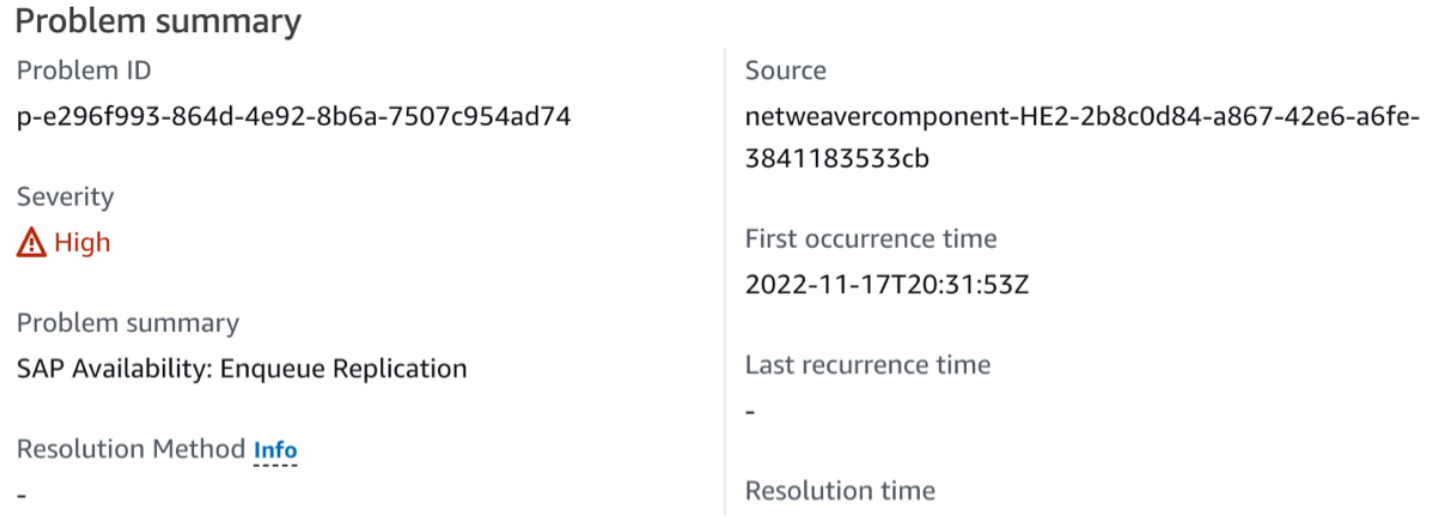
Replikasi SAP NetWeaver High Availability Enqueue Anda berhenti bekerja.

Penyebab

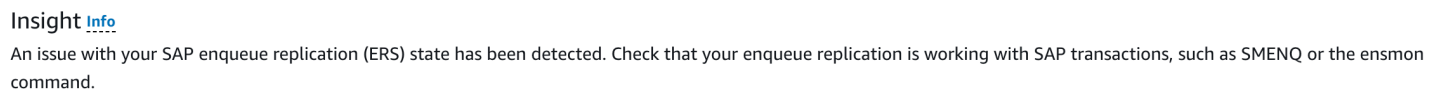
Anda dapat mengidentifikasi masalah konektivitas dengan membuka konsol CloudWatch Application Insights dan memeriksa dasbor masalah SAP NetWeaver Application Insights. Pilih tautan pada Ringkasan masalah untuk melihat masalah spesifik.



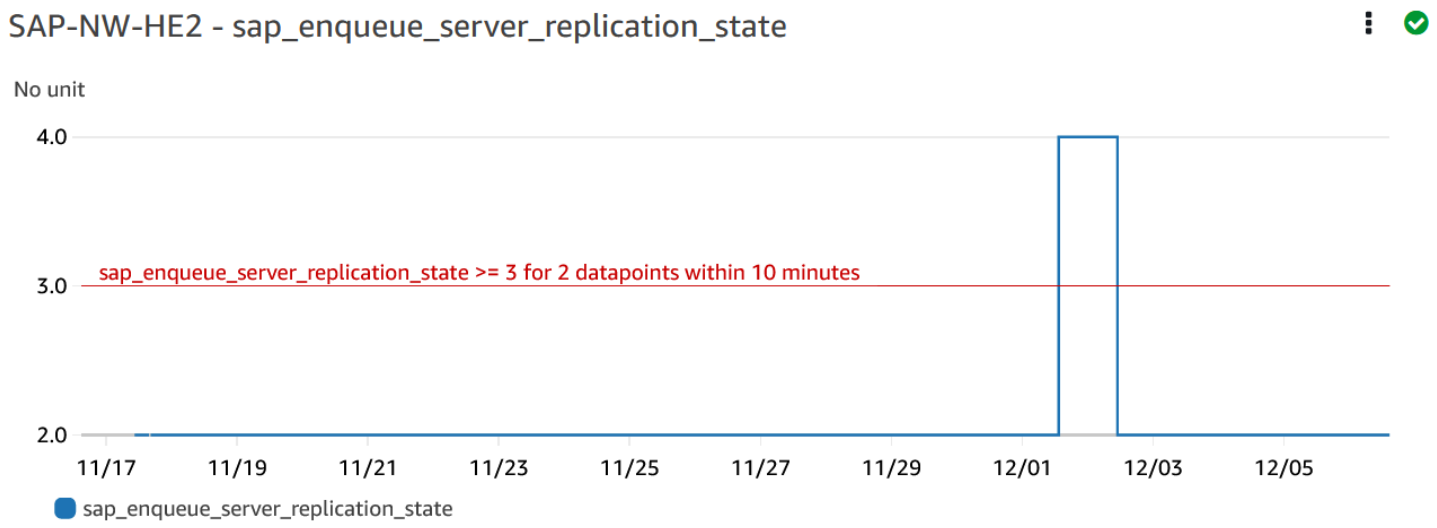
Dalam contoh berikut, pada Ringkasan masalah, Replikasi Antre Ketersediaan Tinggi menjadi masalahnya.



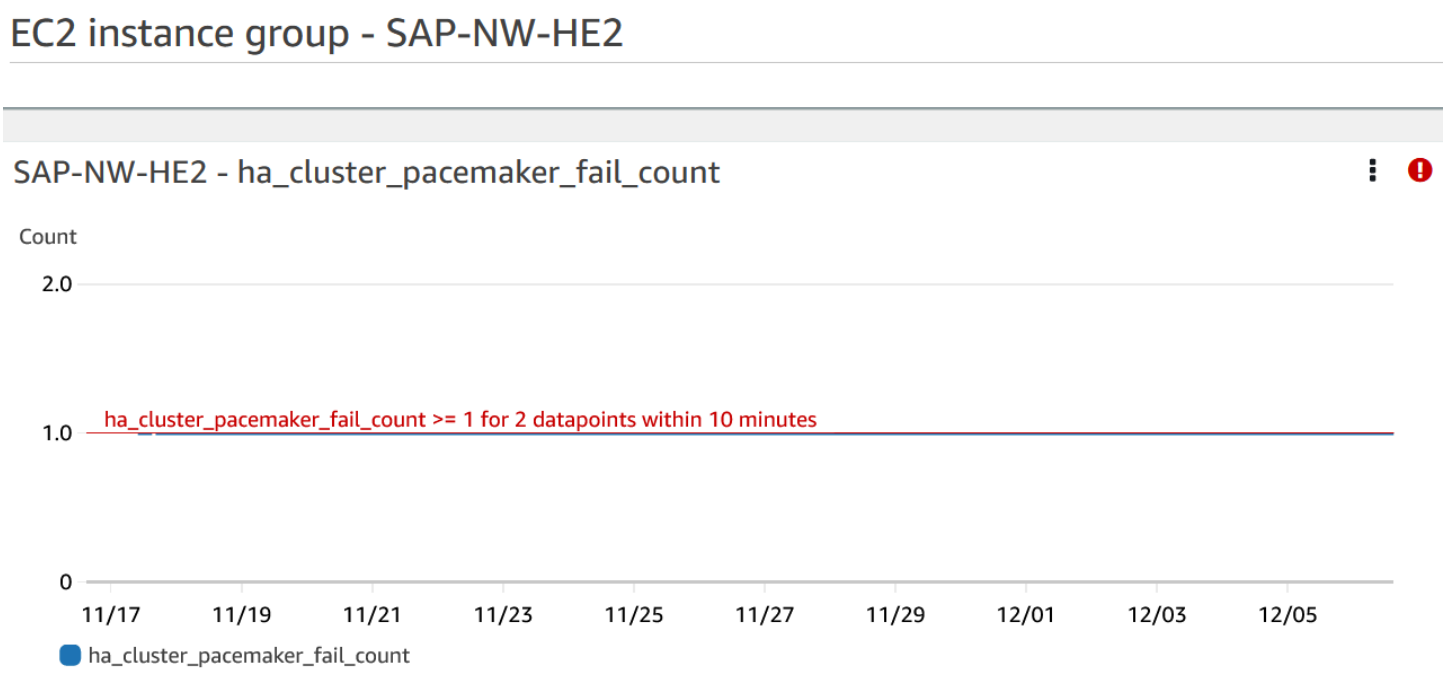
Langsung dilanjutkan dengan Ringkasan masalah, bagian Wawasan memberikan lebih banyak konteks tentang kesalahan dan di mana Anda bisa mendapatkan informasi lebih lanjut tentang penyebab masalah.



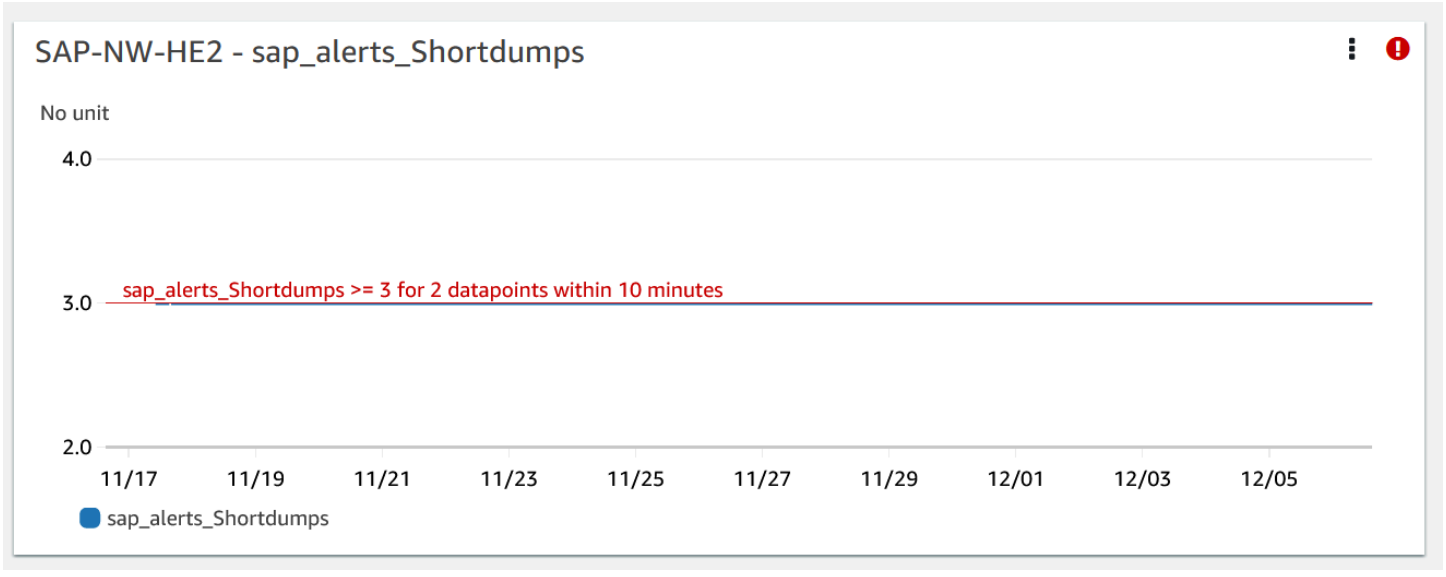
Contoh berikut menunjukkan dasbor masalah tempat Anda melihat log dan metrik yang dikelompokkan untuk membantu Anda mengisolasi penyebab kesalahan. `sap_enqueue_server_replication_state`Metrik melacak nilai dari sepanjang waktu. Anda dapat menggunakan pelacakan riawayat untuk mengkorelasikan ketika metrik menginisiasi status kesalahan atau melanggar ambang batas alarm.



Dalam contoh berikut, metrik `ha_cluster_pacemaker_fail_count` menunjukkan bahwa klaster pacemaker ketersediaan tinggi mengalami kegagalan sumber daya. Sumber daya alat pacu jantung tertentu yang memiliki jumlah gagal lebih besar dari atau sama dengan satu diidentifikasi di dasbor komponen.

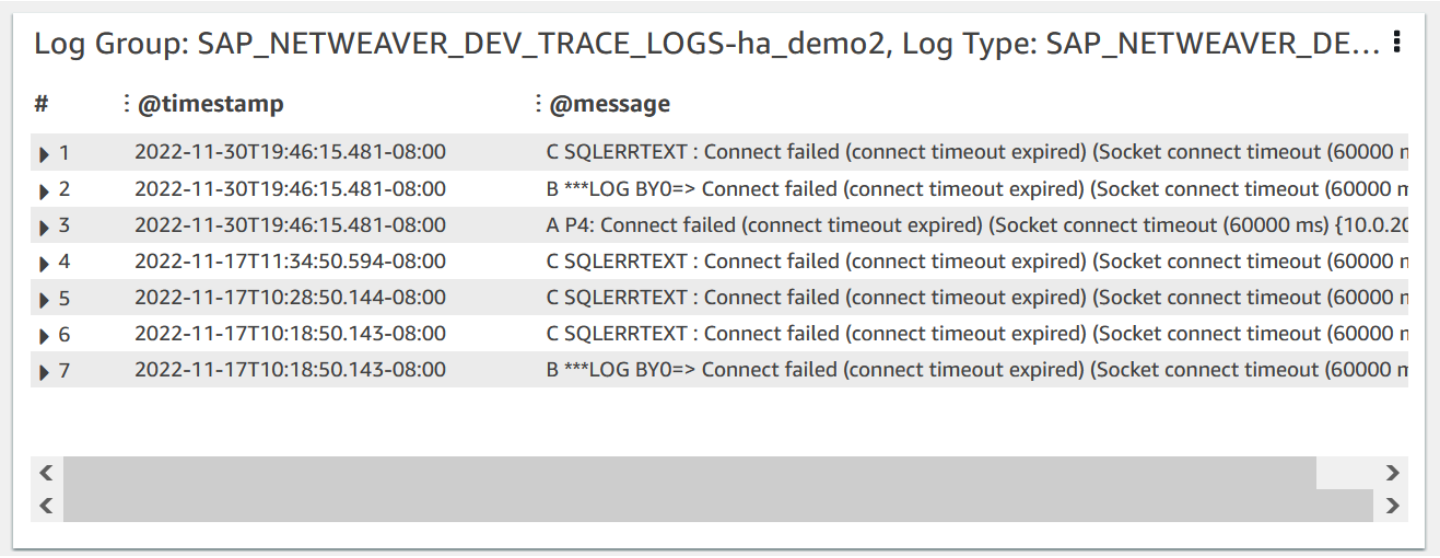


Contoh berikut menunjukkan metrik `sap_alerts_Shortdumps`, yang menunjukkan bahwa performa aplikasi SAP berkurang ketika masalah terdeteksi.

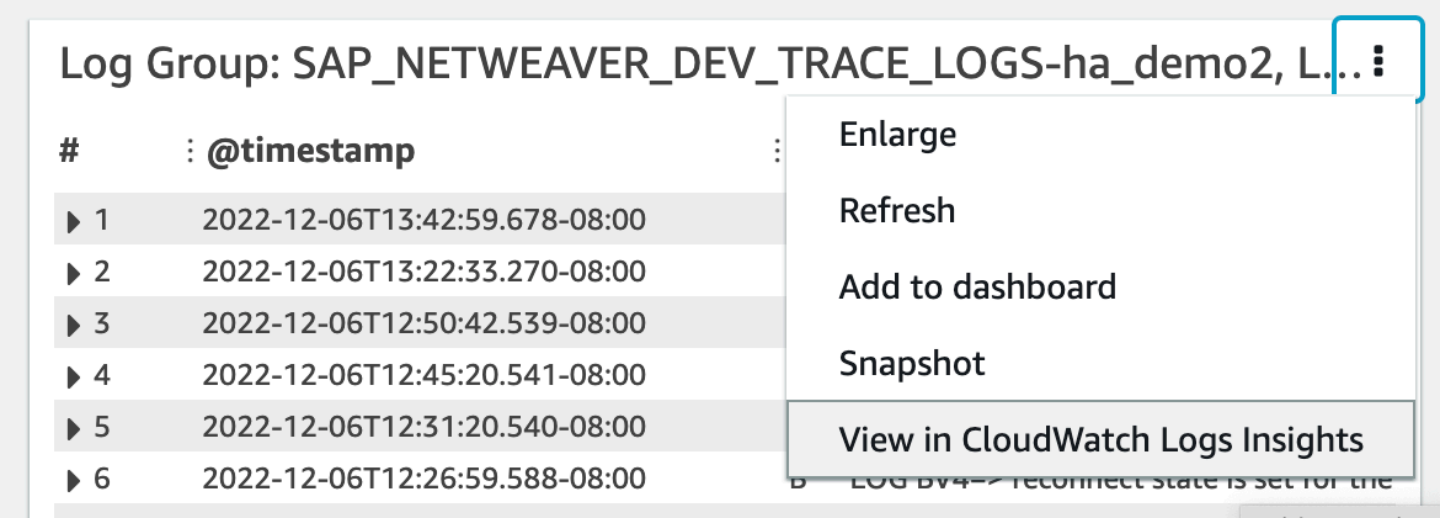


Log

Entri log sangat membantu untuk mendapatkan pemahaman yang lebih baik tentang masalah yang terjadi pada NetWeaver lapisan SAP ketika masalah terdeteksi. Widget grup log di dasbor masalah menunjukkan waktu spesifik masalah.



Untuk melihat informasi rinci tentang log, pilih tiga titik vertikal di sudut kanan atas, dan pilih Lihat di CloudWatch Wawasan Log.



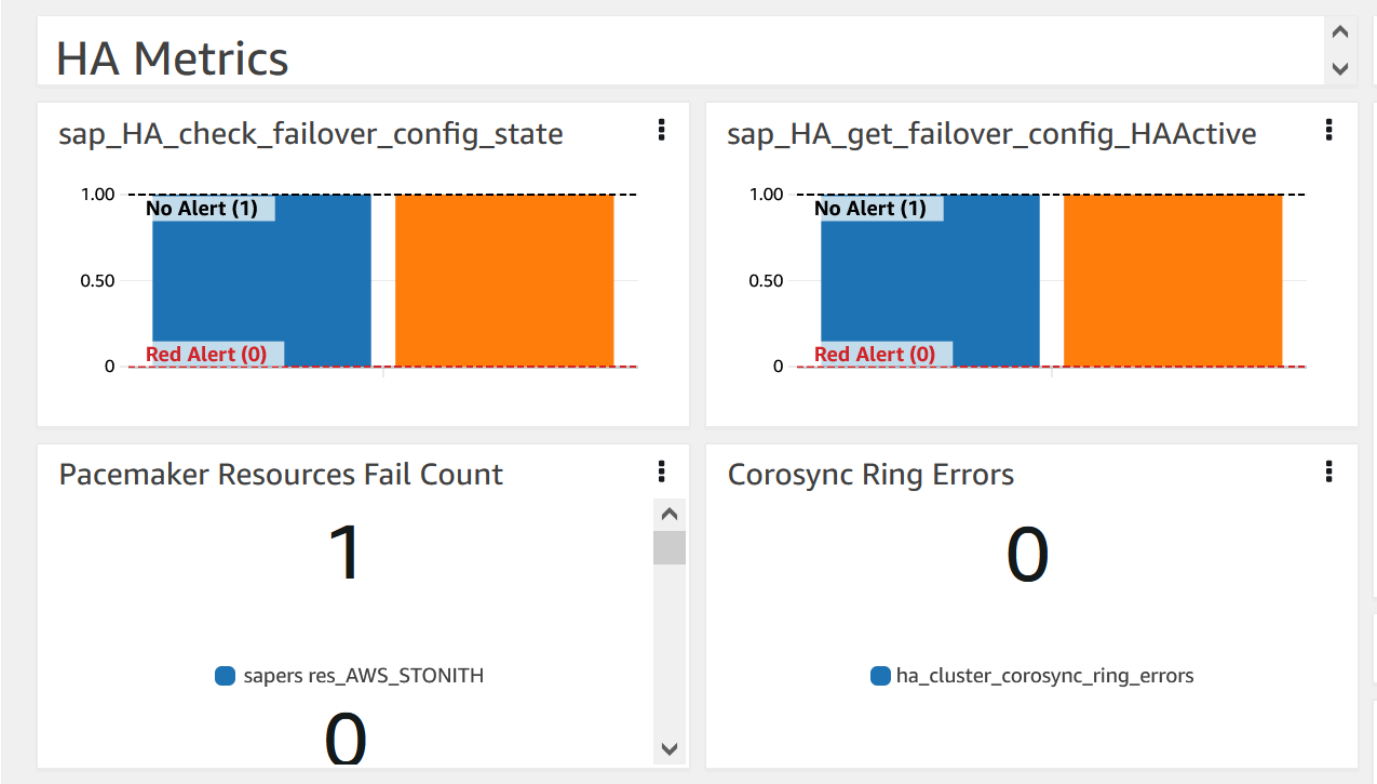
Gunakan langkah-langkah berikut untuk mendapatkan informasi selengkapnya tentang metrik dan alarm yang ditampilkan di dasbor masalah.

Untuk mendapatkan informasi selengkapnya tentang metrik dan alarm.

1. Buka konsol CloudWatch .

2. Pada panel navigasi sebelah kiri, di Pemantauan infrastruktur, pilih Wawasan Aplikasi. Kemudian, pilih tab Tampilan daftar, dan pilih aplikasi Anda.
3. Pilih tab Komponen. Kemudian, pilih NetWeaver komponen SAP yang ingin Anda dapatkan informasi lebih lanjut.

   Contoh berikut menunjukkan bagian Metrik HA dengan metrik `ha_cluster_pacemaker_fail_count` yang ditampilkan di dasbor masalah.

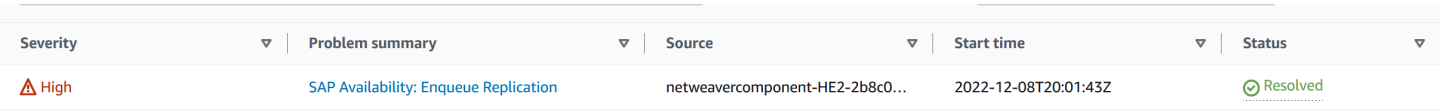


Penyelesaian

Wawasan Aplikasi memantau masalah yang terdeteksi setiap jam. Jika tidak ada entri log terkait baru dalam file NetWeaver log SAP Anda, entri log lama akan diperlakukan sebagai diselesaikan. Anda harus memperbaiki kondisi kesalahan apa pun yang berhubungan dengan masalah ini.

Untuk `sap_alerts_Shortdumps` alarm, Anda harus menyelesaikan peringatan di NetWeaver sistem SAP dengan menggunakan kode transaksi `RZ20 # R3Abap # Shortdumps` untuk menavigasi ke peringatan CCMS. Untuk informasi selengkapnya tentang peringatan CCMS, silakan lihat situs web SAP. Selesaikan semua peringatan CCMS di pohon Shortdumps. Setelah semua peringatan diselesaikan dalam NetWeaver sistem SAP, CloudWatch tidak lagi melaporkan metrik dalam keadaan alarm.

Ketika semua kesalahan CloudWatch log dan alarm diselesaikan, Application Insights berhenti mendeteksi kesalahan dan masalah secara otomatis diselesaikan dalam waktu satu jam. Kami menyarankan Anda menyelesaikan semua kondisi kesalahan log dan peringatan sehingga Anda hanya memiliki masalah terbaru di dasbor masalah. Dalam contoh berikut, masalah SAP Netweaver High Availability Enqueue Replication diselesaikan.



Severity	Problem summary	Source	Start time	Status
High	SAP Availability: Enqueue Replication	netweavercomponent-HE2-2b8c0...	2022-12-08T20:01:43Z	Resolved

Pemecahan Masalah Wawasan Aplikasi untuk SAP NetWeaver

Bagian ini akan menjelaskan kepada Anda tentang langkah-langkah yang akan membantu Anda dalam mengatasi kesalahan-kesalahan umum yang ditampilkan dalam dasbor Wawasan Aplikasi.

Tidak dapat menambahkan metrik monitor lebih dari 60 metrik monitor

Kesalahan yang ditampilkan: `Component cannot have more than 60 monitored metrics.`

Penyebab galat: `The current metric limit is 60 monitor metrics per component.`

Resolusi: Menghapus metrik yang tidak diperlukan agar bisa mematuhi batas yang ditentukan.

Metrik SAP tidak muncul di dasbor setelah proses onboarding.

Penyebab galat: Dasbor Komponen menggunakan periode metrik lima menit untuk mengumpulkan titik data.

Resolusi: Semua metrik akan muncul di dasbor setelah lima menit.

Metrik dan peringatan SAP tidak muncul di dasbor

Gunakan langkah-langkah berikut untuk mengidentifikasi penyebab metrik dan peringatan SAP tidak muncul di dasbor setelah proses onboarding.

Mengidentifikasi masalah dengan metrik dan peringatan

1. Buka konsol CloudWatch .
2. Pada panel navigasi sebelah kiri, di Pemantauan infrastruktur, pilih Wawasan Aplikasi. Kemudian, pilih tab Tampilan daftar, dan pilih aplikasi Anda.
3. Pilih tab Riwayat konfigurasi.
4. Jika Anda melihat titik data metriknya hilang, periksa kesalahan yang terkait dengan. `prometheus-sap_host_exporter`

5. Jika Anda tidak menemukan kesalahan di langkah sebelumnya, Sambungkan ke instans Linux Anda. Untuk deployment Ketersediaan Tinggi, sambungkan ke instans Amazon EC2 klaster utama.
6. Dalam instans Anda, pastkan bahwa pengekspor berjalan dengan menggunakan perintah berikut ini. Port dafult adalah 9680. Jika Anda menggunakan port yang berbeda, ganti 9680 dengan port yang saat ini Anda gunakan.

```
curl localhost:9680/metrics
```

   Jika tidak ada data yang dikembalikan, maka pengekspor gagal memulai.
7. Untuk menemukan konvensi penamaan yang benar untuk digunakan WORKLOAD_SHORT_NAME dalam dua langkah berikutnya, jalankan perintah berikut.

> **Note**
>
> Application Insights menambahkan akhiranWORKLOAD_SHORT_NAME,, ke nama layanan tergantung pada beban kerja yang sedang berjalan. Nama pendek untuk penyebaran NetWeaver Distributed, Standard, dan High Availability adalahSAP_NWD,SAP_NWS, dan. SAP_NWH

```
sudo systemctl | grep exporter
```

8. Untuk memeriksa kesalahan dalam log layanan eksportir, jalankan perintah berikut:

```
sudo journalctl -e --unit=prometheus-sap_host_exporter_WORKLOAD_SHORT_NAME.service
```

9. Untuk memeriksa kesalahan dalam log layanan manajer eksportir, jalankan perintah berikut:

```
sudo journalctl -e --unit=prometheus-
sap_host_exporter_manager_WORKLOAD_SHORT_NAME.service
```

> **Note**
>
> Layanan ini harus aktif dan berjalan setiap saat.

Jika perintah ini tidak mengembalikan kesalahan, lanjutkan ke langkah selanjutnya.

10. Untuk memulai eksportir secara manual, jalankan perintah berikut. Kemudian, periksa output pengekspor.

```
sudo /opt/aws/sap_host_exporter/sap_host_exporter
```

Anda dapat keluar dari proses pengekspor setelah Anda memeriksa apakah ada kesalahan.

Penyebab galat: Ada beberapa kemungkinan penyebab masalah ini. Penyebab umum yaitu pengekspor tidak dapat terhubung ke salah satu instansserver aplikasi.

Resolusi

Gunakan langkah-langkah berikut untuk menghubungkan pengekspor ke instans server aplikasi. Anda akan memverifikasi bahwa instans aplikasi SAP sedang berjalan dan menggunakan SapControl untuk terhubung ke instans tersebut.

Untuk menghubungkan pengekspor ke instans server aplikasi

1. Di instans Amazon EC2 Anda, jalankan perintah berikut untuk memverifikasi bahwa aplikasi SAP sedang berjalan.

```
sapcontrol -nr <App_InstNo> -function GetProcessList
```

2. Anda harus membuat koneksi SapControl yang berfungsi. Jika koneksi SAPControl tidak berfungsi, temukan akar masalah pada instans pada aplikasi SAP yang relevan.
3. Untuk memulai pengekspor secara manual setelah Anda memperbaiki masalah koneksi SAP Control, jalankan perintah berikut:

```
sudo systemctl start prometheus-sap_host_exporter.service
```

4. Jika Anda tidak dapat menyelesaikan masalah koneksi SapControl, gunakan prosedur berikut ini sebagai perbaikan sementara.

   a. Buka konsol AWS Systems Manager.

   b. Dari panel navigasi kiri, pilih Manager Status.

   c. Di bawah Asosiasi mencari asosiasi NetWeaver sistem SAP.

```
Association Name: Equal: AWS-ApplicationInsights-
SSMSAPHostExporterAssociationForCUSTOMSAPNW<SID>-1
```

d. Pilih Asosiasi id.

e. Pilih tab Parameter dan hapus nomor server aplikasi dari additionalArguments.

f. Pilih Terapkan Asosiasi Sekarang.

> **Note**
>
> Ini hanyalah perbaikan sementara. Jika pembaruan dilakukan pada konfigurasi pemantauan komponen, instans akan ditambahkan kembali.

Melihat dan memecahkan masalah yang terdeteksi oleh Amazon CloudWatch Application Insights

Topik-topik dalam bagian ini akan memberikan informasi detail tentang masalah-masalah yang terdeteksi dan wawasan yang ditampilkan oleh Wawasan Aplikasi. Bagian ini juga menyediakan penyelesaian masalah yang disarankan untuk masalah-masalah yang terdeteksi pada akun Anda atau konfigurasi Anda.

Topik-topik penyelesaian masalah

- CloudWatch ikhtisar konsol
- Halaman ringkasan masalah Wawasan Aplikasi
- CloudWatch agen menggabungkan kegagalan konflik
- Alarm tidak dibuat
- Umpan balik
- Kesalahan konfigurasi

CloudWatch ikhtisar konsol

Ikhtisar masalah yang memengaruhi aplikasi yang dipantau dapat ditemukan di panel CloudWatch Application Insights di halaman ikhtisar konsol. CloudWatch Untuk informasi selengkapnya, lihat Memulai dengan Amazon CloudWatch Application Insights.

Panel ikhtisar CloudWatch Application Insights menampilkan hal-hal berikut:

- Tingkat keparahan masalah yang terdeteksi: Tinggi/Medium/Rendah
- Ringkasan singkat masalah
- Sumber masalah
- Waktu masalah dimulai
- Status penyelesaian masalah
- Grup sumber daya terdampak

Untuk melihat detail masalah-masalah tertentu, pada Ringkasan Masalah, pilih deskripsi masalah. Sebuah dasbor terperinci akan menampilkan wawasan mengenai masalah dan anomali metrik terkait serta snippet kesalahan log dari masalah tersebut. Anda dapat memberikan umpan balik tentang relevansi wawasan yang diberikan untuk Anda dengan memilih apakah hal itu bermanfaat atau tidak.

Jika ada sebuah sumber daya baru terdeteksi yang tidak dikonfigurasi, maka deskripsi ringkasan masalah akan membawa Anda ke panduan Edit konfigurasi untuk mengonfigurasi sumber daya baru Anda. Anda dapat melihat atau menyunting konfigurasi grup sumber daya Anda dengan memilih Tampilkan/edit konfigurasi yang ada di sudut kanan atas dasbor terperinci.

Untuk kembali ke ikhtisar, pilih Kembali ke ikhtisar, yang berada di sebelah header dasbor detail Wawasan CloudWatch Aplikasi.

Halaman ringkasan masalah Wawasan Aplikasi

Halaman ringkasan masalah Wawasan Aplikasi

CloudWatch Application Insights memberikan informasi berikut tentang masalah yang terdeteksi pada halaman ringkasan masalah:

- Ringkasan singkat masalah
- Waktu dan tanggal mulai masalah
- Tingkat keparahan masalah: Tinggi/Medium/Rendah
- Status masalah yang terdeteksi: Sedang Berlangsung/Terselesaikan
- Wawasan: Secara otomatis menghasilkan wawasan terkait masalah yang terdeteksi dan kemungkinan akar masalah
- Umpan balik tentang wawasan: Umpan balik yang Anda berikan tentang kegunaan wawasan yang dihasilkan oleh Wawasan Aplikasi CloudWatch

- Observasi terkait: Tampilan terperinci dari anomali metrik dan cuplikan kesalahan dari log yang relevan terkait masalah di berbagai komponen aplikasi

CloudWatch agen menggabungkan kegagalan konflik

CloudWatch Application Insights menginstal dan mengonfigurasi CloudWatch agen pada instans pelanggan. Ini termasuk pembuatan file konfigurasi CloudWatch agen dengan konfigurasi untuk metrik atau log. Konflik gabungan dapat terjadi jika instance pelanggan sudah memiliki file konfigurasi CloudWatch agen dengan konfigurasi berbeda yang ditentukan untuk metrik atau log yang sama. Untuk mengatasi konflik penggabungan tersebut, lakukan langkah-langkah berikut:

1. Identifikasi file konfigurasi CloudWatch agen di sistem Anda. Untuk informasi selengkapnya tentang lokasi file, silakan lihat CloudWatch file dan lokasi agen.
2. Hapus konfigurasi sumber daya yang ingin Anda gunakan di Application Insights dari file konfigurasi CloudWatch agen yang ada. Jika Anda hanya ingin menggunakan konfigurasi Application Insights, hapus file konfigurasi CloudWatch agen yang ada.

Alarm tidak dibuat

Untuk beberapa metrik, Wawasan Aplikasi akan memprediksi ambang batas alarm berdasarkan titik data sebelumnya untuk metrik tersebut. Untuk memungkinkan prediksi ini, kriteria berikut harus dipenuhi.

- Poin data terbaru – Harus ada minimal 100 titik data dalam 24 jam terakhir. Titik data tidak perlu terus-menerus dan dapat tersebar di seluruh kerangka waktu 24 jam.
- Data historis – Harus ada minimal 100 titik data yang mencakup kerangka waktu dari 15 hari sebelum tanggal saat ini hingga 1 hari sebelum tanggal saat ini. Titik data tidak perlu terus-menerus dan dapat tersebar di seluruh kerangka waktu 15 hari.

> **Note**
>
> Untuk beberapa metrik, Wawasan Aplikasi akan menunda pembuatan alarm hingga kondisi-kondisi sebelumnya terpenuhi. Dalam hal ini, Anda akan mendapatkan peristiwa riwayat konfigurasi bahwa metrik tidak memiliki titik data yang memadai untuk menetapkan ambang batas alarm.

Umpan balik

Umpan balik

Anda dapat memberikan umpan balik mengenai wawasan yang dihasilkan secara otomatis terkait masalah yang terdeteksi dengan menetapkannya sebagai berguna atau tidak berguna. Umpan balik mengenai wawasan tersebut, beserta diagnostik aplikasi Anda (anomali metrik dan pengecualian log), digunakan untuk meningkatkan deteksi masalah serupa pada masa mendatang.

Kesalahan konfigurasi

CloudWatch Application Insights menggunakan konfigurasi Anda untuk membuat telemetri pemantauan untuk komponen. Bila Wawasan Aplikasi mendeteksi adanya masalah yang terjadi pada akun Anda atau konfigurasi Anda, informasi akan diberikan di bidang Keterangan pada ringkasan Aplikasi tentang cara mengatasi masalah konfigurasi untuk aplikasi Anda.

Tabel berikut menunjukkan penyelesaian masalah yang disarankan untuk keterangan khusus.

Keterangan	Resolusi yang disarankan	Catatan tambahan
Kuota untuk CloudFormation sudah tercapai.	Application Insights membuat satu CloudFormation tumpukan untuk setiap aplikasi untuk mengelola instalasi dan konfigurasi CloudWatch agen untuk semua komponen aplikasi. Secara default, setiap AWS akun dapat memiliki 2000 tumpukan. Lihat BatasAWS CloudFormation. Untuk mengatasi hal ini, naikkan batas untuk CloudFormation tumpukan.	T/A
Tidak ada peran instans SSM pada instans berikut.	Agar Application Insights dapat menginstal dan mengonfigurasi CloudWatch agen pada instance aplikasi, AmazonSSM ManagedIn	Application Insights menghubungi SSM DescribeInstanceInformation API untuk mendapatkan daftar contohs dengan izin SSM. Setelah

Keterangan	Resolusi yang disarankan	Catatan tambahan
	stanceCore dan CloudWatc hAgentServerPolicy kebijakan harus dilampirkan ke peran instance.	peran tersebut dilampirkan ke contoh, dibutuhkan waktu bagi SSM untuk memasukka n contoh dalam DescribeI nstanceInformation hasil. Sampai SSM mencakup instans dalam hasilnya, kesalahan NO_SSM_IN STANCE_ROLE tetap akan ditampilkan untuk aplikasi tersebut.
Komponen-komponen baru mungkin memerlukan konfigurasi.	Wawasan Aplikasi mendeteksi adanya komponen-komponen baru dalam Grup Sumber Daya aplikasi. Untuk mengatasi hal ini, Anda harus mengonfigurasi komponen baru tersebut.	T/A

Log dan metrik yang didukung oleh Amazon CloudWatch Application Insights

Daftar berikut menunjukkan log dan metrik yang didukung untuk Amazon CloudWatch Application Insights.

CloudWatch Application Insights mendukung log berikut:

- Log Microsoft Internet Information Services (IIS)
- Log kesalahan untuk Server SQL pada EC2
- Log aplikasi .NET kustom, seperti Log4Net
- Log Peristiwa Windows, termasuk log Windows (Sistem, Aplikasi, dan Keamanan) serta log Aplikasi dan Layanan
- CloudWatch Log Amazon untuk AWS Lambda

- Log kesalahan dan log lambat untuk RDS MySQL, Aurora MySQL, dan MySQL di EC2
- Log Postgresql untuk PostgreSQL RDS dan PostgreSQL di EC2
- CloudWatch Log Amazon untuk AWS Step Functions
- Log eksekusi dan log akses (JSON, CSV, dan XML, tetapi bukan CLF) untuk tahapan-tahapan API REST API Gateway
- Log pengekspor Prometheus JMX (EMF)
- Log peringatan dan log pendengar untuk Oracle di Amazon RDS dan Oracle di Amazon EC2
- Container log routing dari Amazon ECS container untuk CloudWatch menggunakan `awslogslog` driver.
- Container log routing dari kontainer Amazon ECS untuk CloudWatch menggunakan router log FireLens kontainer.
- Container mencatat routing dari Amazon EKS atau Kubernetes yang berjalan di Amazon EC2 untuk CloudWatch menggunakan prosesor log Fluent Bit atau Fluentd dengan Container Insights.
- Log jejak dan kesalahan SAP HANA
- Log Pacemaker HA
- Log server SAP ASE
- Log server cadangan SAP ASE
- Log server Replikasi SAP ASE
- Log agen SAP ASE RMA
- Log Manajer Kesalahan SAP ASE
- Log jejak NetWeaver pengembang SAP
- Metrik proses untuk proses Windows menggunakan plugin proctstat untuk agen CloudWatch
- Log kueri DNS publik untuk zona yang di-hosting
- Amazon Route 53 Resolver Log kueri DNS

CloudWatch Application Insights mendukung kelas log berikut:

- Standar - Amazon CloudWatch Application Insights mengharuskan grup log dikonfigurasi dengan kelas CloudWatch log Standar Log untuk mengaktifkan pemantauan.

CloudWatch Application Insights mendukung metrik untuk komponen aplikasi berikut:

- Amazon Elastic Compute Cloud (EC2)

  - CloudWatch metrik bawaan
  - CloudWatch metrik agen (server Windows)
  - CloudWatch metrik proses agen (server Windows)
  - CloudWatch metrik agen (server Linux)
- Elastic Block Store (EBS)
- Amazon Elastic File System (Amazon EFS)
- Elastic Load Balancer (ELB)
- ELB Aplikasi
- Grup Amazon EC2 Auto Scaling
- Amazon Simple Queue Server (SQS)
- Amazon Relational Database Service (RDS)
  - Instans-instans Basis Data RDS
  - Klaster-klaster Basis Data RDS
- AWS Lambda fungsi
- Tabel Amazon DynamoDB
- Bucket Amazon S3
- AWS Step Functions
  - Execution-level
  - Aktifitas
  - Fungsi Lambda
  - Integrasi layanan
  - API Fungi Langkah
- Tahapan-tahapan API REST API Gateway
- SAP HANA
- SAP ASE
- SAP ASE Ketersediaan Tinggi di Amazon EC2
- GETAH NetWeaver
- Klaster HA
- Java
- Amazon Elastic Container Service (Amazon ECS)

- CloudWatch metrik bawaan
- Metrik-metrik Wawasan Kontainer
- Metrik-metrik Prometheus Wawasan Kontainer
- Kubernetes di AWS
  - Metrik-metrik Wawasan Kontainer
  - Metrik-metrik Prometheus Wawasan Kontainer
- Amazon FSx
- Amazon VPC
- NAT Gateway Amazon VPC
- Pemeriksaan kondisi kesehatan Amazon Route 53
- Zona yang di-hosting Amazon Route 53
- Amazon Route 53 Resolver titik akhir
- AWS Network Firewall kelompok aturan
- AWS Network Firewall asosiasi kelompok aturan
- Metrik-metrik dengan persyaratan titik data
  - AWS/ApplicationELB
  - AWS/ AutoScaling
  - AWS/EC2
  - Elastic Block Store (EBS)
  - AWS/ELB
  - AWS/RDS
  - AWS/Lambda
  - AWS/SQS
  - AWS/CWAgent
  - AWS/DynamoDB
  - AWS/S3
  - AWS/States
  - AWS/ ApiGateway
  - AWS/SNS
- Metrik-metrik yang direkomendasikan

- Metrik-metrik Penghitung Performa

Amazon Elastic Compute Cloud (EC2)

CloudWatch Application Insights mendukung metrik berikut:

Metrik

- CloudWatch metrik bawaan
- CloudWatch metrik agen (server Windows)
- CloudWatch metrik proses agen (server Windows)
- CloudWatch metrik agen (server Linux)

CloudWatch metrik bawaan

CPU CreditBalance

CPU CreditUsage

CPU SurplusCreditBalance

CPU SurplusCreditsCharged

CPUUtilization

DiskReadBytes

DiskReadOps

DiskWriteBytes

DiskWriteOps

EBS% ByteBalance

EBSIOBalance%

EBS ReadBytes

EBS ReadOps

EBS WriteBytes

EBS WriteOps

NetworkIn

NetworkOut

NetworkPacketsIn

NetworkPacketsOut

StatusCheckFailed

StatusCheckFailed_Contoh

StatusCheckFailed_Sistem

CloudWatch metrik agen (server Windows)

Pengecualian CLR .NET # dari Exceps Thrown

Pengecualian .NET CLR # dari Exceps Thrown/Detik

Pengecualian .NET CLR # dari Penyaring/detik

Pengecualian .NET CLR # dari Finallys/detik

Pengecualian .NET CLR untuk Menelusuri Kedalaman/detik

Interop .NET CLR # dari CCW

Interop .NET CLR # dari Stub

Interop .NET CLR # ekspor TLB/detik

Interop .NET CLR # impor TLB/detik

Interop .NET CLR # marshaling

.NET CLR Jit % Waktu di Jit

Kegagalan Jit Standar Jit .NET CLR

Persentase Waktu Pemuatan Pemuatan .NET CLR

Tingkat Pemuatan Kegagalan Beban CLR .NET

.NET CLR Tingkat LocksAndThreads Perbalahan/detik

.NET CLR Panjang LocksAndThreads Antrian/detik

Memori CLR .NET # Total Byte Berkomitmen

Memori CLR .NET % Waktu dalam GC

.NET CLR Networking 4.0.0.0 Waktu HttpWebRequest Antrian Rata-rata

.NET CLR Networking 4.0.0.0 HttpWebRequests Dibatarkan/sec

.NET CLR Networking 4.0.0.0 HttpWebRequests Gagal/detik

.NET CLR Networking 4.0.0.0 Antrean/detik HttpWebRequests

Kegagalan Ping Proses Pekerja Total APP_POOL_WAS

Memulai Ulang Aplikasi ASP.NET

Persentase Waktu Prosesor Terkelola Aplikasi ASP.NET (perkiraan)

Total Kesalahan Aplikasi ASP.NET/Detik

Kesalahan-kesalahan Aplikasi ASP.NET yang Tidak Ditangani Selama Eksekusi/detik

Permintaan Aplikasi ASP.NET dalam Antrean Aplikasi

Permintaan Aplikasi ASP.NET/Detik

Waktu Tunggu Permintaan ASP.NET

Permintaan ASP.NET yang Diantrekan

Antrian Permintaan Layanan HTTP CurrentQueueSize

LogicalDisk % Ruang Bebas

Memori % Byte Berkomitmen yang Digunakan

Mbyte Memori yang Tersedia

Halaman Memori/detik

Total Byte Antarmuka Jaringan/detik

File Halaman % Penggunaan

PhysicalDisk % Waktu Disk

PhysicalDisk Rata-rata. Panjang Antrean Cakram

PhysicalDisk Rata-rata. Detik Cakram/Baca

PhysicalDisk Rata-rata. Detik Cakram/Tulis

PhysicalDisk Byte Baca Disk/detik

PhysicalDisk Pembacaan Disk/detik

PhysicalDisk Byte Tulis Disk/detik

PhysicalDisk Disk Menulis/detik

Pemroses % Waktu Menganggur

Persentase Waktu Interupsi Prosesor

Persentase Pemroses Processor Time

Persentase Waktu Pengguna Prosesor

SQLServer:Metode Akses Catatan yang Diteruskan/detik

SQLServer:Metode Akses Pemindaian Penuh/detik

SQLServer:Metode Akses Pembagian Halaman/detik

SQLServer:Rasio hit cache Buffer Manajer Buffer

SQLServer:Harapan hidup halaman Manajer Buffer

SQLServer:Proses Statistik Umum yang diblokir

SQLServer:Koneksi Pengguna Statistik Umum

SQLServer:Waktu Tunggu Latch Rata-rata (ms)

SQLServer:Waktu Tunggu Rata-rata Kunci (ms)

SQLServer:Kunci Batas Waktu Kunci/detik

SQLServer:Kunci Kunci Menunggu/detik

SQLServer:Jumlah Kunci Penguncian/detik

SQLServer:Pemberian Memori Manajer Memori Tertunda

SQLServer:Permintaan Batch Statistik SQL/detik

SQLServer:Statistik SQL Kompilasi SQL/detik

SQLServer:Statistik SQL Re-Kompilasi SQL/detik

Panjang Antrean Prosesor Sistem

Koneksi TCPv4 Ditetapkan

Koneksi TCPv6 Ditetapkan

W3SVC_W3WP File Cache Terbilas

W3SVC_W3WP File Cache Hilang

W3SVC_W3WP Permintaan/Detik

W3SVC_W3WP URI Cache Terbilas

W3SVC_W3WP URI Cache Hilang

Byte Layanan Web yang Diterima/Detik

Byte Layanan Web yang Dikirim/Detik

Upaya Koneksi Layanan Web/detik

Koneksi Saat Ini Layanan Web

Permintaan Get Layanan Web/detik

Permintaan Post Layanan Web/detik

Byte Diterima/detik

Panjang Antrean Pesan Normal/detik

Panjang Antrean Pesan Mendesak/detik

Hubungkan Kembali Hitungan

Panjang Antrean Pesan Tidak Diketahui/detik

Pesan Terutang

Pesan Dikirim/detik

Pesan Pembaruan Basis Data/detik

Pembaruan Pesan/detik

Pembilasan/detik

Pos Pemeriksaan Kripto Disimpan/detik

Pos Pemeriksaan Kripto Dipulihkan/detik

Pos Pemeriksaan Registri Dipulihkan/detik

Pos Pemeriksaan Registri Disimpan/detik

Panggilan API Klaster/detik

Panggilan API Sumber Daya/detik

Penanganan Klaster/detik

Penanganan Sumber Daya/detik

CloudWatch metrik proses agen (server Windows)

Metrik proses dikumpulkan menggunakan plugin CloudWatch agent procstat. Hanya instans Amazon EC2 yang menjalankan beban kerja Windows yang mendukung metrik proses.

procstat cpu_time_system

procstat cpu_time_user

procstat cpu_usage

procstat memory_rss

procstat memory_vms

procstat read_bytes

procstat write_bytes

.procstat read_count

procstat write_count

CloudWatch metrik agen (server Linux)

cpu_time_active

cpu_time_guest

cpu_time_guest_nice

cpu_time_idle

cpu_time_iowait

cpu_time_irq

cpu_time_nice

cpu_time_softirq

cpu_time_steal

cpu_time_system

cpu_time_user

cpu_usage_active

cpu_usage_guest

cpu_usage_guest_nice

cpu_usage_idle

cpu_usage_iowait

cpu_usage_irq

cpu_usage_nice

cpu_usage_softirq

cpu_usage_steal

cpu_usage_system

cpu_usage_user

disk_free

disk_inodes_free

disk_inodes_used

disk_used

disk_used_percent

diskio_io_time

diskio_iops_in_progress

diskio_read_bytes

diskio_read_time

diskio_reads

diskio_write_bytes

diskio_write_time

diskio_writes

mem_active

mem_available

mem_available_percent

mem_buffered

mem_cached

mem_free

mem_inactive

mem_used

mem_used_percent

net_bytes_recv

net_bytes_sent

net_drop_in

net_drop_out

net_err_in

net_err_out

net_packets_recv

net_packets_sent

netstat_tcp_close

netstat_tcp_close_wait

netstat_tcp_closing

netstat_tcp_established

netstat_tcp_fin_wait1

netstat_tcp_fin_wait2

netstat_tcp_last_ack

netstat_tcp_listen

netstat_tcp_none

netstat_tcp_syn_recv

netstat_tcp_syn_sent

netstat_tcp_time_wait

netstat_udp_socket

processes_blocked

processes_dead

processes_idle

processes_paging

processes_running

processes_sleeping

processes_stopped

processes_total

processes_total_threads

processes_wait

processes_zombies

swap_free

swap_used

swap_used_percent

Elastic Block Store (EBS)

CloudWatch Application Insights mendukung metrik berikut:

VolumeReadBytes

VolumeWriteBytes

VolumeReadOps

VolumeWriteOps

VolumeTotalReadTime

VolumeTotalWriteTime

VolumeIdleTime

VolumeQueueLength

VolumeThroughputPercentage

VolumeConsumedReadWriteOps

BurstBalance

Amazon Elastic File System (Amazon EFS)

CloudWatch Application Insights mendukung metrik berikut:

BurstCreditBalance

PercentIOLimit

PermittedThroughput

MeteredIOBytes

TotalIOBytes

DataWriteIObytes

DataReadIObytes

MetadataIOBytes

ClientConnections

TimeSinceLastSync

StorageBytes

Throughput

PercentageOfPermittedThroughputUtilization

ThroughputIOPS

PercentThroughputDataReadIObyte

PercentThroughputDataWriteIObytes

PercentageOfIOPS IOBYTES DataRead

PercentageOfIOPS IOBYTES DataWrite

AverageDataReadIO BytesSize

AverageDataWriteIO BytesSize

Elastic Load Balancer (ELB)

CloudWatch Application Insights mendukung metrik berikut:

DiperkirakanAlb ActiveConnectionCount

EstimatedALBConsumedLCUs

DiperkirakanAlb NewConnectionCount

EstimatedProcessedBytes

HTTPCode_Backend_4XX

HTTPCode_Backend_5XX

HealthyHostCount

RequestCount

UnHealthyHostCount

ELB Aplikasi

CloudWatch Application Insights mendukung metrik berikut:

DiperkirakanAlb ActiveConnectionCount

EstimatedALBConsumedLCUs

DiperkirakanAlb NewConnectionCount

EstimatedProcessedBytes

HTTPCode_Backend_4XX

HTTPCode_Backend_5XX

HealthyHostCount

Latensi

RequestCount

SurgeQueueLength

UnHealthyHostCount

Grup Amazon EC2 Auto Scaling

CloudWatch Application Insights mendukung metrik berikut:

CPU CreditBalance

CPU CreditUsage

CPU SurplusCreditBalance

CPU SurplusCreditsCharged

CPUUtilization

DiskReadBytes

DiskReadOps

DiskWriteBytes

DiskWriteOps

EBS% ByteBalance

EBSIOBalance%

EBS ReadBytes

EBS ReadOps

EBS WriteBytes

EBS WriteOps

NetworkIn

NetworkOut

NetworkPacketsIn

NetworkPacketsOut

StatusCheckFailed

StatusCheckFailed_Contoh

StatusCheckFailed_Sistem

Amazon Simple Queue Server (SQS)

CloudWatch Application Insights mendukung metrik berikut:

ApproximateAgeOfOldestMessage

ApproximateNumberOfMessagesDelayed

ApproximateNumberOfMessagesNotVisible

ApproximateNumberOfMessagesVisible

NumberOfEmptyReceives

NumberOfMessagesDeleted

NumberOfMessagesReceived

NumberOfMessagesSent

Amazon Relational Database Service (RDS)

CloudWatch Application Insights mendukung metrik berikut:

Metrik

- Instans-instans Basis Data RDS

- Klaster-klaster Basis Data RDS

Instans-instans Basis Data RDS

BurstBalance

CPU CreditBalance

CPUUtilization

DatabaseConnections

DiskQueueDepth

GagalSQL ServerAgentJobsCount

FreeStorageSpace

FreeableMemory

NetworkReceiveThroughput

NetworkTransmitThroughput

ReadIOPS

ReadLatency

ReadThroughput

WriteIOPS

WriteLatency

WriteThroughput

Klaster-klaster Basis Data RDS

ActiveTransactions

AuroraBinlogReplicaLag

AuroraReplicaLag

BackupRetentionPeriodStorageUsed

BinLogDiskUsage

BlockedTransactions

BufferCacheHitRatio

CPUUtilization

CommitLatency

CommitThroughput

DDLLatency

DDLThroughput

DMLLatency

DMLThroughput

DatabaseConnections

Deadlocks

DeleteLatency

DeleteThroughput

EngineUptime

FreeLocalStorage

FreeableMemory

InsertLatency

InsertThroughput

LoginFailures

NetworkReceiveThroughput

NetworkThroughput

NetworkTransmitThroughput

Mengajukan Kueri

ResultSetCacheHitRatio

SelectLatency

SelectThroughput

SnapshotStorageUsed

TotalBackupStorageBilled

UpdateLatency

UpdateThroughput

VolumeBytesUsed

VolumeReadIOP

VolumeWriteIOP

AWS Lambda fungsi

CloudWatch Application Insights mendukung metrik berikut:

Kesalahan

DeadLetterErrors

Durasi

Trotel

IteratorAge

ProvisionedConcurrencySpilloverInvocations

Tabel Amazon DynamoDB

CloudWatch Application Insights mendukung metrik berikut:

SystemErrors

UserErrors

ConsumedReadCapacityUnits

ConsumedWriteCapacityUnits

ReadThrottleEvents

WriteThrottleEvents

TimeToLiveDeletedItemCount

ConditionalCheckFailedRequests

TransactionConflict

ReturnedRecordsCount

PendingReplicationCount

ReplicationLatency

Bucket Amazon S3

CloudWatch Application Insights mendukung metrik berikut:

ReplicationLatency

BytesPendingReplication

OperationsPendingReplication

4xxKesalahan

5xxKesalahan

AllRequests

GetRequests

PutRequests

DeleteRequests

HeadRequests

PostRequests

SelectRequests

ListRequests

SelectScannedBytes

SelectReturnedBytes

FirstByteLatency

TotalRequestLatency

BytesDownloaded

BytesUploaded

AWS Step Functions

CloudWatch Application Insights mendukung metrik berikut:

Metrik

- Execution-level
- Aktifitas
- Fungsi Lambda
- Integrasi layanan
- API Fungi Langkah

Execution-level

ExecutionTime

ExecutionThrottled

ExecutionsFailed

ExecutionsTimedOut

ExecutionsAborted

ExecutionsSucceeded

ExecutionsStarted

Aktifitas

ActivityRunTime

ActivityScheduleTime

ActivityTime

ActivitiesFailed

ActivitiesHeartbeatTimedOut

ActivitiesTimedOut

ActivitiesScheduled

ActivitiesSucceeded

ActivitiesStarted

Fungsi Lambda

LambdaFunctionRunTime

LambdaFunctionScheduleTime

LambdaFunctionTime

LambdaFunctionsFailed

LambdaFunctionsTimedOut

LambdaFunctionsScheduled

LambdaFunctionsSucceeded

LambdaFunctionsStarted

Integrasi layanan

ServiceIntegrationRunTime

ServiceIntegrationScheduleTime

ServiceIntegrationTime

ServiceIntegrationsFailed

ServiceIntegrationsTimedOut

ServiceIntegrationsScheduled

ServiceIntegrationsSucceeded

ServiceIntegrationsStarted

API Fungi Langkah

ThrottledEvents

ProvisionedBucketSize

ProvisionedRefillRate

ConsumedCapacity

Tahapan-tahapan API REST API Gateway

CloudWatch Application Insights mendukung metrik berikut:

4XXKesalahan

5XXKesalahan

IntegrationLatency

Latensi

CacheHitCount

CacheMissCount

SAP HANA

> **Note**
>
> CloudWatch Application Insights hanya mendukung lingkungan SID HANA tunggal. Jika ada beberapa HANA SID yang dilampirkan, maka pemantauan akan diatur hanya untuk SID pertama yang terdeteksi.

CloudWatch Application Insights mendukung metrik berikut:

hanadb_every_service_started_status

hanadb_daemon_service_started_status

hanadb_preprocessor_service_started_status

hanadb_webdispatcher_service_started_status

hanadb_compileserver_service_started_status

hanadb_nameserver_service_started_status

hanadb_server_startup_time_variations_seconds

hanadb_level_5_alerts_count

hanadb_level_4_alerts_count

hanadb_out_of_memory_events_count

hanadb_max_trigger_read_ratio_percent

hanadb_max_trigger_write_ratio_percent

hanadb_log_switch_wait_ratio_percent

hanadb_log_switch_race_ratio_percent

hanadb_time_since_last_savepoint_seconds

hanadb_disk_usage_highlevel_percent

hanadb_max_converter_page_number_count

hanadb_long_running_savepoints_count

hanadb_failed_io_reads_count

hanadb_failed_io_writes_count

hanadb_disk_data_unused_percent

hanadb_current_allocation_limit_used_percent

hanadb_table_allocation_limit_used_percent

hanadb_host_total_physical_memory_mb

hanadb_host_physical_memory_used_mb

hanadb_host_physical_memory_free_mb

hanadb_swap_memory_free_mb

hanadb_swap_memory_used_mb

hanadb_host_allocation_limit_mb

hanadb_host_total_memory_used_mb

hanadb_host_total_peak_memory_used_mb

hanadb_host_total_allocation_limit_mb

hanadb_host_code_size_mb

hanadb_host_shared_memory_allocation_mb

hanadb_cpu_usage_percent

hanadb_cpu_user_percent

hanadb_cpu_system_percent

hanadb_cpu_waitio_percent

hanadb_cpu_busy_percent

hanadb_cpu_idle_percent

hanadb_long_delta_merge_count

hanadb_unsuccessful_delta_merge_count

hanadb_successful_delta_merge_count

hanadb_row_store_allocated_size_mb

hanadb_row_store_free_size_mb

hanadb_row_store_used_size_mb

hanadb_temporary_tables_count

hanadb_large_non_compressed_tables_count

hanadb_total_non_compressed_tables_count

hanadb_longest_running_job_seconds

hanadb_average_commit_time_milliseconds

hanadb_suspended_sql_statements_count

hanadb_plan_cache_hit_ratio_percent

hanadb_plan_cache_lookup_count

hanadb_plan_cache_hit_count

hanadb_plan_cache_total_execution_microseconds

hanadb_plan_cache_cursor_duration_microseconds

hanadb_plan_cache_preparation_microseconds

hanadb_plan_cache_evicted_count

hanadb_plan_cache_evicted_microseconds

hanadb_plan_cache_evicted_preparation_count

hanadb_plan_cache_evicted_execution_count

hanadb_plan_cache_evicted_preparation_microseconds

hanadb_plan_cache_evicted_cursor_duration_microseconds

hanadb_plan_cache_evicted_total_execution_microseconds

hanadb_plan_cache_evicted_plan_size_mb

hanadb_plan_cache_count

hanadb_plan_cache_preparation_count

hanadb_plan_cache_execution_count

hanadb_network_collision_rate

hanadb_network_receive_rate

hanadb_network_transmit_rate

hanadb_network_packet_receive_rate

hanadb_network_packet_transmit_rate

hanadb_network_transmit_error_rate

hanadb_network_receive_error_rate

hanadb_time_until_license_expires_days

hanadb_is_license_valid_status

hanadb_local_running_connections_count

hanadb_local_idle_connections_count

hanadb_remote_running_connections_count

hanadb_remote_idle_connections_count

hanadb_last_full_data_backup_age_days

hanadb_last_data_backup_age_days

hanadb_last_log_backup_age_hours

hanadb_failed_data_backup_past_7_days_count

hanadb_failed_log_backup_past_7_days_count

hanadb_oldest_backup_in_catalog_age_days

hanadb_backup_catalog_size_mb

hanadb_hsr_replication_status

hanadb_hsr_log_shipping_delay_seconds

hanadb_hsr_secondary_failover_count

hanadb_hsr_secondary_reconnect_count

hanadb_hsr_async_buffer_used_mb

hanadb_hsr_secondary_active_status

hanadb_handle_count

hanadb_ping_time_milliseconds

hanadb_connection_count

hanadb_internal_connection_count

hanadb_external_connection_count

hanadb_idle_connection_count

hanadb_transaction_count

hanadb_internal_transaction_count

hanadb_external_transaction_count

hanadb_user_transaction_count

hanadb_blocked_transaction_count

hanadb_statement_count

hanadb_active_commit_id_range_count

hanadb_mvcc_version_count

hanadb_pending_session_count

hanadb_record_lock_count

hanadb_read_count

hanadb_write_count

hanadb_merge_count

hanadb_unload_count

hanadb_active_thread_count

hanadb_waiting_thread_count

hanadb_total_thread_count

hanadb_active_sql_executor_count

hanadb_waiting_sql_executor_count

hanadb_total_sql_executor_count

hanadb_data_write_size_mb

hanadb_data_write_time_milliseconds

hanadb_log_write_size_mb

hanadb_log_write_time_milliseconds

hanadb_data_read_size_mb

hanadb_data_read_time_milliseconds

hanadb_log_read_size_mb

hanadb_log_read_time_milliseconds

hanadb_data_backup_write_size_mb

hanadb_data_backup_write_time_milliseconds

hanadb_log_backup_write_size_mb

hanadb_log_backup_write_time_milliseconds

hanadb_mutex_collision_count

hanadb_read_write_lock_collision_count

hanadb_admission_control_admit_count

hanadb_admission_control_reject_count

hanadb_admission_control_queue_size_mb

hanadb_admission_control_wait_time_milliseconds

SAP ASE

CloudWatch Application Insights mendukung metrik berikut:

asedb_database_availability

asedb_trunc_log_on_chkpt_enabled

asedb_last_db_backup_age_in_days

asedb_last_transaction_log_backup_age_in_hours

asedb_suspected_database

asedb_db_space_usage_percent

asedb_db_log_space_usage_percent

asedb_locked_login

asedb_has_mixed_log_and_data

asedb_runtime_for_open_transactions

asedb_data_cache_hit_ratio

asedb_data_cache_usage

asedb_sql_cache_hit_ratio

asedb_cache_usage

asedb_run_queue_length

asedb_number_of_rollbacks

asedb_number_of_commits

asedb_number_of_transactions

asedb_outstanding_disk_io

asedb_percent_io_busy

asedb_percent_system_busy

asedb_percent_locks_active

asedb_scheduled_jobs_failed_percent

asedb_user_connections_percent

asedb_query_logical_reads

asedb_query_physical_reads

asedb_query_cpu_time

asedb_query_memory_usage

SAP ASE Ketersediaan Tinggi di Amazon EC2

CloudWatch Application Insights mendukung metrik berikut:

asedb_ha_replication_state

asedb_ha_replication_mode

asedb_ha_replication_latency_in_minutes

GETAH NetWeaver

CloudWatch Application Insights mendukung metrik berikut:

Metrik	Deskripsi
sap_alerts_ ResponseTime	Peringatan waktu respons SAP dari CCMS (RZ20) > R3Layanan> Dialog>. ResponseTime
sap_alerts_ ResponseTimeDialog	Peringatan dialog waktu respons SAP dari CCMS (RZ20) > R3Layanan> Dialog>. ResponseTimeDialog
sap_alerts_ RFC ResponseTimeDialog	Peringatan waktu respons SAP dari CCMS (RZ20) > R3Services> Dialog> RFC. ResponseTimeDialog
SAP_Alerts_DB RequestTime	Peringatan waktu respons SAP dari CCMS (RZ20) > R3Services> Dialog> DB. RequestTime
sap_alerts_ FrontendResponseTime	Peringatan waktu respons SAP dari CCMS (RZ20) > R3Services > Dialog>. FrontEndResponseTime
sap_alerts_Database	Sistem SAP telah mencatat log kesalahan terkait basis data. Peringatan dari SM21 atau CCMS (RZ20)>R3Syslog>Database.
sap_alerts_ QueueTime	Peringatan waktu antrian SAP dari CCMS (RZ20) > R3Layanan> Dialog>. QueueTime
sap_alerts_ AbortedJobs	Pekerjaan latar belakang yang gagal dalam sistem SAP. Peringatan dari (RZ20) > R3Services > Latar Belakang>. AbortedJobs
sap_alerts_ BasisSystem	Sistem SAP mencatat log kesalahan tingkat sistem. Peringatan dari SM21 atau CCMS (RZ20) > R3Syslog>. BasisSystem
sap_alerts_Security	Sistem SAP mencatat log pesan terkait keamanan. Peringatan dari SM21 atau CCMS (RZ20)>R3Syslog>Security.

Metrik	Deskripsi
sap_alerts_System	Sistem SAP mencatat log keamanan atau pesan terkait audit. Peringatan dari SM21 atau CCMS (RZ20)>Security>System.
sap_alerts_ LongRunners	Ada program-program yang berjalan lama di sistem SAP Anda. Peringatan dari CCMS (RZ20) > R3Services > Dialog>. LongRunners
sap_alerts_ SqlError	Ada log kesalahan lapisan klien basis data SAP. Peringatan dari CCMS (RZ20) > >DatabaseClient. AbapSql SqlError
sap_alerts_State	Peringatan status dari CCMS (RZ20)>OS Collector>State.
sap_alerts_Shortdumps	Peringatan shortdumps dari ST22 dan CCMS (RZ20)>R3Abap>Shortdumps.
sap_alerts_Availability	Peringatan ketersediaan untuk instance server aplikasi SAP dari SM21, SM50, SM51, SM66, dan CCMS (RZ20) > > Ketersediaan. InstanceA sTask
sap_dispatcher_queue_high	Fungsi `GetQueueStatistic` Layanan Web SapControl menyediakan jumlah antrean dispatcher yang tinggi.
sap_dispatcher_queue_max	Fungsi `GetQueueStatistic` Layanan Web SapControl menyediakan jumlah antrean dispatcher maksimum.
sap_dispatcher_queue_now	Fungsi `GetQueueStatistic` Layanan Web SapControl menyediakan antrean dispatcher yang sekarang dihitung.

Metrik	Deskripsi
sap_dispatcher_queue_reads	Fungsi `GetQueueStatistic` Layanan Web SapControl menyediakan jumlah pembacaan antrean dispatcher.
sap_dispatcher_queue_writes	Fungsi `GetQueueStatistic` Layanan Web SapControl menyediakan jumlah penulisan antrean dispatcher.
sap_enqueue_server_arguments_high	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan argumen penambahan data pada antrean tinggi.
sap_enqueue_server_arguments_max	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan argumen penambahan data pada antrean maksimal.
sap_enqueue_server_arguments_now	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan argumen penambahan data pada antrean sekarang.
sap_enqueue_server_arguments_state	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan argumen penambahan data pada antrean status.
sap_enqueue_server_backup_requests	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan permintaan pencadangan penambahan data pada antrean.
sap_enqueue_server_cleanup_requests	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan permintaan pembersihan penambahan data pada antrean.
sap_enqueue_server_dequeue_all_requests	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan penghapusan data semua permintaan.

Metrik	Deskripsi
sap_enqueue_server_dequeue_errors	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan penghapusan data kesalahan.
sap_enqueue_server_dequeue_requests	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan penghapusan data permintaan.
sap_enqueue_server_enqueue_errors	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan penambahan data pada antrean untuk kesalahan.
sap_enqueue_server_enqueue_rejects	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan penambahan data pada antrean untuk penolakan.
sap_enqueue_server_enqueue_requests	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan permintaa n penambahan data pada antrean untuk permintaan.
sap_enqueue_server_lock_time	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan penambahan data pada antrean untuk waktu mengunci.
sap_enqueue_server_lock_wait_time	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan penambahan data pada antrean untuk waktu tunggu mengunci.
sap_enqueue_server_locks_high	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan penambahan data pada antrean untuk mengunci tinggi.
sap_enqueue_server_locks_max	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan penambahan data pada antrean untuk mengunci maksimal.

Metrik	Deskripsi
sap_enqueue_server_locks_now	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan penambahan data pada antrean untuk mengunci sekarang.
sap_enqueue_server_locks_state	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan penambahan data pada antrean untuk mengunci status.
sap_enqueue_server_owner_high	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan penambahan data pada antrean untuk pemilik tinggi.
sap_enqueue_server_owner_max	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan penambahan data pada antrean untuk pemilik maksimal.
sap_enqueue_server_owner_now	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan penambahan data pada antrean untuk pemilik sekarang.
sap_enqueue_server_owner_state	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan penambahan data pada antrean untuk pemilik status.
sap_enqueue_server_replication_state	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan penambahan data pada antrean untuk keadaan status replikasi.
sap_enqueue_server_reporting_requests	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan permintaan pelaporan.
sap_enqueue_server_server_time	Fungsi `EnqGetStatistic` Layanan Web SAPControl menyediakan penambahan data pada antrean untuk waktu server.

Metrik	Deskripsi
sap_HA_check_failover_config_state	Fungsi `HACheckFailoverConfig` Layanan Web SAPControl menyediakan status SAP High Availability.
sap_HA_get_failover_config_HAActive	Fungsi `HAGetFailoverConfig` Layanan Web SAPControl menyediakan konfigurasi dan status Klaster Ketersediaan Tinggi SAP.
sap_start_service_processes	Fungsi `GetProcessList` Layanan Web SAPControl menyediakan status proses disp +work, IGS, gwrd, icman, server pesan, dan server penambahan data pada antrean.

Klaster HA

CloudWatch Application Insights mendukung metrik berikut:

ha_cluster_pacemaker_stonith_enabled

ha_cluster_corosync_quorate

hanadb_webdispatcher_service_started_status

ha_cluster_pacemaker_nodes

ha_cluster_corosync_ring_errors

ha_cluster_pacemaker_fail_count

Java

CloudWatch Application Insights mendukung metrik berikut:

java_lang_memory_heapmemoryusage_used

java_lang_memory_heapmemoryusage_committed

java_lang_operatingsystem_openfiledescriptorcount

java_lang_operatingsystem_maxfiledescriptorcount

java_lang_operatingsystem_freephysicalmemorysize

java_lang_operatingsystem_freeswapspacesize

java_lang_threading_threadcount

java_lang_threading_daemonthreadcount

java_lang_classloading_loadedclasscount

java_lang_garbagecollector_collectiontime_copy

java_lang_garbagecollector_collectiontime_ps_scavenge

java_lang_garbagecollector_collectiontime_parnew

java_lang_garbagecollector_collectiontime_marksweepcompact

java_lang_garbagecollector_collectiontime_ps_marksweep

java_lang_garbagecollector_collectiontime_concurrentmarksweep

java_lang_garbagecollector_collectiontime_g1_young_generation

java_lang_garbagecollector_collectiontime_g1_old_generation

java_lang_garbagecollector_collectiontime_g1_mixed_generation

java_lang_operatingsystem_committedvirtualmemorysize

Amazon Elastic Container Service (Amazon ECS)

CloudWatch Application Insights mendukung metrik berikut:

Metrik

- CloudWatch metrik bawaan
- Metrik-metrik Wawasan Kontainer
- Metrik-metrik Prometheus Wawasan Kontainer

CloudWatch metrik bawaan

CPUReservation

CPUUtilization

MemoryReservation

MemoryUtilization

GPUReservation

Metrik-metrik Wawasan Kontainer

ContainerInstanceCount

CpuUtilized

CpuReserved

DeploymentCount

DesiredTaskCount

MemoryUtilized

MemoryReserved

NetworkRxBytes

NetworkTxBytes

PendingTaskCount

RunningTaskCount

ServiceCount

StorageReadBytes

StorageWriteBytes

TaskCount

TaskSetCount

instance_cpu_limit

instance_cpu_reserved_capacity

instance_cpu_usage_total

instance_cpu_utilization

instance_filesystem_utilization

instance_memory_limit

instance_memory_reserved_capacity

instance_memory_utilization

instance_memory_working_set

instance_network_total_bytes

instance_number_of_running_tasks

Metrik-metrik Prometheus Wawasan Kontainer

Metrik-metrik Java JMX

java_lang_memory_heapmemoryusage_used

java_lang_memory_heapmemoryusage_committed

java_lang_operatingsystem_openfiledescriptorcount

java_lang_operatingsystem_maxfiledescriptorcount

java_lang_operatingsystem_freephysicalmemorysize

java_lang_operatingsystem_freeswapspacesize

java_lang_threading_threadcount

java_lang_classloading_loadedclasscount

java_lang_threading_daemonthreadcount

java_lang_garbagecollector_collectiontime_copy

java_lang_garbagecollector_collectiontime_ps_scavenge

java_lang_garbagecollector_collectiontime_parnew

java_lang_garbagecollector_collectiontime_marksweepcompact

java_lang_garbagecollector_collectiontime_ps_marksweep

java_lang_garbagecollector_collectiontime_concurrentmarksweep

java_lang_garbagecollector_collectiontime_g1_young_generation

java_lang_garbagecollector_collectiontime_g1_old_generation

java_lang_garbagecollector_collectiontime_g1_mixed_generation

java_lang_operatingsystem_committedvirtualmemorysize

Kubernetes di AWS

CloudWatch Application Insights mendukung metrik berikut:

Metrik

- Metrik-metrik Wawasan Kontainer
- Metrik-metrik Prometheus Wawasan Kontainer

Metrik-metrik Wawasan Kontainer

cluster_failed_node_count

cluster_node_count

namespace_number_of_running_pods

node_cpu_limit

node_cpu_reserved_capacity

node_cpu_usage_total

node_cpu_utilization

node_filesystem_utilization

node_memory_limit

node_memory_reserved_capacity

node_memory_utilization

node_memory_working_set

node_network_total_bytes

node_number_of_running_containers

node_number_of_running_pods

pod_cpu_reserved_capacity

pod_cpu_utilization

pod_cpu_utilization_over_pod_limit

pod_memory_reserved_capacity

pod_memory_utilization

pod_memory_utilization_over_pod_limit

pod_network_rx_bytes

pod_network_tx_bytes

service_number_of_running_pods

Metrik-metrik Prometheus Wawasan Kontainer

Metrik-metrik Java JMX

java_lang_memory_heapmemoryusage_used

java_lang_memory_heapmemoryusage_committed

java_lang_operatingsystem_openfiledescriptorcount

java_lang_operatingsystem_maxfiledescriptorcount

java_lang_operatingsystem_freephysicalmemorysize

java_lang_operatingsystem_freeswapspacesize

java_lang_threading_threadcount

java_lang_classloading_loadedclasscount

java_lang_threading_daemonthreadcount

java_lang_garbagecollector_collectiontime_copy

java_lang_garbagecollector_collectiontime_ps_scavenge

java_lang_garbagecollector_collectiontime_parnew

java_lang_garbagecollector_collectiontime_marksweepcompact

java_lang_garbagecollector_collectiontime_ps_marksweep

java_lang_garbagecollector_collectiontime_concurrentmarksweep

java_lang_garbagecollector_collectiontime_g1_young_generation

java_lang_garbagecollector_collectiontime_g1_old_generation

java_lang_garbagecollector_collectiontime_g1_mixed_generation

java_lang_operatingsystem_committedvirtualmemorysize

Amazon FSx

CloudWatch Application Insights mendukung metrik berikut:

DataReadBytes

DataWriteBytes

DataReadOperations

DataWriteOperations

MetadataOperations

FreeStorageCapacity

FreeDataStorageCapacity

LogicalDiskUsage

PhysicalDiskUsage

Amazon VPC

CloudWatch Application Insights mendukung metrik berikut:

NetworkAddressUsage

NetworkAddressUsagePeered

VPC FirewallQueryVolume

NAT Gateway Amazon VPC

CloudWatch Application Insights mendukung metrik berikut:

ErrorPortAllocation

IdleTimeoutCount

Pemeriksaan kondisi kesehatan Amazon Route 53

CloudWatch Application Insights mendukung metrik berikut:

ChildHealthCheckHealthyCount

ConnectionTime

HealthCheckPercentageHealthy

HealthCheckStatus

SSL HandshakeTime

TimeToFirstByte

Zona yang di-hosting Amazon Route 53

CloudWatch Application Insights mendukung metrik berikut:

DNSQueries

DNSSEC InternalFailure

DNSSEC KeySigningKeysNeedingAction

DNSSEC KeySigningKeyMaxNeedingActionAge

DNSSEC KeySigningKeyAge

Amazon Route 53 Resolver titik akhir

CloudWatch Application Insights mendukung metrik berikut:

EndpointHealthyeniCount

EndpointUnHealthyeniCount

InboundQueryVolume

OutboundQueryVolume

OutboundQueryAggregateVolume

AWS Network Firewall kelompok aturan

CloudWatch Application Insights mendukung metrik berikut:

FirewallRuleGroupQueryVolume

AWS Network Firewall asosiasi kelompok aturan

CloudWatch Application Insights mendukung metrik berikut:

FirewallRuleGroupVpcQueryVolume

Metrik-metrik dengan persyaratan titik data

Untuk metrik-metrik yang tidak memiliki ambang batas bawaan yang jelas untuk alarm, Wawasan Aplikasi akan menunggu hingga metrik-metrik tersebut memiliki titik data yang cukup untuk memprediksi ambang batas yang masuk akal untuk mengaktifkan peringatan. Persyaratan titik data metrik yang diperiksa oleh CloudWatch Application Insights sebelum alarm dibuat adalah:

- Metrik tersebut memiliki setidaknya 100 titik data dari 15 hingga 2 hari terakhir.

- Metrik tersebut memiliki setidaknya 100 titik data dari hari terakhir.

Metrik-metrik berikut memenuhi persyaratan titik data ini. Perhatikan bahwa metrik CloudWatch agen membutuhkan waktu hingga satu jam untuk membuat alarm.

Metrik-metrik

- AWS/ApplicationELB
- AWS/ AutoScaling
- AWS/EC2
- Elastic Block Store (EBS)
- AWS/ELB
- AWS/RDS
- AWS/Lambda
- AWS/SQS
- AWS/CWAgent
- AWS/DynamoDB
- AWS/S3
- AWS/States
- AWS/ ApiGateway
- AWS/SNS

AWS/ApplicationELB

ActiveConnectionCount

ConsumedLCUs

HTTPCode_ELB_4XX_Count

HTTPCode_Target_2XX_Count

HTTPCode_Target_3XX_Count

HTTPCode_Target_4XX_Count

HTTPCode_Target_5XX_Count

NewConnectionCount

ProcessedBytes

TargetResponseTime

UnHealthyHostCount

AWS/ AutoScaling

GroupDesiredCapacity

GroupInServiceInstances

GroupMaxSize

GroupMinSize

GroupPendingInstances

GroupStandbyInstances

GroupTerminatingInstances

GroupTotalInstances

AWS/EC2

CPU CreditBalance

CPU CreditUsage

CPU SurplusCreditBalance

CPU SurplusCreditsCharged

CPUUtilization

DiskReadBytes

DiskReadOps

DiskWriteBytes

DiskWriteOps

EBS% ByteBalance

EBSIOBalance%

EBS ReadBytes

EBS ReadOps

EBS WriteBytes

EBS WriteOps

NetworkIn

NetworkOut

NetworkPacketsIn

NetworkPacketsOut

Elastic Block Store (EBS)

VolumeReadBytes

VolumeWriteBytes

VolumeReadOps

VolumeWriteOps

VolumeTotalReadTime

VolumeTotalWriteTime

VolumeIdleTime

VolumeQueueLength

VolumeThroughputPercentage

VolumeConsumedReadWriteOps

BurstBalance

AWS/ELB

DiperkirakanAlb ActiveConnectionCount

EstimatedALBConsumedLCUs

DiperkirakanAlb NewConnectionCount

EstimatedProcessedBytes

HTTPCode_Backend_4XX

HTTPCode_Backend_5XX

HealthyHostCount

Latensi

RequestCount

SurgeQueueLength

UnHealthyHostCount

AWS/RDS

ActiveTransactions

AuroraBinlogReplicaLag

AuroraReplicaLag

BackupRetentionPeriodStorageUsed

BinLogDiskUsage

BlockedTransactions

CPU CreditBalance

CommitLatency

CommitThroughput

DDLLatency

DDLThroughput

DMLLatency

DMLThroughput

DatabaseConnections

Deadlocks

DeleteLatency

DeleteThroughput

DiskQueueDepth

EngineUptime

FreeLocalStorage

FreeStorageSpace

FreeableMemory

InsertLatency

InsertThroughput

LoginFailures

NetworkReceiveThroughput

NetworkThroughput

NetworkTransmitThroughput

Queries

ReadIOPS

ReadThroughput

SelectLatency

SelectThroughput

SnapshotStorageUsed

TotalBackupStorageBilled

UpdateLatency

UpdateThroughput

VolumeBytesUsed

VolumeReadIOP

VolumeWriteIOP

WriteIOPS

WriteThroughput

AWS/Lambda

Kesalahan

DeadLetterErrors

Durasi

Trotel

IteratorAge

ProvisionedConcurrencySpilloverInvocations

AWS/SQS

ApproximateAgeOfOldestMessage

ApproximateNumberOfMessagesDelayed

ApproximateNumberOfMessagesNotVisible

ApproximateNumberOfMessagesVisible

NumberOfEmptyReceives

NumberOfMessagesDeleted

NumberOfMessagesReceived

NumberOfMessagesSent

AWS/CWAgent

LogicalDisk % Ruang Bebas

Memori % Byte Berkomitmen yang Digunakan

Mbyte Memori yang Tersedia

Total Byte Antarmuka Jaringan/detik

File Halaman % Penggunaan

PhysicalDisk % Waktu Disk

PhysicalDisk Rata-rata. Detik Cakram/Baca

PhysicalDisk Rata-rata. Detik Cakram/Tulis

PhysicalDisk Byte Baca Disk/detik

PhysicalDisk Pembacaan Disk/detik

PhysicalDisk Byte Tulis Disk/detik

PhysicalDisk Disk Menulis/detik

Pemroses % Waktu Menganggur

Persentase Waktu Interupsi Prosesor

Persentase Pemroses Processor Time

Persentase Waktu Pengguna Prosesor

SQLServer:Metode Akses Catatan yang Diteruskan/detik

SQLServer:Metode Akses Pembagian Halaman/detik

SQLServer:Rasio hit cache Buffer Manajer Buffer

SQLServer:Harapan hidup halaman Manajer Buffer

SQLServer:Byte File Replika Basis Data yang Diterima/detik

SQLServer:Byte Log Replika Basis Data yang Diterima/detik

SQLServer:Log Replika Basis Data tersisa yang akan dibatalkan

SQLServer:Antrean Kiriman Log Replika Basis Data

SQLServer:Transaksi Tulis Tercermin Replika Basis Data/detik

SQLServer:Antrean Pemulihan Replika Basis Data

SQLServer:Byte Mengulang Replika Basis Data yang Tersisa

SQLServer:Byte Mengulang Replika Basis Data/detik

SQLServer:Log Total Replika Basis Data yang memerlukan pembatalan

SQLServer:Keterlambatan Transaksi Replika Basis Data

SQLServer:Proses Statistik Umum yang diblokir

SQLServer:Permintaan Batch Statistik SQL/detik

SQLServer:Statistik SQL Kompilasi SQL/detik

SQLServer:Statistik SQL Re-Kompilasi SQL/detik

Panjang Antrean Prosesor Sistem

Koneksi TCPv4 Ditetapkan

Koneksi TCPv6 Ditetapkan

AWS/DynamoDB

ConsumedReadCapacityUnits

ConsumedWriteCapacityUnits

ReadThrottleEvents

WriteThrottleEvents

TimeToLiveDeletedItemCount

ConditionalCheckFailedRequests

TransactionConflict

ReturnedRecordsCount

PendingReplicationCount

ReplicationLatency

AWS/S3

ReplicationLatency

BytesPendingReplication

OperationsPendingReplication

4xxKesalahan

5xxKesalahan

AllRequests

GetRequests

PutRequests

DeleteRequests

HeadRequests

PostRequests

SelectRequests

ListRequests

SelectScannedBytes

SelectReturnedBytes

FirstByteLatency

TotalRequestLatency

BytesDownloaded

BytesUploaded

AWS/States

ActivitiesScheduled

ActivitiesStarted

ActivitiesSucceeded

ActivityScheduleTime

ActivityRuntime

ActivityTime

LambdaFunctionsScheduled

LambdaFunctionsStarted

LambdaFunctionsSucceeded

LambdaFunctionScheduleTime

LambdaFunctionRuntime

LambdaFunctionTime

ServiceIntegrationsScheduled

ServiceIntegrationsStarted

ServiceIntegrationsSucceeded

ServiceIntegrationScheduleTime

ServiceIntegrationRuntime

ServiceIntegrationTime

ProvisionedRefillRate

ProvisionedBucketSize

ConsumedCapacity

ThrottledEvents

AWS/ ApiGateway

4XXError

IntegrationLatency

Latensi

DataProcessed

CacheHitCount

CacheMissCount

AWS/SNS

NumberOfNotificationsDelivered

NumberOfMessagesPublished

NumberOfNotificationsFailed

NumberOfNotificationsFilteredOut

NumberOfNotificationsFilteredOut-InvalidAttributes

NumberOfNotificationsFilteredOut-NoMessageAttributes

NumberOfNotificationsRedrivenToDlq

NumberOfNotificationsFailedToRedriveToDlq

SMS SuccessRate

Metrik-metrik yang direkomendasikan

Tabel berikut mencantumkan metrik-metrik yang direkomendasikan untuk setiap jenis komponen.

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
Instans EC2 (server Windows)	Default/Kustom	CPUUtilization StatusCheckFailed Pemroses % Waktu Pemroses Persentase Memori Bytes yang Dipesan dan Digunakan LogicalDisk % Ruang Bebas Mbyte Memori yang Tersedia
	Direktori Aktif	CPUUtilization StatusCheckFailed Pemroses % Waktu Pemroses Persentase Memori Bytes yang Dipesan dan Digunakan Mbyte Memori yang Tersedia Database ==> Persentase Cache Basis Data Instans yang di-Hit DirectoryServices Operasi Replikasi Tertunda DRA DirectoryServices Sinkronisasi Replikasi Tertunda DRA Kegagalan Kueri Rekursif DNS/detik

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
		LogicalDisk Rata-rata. Panjang Antrean Cakram
	Aplikasi Java	CPUUtilization StatusCheckFailed Pemroses % Waktu Pemroses Persentase Memori Bytes yang Dipesan dan Digunakan Mbyte Memori yang Tersedia java_lang_threading_threadcount java_lang_classloading_loadedclasscount java_lang_memory_heapmemoryusage_used java_lang_memory_heapmemoryusage_committed java_lang_operatingsystem_freephysicalmemorysize java_lang_operatingsystem_freeswapspacesize

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
	Front-End Web Microsoft IIS/.NET	CPUUtilization StatusCheckFailed Pemroses % Waktu Pemroses Persentase Memori Bytes yang Dipesan dan Digunakan Mbyte Memori yang Tersedia Pengecualian .NET CLR # dari Exceps Thrown/Detik Memori .NET CLR # Total Byte Berkomitmen Persentase Waktu dalam GC Memori .NET CLR Permintaan Aplikasi ASP.NET dalam Antrean Aplikasi Permintaan ASP.NET yang Masuk Antrean Memulai Ulang Aplikasi ASP.NET

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
	Tingkat Basis Data Server Microsoft SQL	CPUUtilization StatusCheckFailed Pemroses % Waktu Pemroses Persentase Memori Bytes yang Dipesan dan Digunakan Mbyte Memori yang Tersedia Persentase Penggunaan File Halaman Panjang Antrean Prosesor Sistem Total/Detik Byte Antarmuka Jaringan PhysicalDisk % Waktu Disk SQLServer:Rasio Hit Cache Buffer Manajer Buffer SQLServer:Harapan Hidup Halaman Manajer Buffer SQLServer:Proses Statistik Umum yang Diblokir SQLServer:Koneksi Pengguna Statistik Umum SQLServer:Jumlah Kunci Penguncian/Detik SQLServer:Permintaan Batch Statistik SQL/detik

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
	MySQL	CPUUtilization StatusCheckFailed Pemroses % Waktu Pemroses Persentase Memori Bytes yang Dipesan dan Digunakan LogicalDisk % Ruang Bebas Mbyte Memori yang Tersedia
	.NET workerpool/Tingkat Medium	CPUUtilization StatusCheckFailed Pemroses % Waktu Pemroses Persentase Memori Bytes yang Dipesan dan Digunakan Mbyte Memori yang Tersedia Pengecualian .NET CLR # dari Exceps Thrown/Detik Memori .NET CLR # Total Byte Berkomitmen Persentase Waktu dalam GC Memori .NET CLR

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
	Tingkat Inti .NET	CPUUtilization StatusCheckFailed Pemroses % Waktu Pemroses Persentase Memori Bytes yang Dipesan dan Digunakan Mbyte Memori yang Tersedia
	Oracle	CPUUtilization StatusCheckFailed Pemroses % Waktu Pemroses Persentase Memori Bytes yang Dipesan dan Digunakan LogicalDisk % Ruang Bebas Mbyte Memori yang Tersedia
	Postgres	CPUUtilization StatusCheckFailed Pemroses % Waktu Pemroses Persentase Memori Bytes yang Dipesan dan Digunakan LogicalDisk % Ruang Bebas Mbyte Memori yang Tersedia

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
	SharePoint	CPUUtilization StatusCheckFailed Pemroses % Waktu Pemroses Persentase Memori Bytes yang Dipesan dan Digunakan Mbyte Memori yang Tersedia Pembuangan API Cache Aplikasi ASP.NET Permintaan ASP.NET yang Ditolak Memulai Ulang Proses Pekerja ASP.NET Halaman Memori/detik SharePoint Menerbitkan Cache Publishing cache flushes /second SharePoint Yayasan Melaksanakan Waktu/Permintaan Halaman SharePoint Cache Berbasis Disk Jumlah total pemadatan cache SharePoint Rasio hit cache Blob Berbasis Disk

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
		SharePoint Rasio pengisian Cache Blob Cache Berbasis Disk SharePoint Cache Blob Berbasis Disk memerah /detik Permintaan ASP.NET yang Masuk Antrean Permintaan Aplikasi ASP.NET dalam Antrean Aplikasi Memulai Ulang Aplikasi ASP.NET LogicalDisk Rata-rata. Detik Cakram/Tulis LogicalDisk Rata-rata. Detik Cakram/Baca Persentase Waktu Interupsi Prosesor
Instans EC2 (server Linux)	Default/Kustom	CPUUtilization StatusCheckFailed disk_used_percent mem_used_percent

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
	Aplikasi Java	CPUUtilization StatusCheckFailed disk_used_percent mem_used_percent java_lang_threading_threadcount java_lang_classloading_loadedclasscount java_lang_memory_heapmemoryusage_used java_lang_memory_heapmemoryusage_committed java_lang_operatingsystem_freephysicalmemorysize java_lang_operatingsystem_freeswapspacesize
	Tingkat Inti .NET atau Tingkat Basis Data Server SQL	CPUUtilization StatusCheckFailed disk_used_percent mem_used_percent

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
	Oracle	CPUUtilization StatusCheckFailed disk_used_percent mem_used_percent
	Postgres	CPUUtilization StatusCheckFailed disk_used_percent mem_used_percent

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
Grup instans EC2	SAP HANA multi-simpul atau simpul tunggal	• hanadb_server_startup_time_variations_seconds • hanadb_level_5_alerts_count • hanadb_level_4_alerts_count • hanadb_out_of_memory_events_count • hanadb_max_trigger_read_ratio_percent • hanadb_max_trigger_write_ratio_percent • hanadb_log_switch_race_ratio_percent • hanadb_time_since_last_savepoint_seconds • hanadb_disk_usage_highlevel_percent • hanadb_current_allocation_limit_used_percent • hanadb_table_allocation_limit_used_percent • hanadb_cpu_usage_percent • hanadb_plan_cache_hit_ratio_percent • hanadb_last_data_backup_age_days

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
Volume EBS	Semua	VolumeReadBytes VolumeWriteBytes VolumeReadOps VolumeWriteOps VolumeQueueLength VolumeThroughputPercentage VolumeConsumedRead WriteOps BurstBalance
ELB Klasik	Setiap	HTTPCode_Backend_4XX HTTPCode_Backend_5XX Latensi SurgeQueueLength UnHealthyHostCount
ELB Aplikasi	Setiap	HTTPCode_Target_4X X_Count HTTPCode_Target_5X X_Count TargetResponseTime UnHealthyHostCount

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
Instans Basis Data RDS	Setiap	CPUUtilization ReadLatency WriteLatency BurstBalance GagalSQL ServerAgentJobsCount
Klaster Basis Data RDS	Setiap	CPUUtilization CommitLatency DatabaseConnections Deadlocks FreeableMemory NetworkThroughput VolumeBytesUsed
Fungsi Lambda	Setiap	Durasi Kesalahan IteratorAge ProvisionedConcurrencySpilloverInvocations Trotel

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
Antrean SQS	Sebarang	ApproximateAgeOfOldestMessage ApproximateNumberOfMessagesVisible NumberOfMessagesSent
Tabel Amazon DynamoDB	Sebarang	SystemErrors UserErrors ConsumedReadCapacityUnits ConsumedWriteCapacityUnits ReadThrottleEvents WriteThrottleEvents ConditionalCheckFailedRequests TransactionConflict

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
Buket Amazon S3	Setiap	Jika konfigurasi aplikasi ulang dengan Pengendalian Waktu Aplikasi Ulang (Replication Time Control [RTC]) diaktifkan: ReplicationLatency BytesPendingReplication OperationsPendingReplication Jika metrik permintaan diaktifkan: 5xxErrors 4xxErrors BytesDownloaded BytesUploaded

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
AWS Step Functions	Sebarang	Umum • ExecutionThrottled • ExecutionsAborted • ProvisionedBucketSize • ProvisionedRefillRate • ConsumedCapacity Jika jenis mesin status adalah **`EXPRESS`** atau tingkat grup log adalah **`OFF`** • ExecutionsFailed • ExecutionsTimedOut Jika mesin status memiliki fungsi Lambda • LambdaFunctionsFailed • LambdaFunctionsTimedOut Jika mesin status memiliki aktivitas • ActivitiesFailed • ActivitiesTimedOut • ActivitiesHeartbea tTimedOut Jika mesin status memiliki integrasi layanan • ServiceIntegrationsFailed

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
		• ServiceIntegration sTimedOut
Tahap API REST API Gateway	Setiap	• 4xxErrors • 5xxErrors • Latensi

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
Klaster ECS	Sebarang	CpuUtilized MemoryUtilized NetworkRxBytes NetworkTxBytes RunningTaskCount PendingTaskCount StorageReadBytes StorageWriteBytes CPUReservation (Khusus Tipe Peluncuran EC2) CPUUtilization (Khusus Tipe Peluncuran EC2) MemoryReservation (Hanya Jenis Peluncuran EC2) MemoryUtilization (Hanya Jenis Peluncuran EC2) GPUReservation (Khusus Tipe Peluncuran EC2) instance_cpu_utilization (Khusus Tipe Peluncuran EC2) instance_filesystem_utilization (Khusus Tipe Peluncuran EC2)

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
		instance_memory_utilization (Khusus Tipe Peluncuran EC2) instance_network_total_bytes (Khusus Tipe Peluncuran EC2)

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
	Aplikasi Java	CpuUtilized MemoryUtilized NetworkRxBytes NetworkTxBytes RunningTaskCount PendingTaskCount StorageReadBytes StorageWriteBytes CPUReservation (Khusus Tipe Peluncuran EC2) CPUUtilization (Khusus Tipe Peluncuran EC2) MemoryReservation (Hanya Jenis Peluncuran EC2) MemoryUtilization (Hanya Jenis Peluncuran EC2) GPUReservation (Khusus Tipe Peluncuran EC2) instance_cpu_utilization (Khusus Tipe Peluncuran EC2) instance_filesystem_utilization (Khusus Tipe Peluncuran EC2)

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
		instance_memory_utilization (Khusus Tipe Peluncuran EC2) instance_network_total_bytes (Khusus Tipe Peluncuran EC2) java_lang_threading_threadcount java_lang_classloading_loadedclasscount java_lang_memory_heapmemoryusage_used java_lang_memory_heapmemoryusage_committed java_lang_operatingsystem_freephysicalmemorysize java_lang_operatingsystem_freeswapspacesize

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
Layanan ECS	Setiap	CPUUtilization MemoryUtilization CpuUtilized MemoryUtilized NetworkRxBytes NetworkTxBytes RunningTaskCount PendingTaskCount StorageReadBytes StorageWriteBytes

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
	Aplikasi Java	CPUUtilization MemoryUtilization CpuUtilized MemoryUtilized NetworkRxBytes NetworkTxBytes RunningTaskCount PendingTaskCount StorageReadBytes StorageWriteBytes java_lang_threading_threadcount java_lang_classloading_loadedclasscount java_lang_memory_heapmemoryusage_used java_lang_memory_heapmemoryusage_committed java_lang_operatingsystem_freephysicalmemorysize java_lang_operatingsystem_freeswapspacesize

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
Klaster EKS	Setiap	cluster_failed_node_count
		node_cpu_reserved_capacity
		node_cpu_utilization
		node_filesystem_utilization
		node_memory_reserv ed_capacity
		node_memory_utilization
		node_network_total_bytes
		pod_cpu_reserved_capacity
		pod_cpu_utilization
		pod_cpu_utilization_over_po d_limit
		pod_memory_reserve d_capacity
		pod_memory_utilization
		pod_memory_utilization_over _pod_limit
		pod_network_rx_bytes
		pod_network_tx_bytes

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
	Aplikasi Java	cluster_failed_node_count node_cpu_reserved_capacity node_cpu_utilization node_filesystem_utilization node_memory_reserved_capacity node_memory_utilization node_network_total_bytes pod_cpu_reserved_capacity pod_cpu_utilization pod_cpu_utilization_over_pod_limit pod_memory_reserved_capacity pod_memory_utilization pod_memory_utilization_over_pod_limit pod_network_rx_bytes pod_network_tx_bytes java_lang_threading_threadcount java_lang_classloading_loadedclasscount

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
		java_lang_memory_heapmemoryusage_used java_lang_memory_heapmemoryusage_committed java_lang_operatingsystem_freephysicalmemorysize java_lang_operatingsystem_freeswapspacesize

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
Klaster Kubernetes pada EC2	Setiap	cluster_failed_node_count node_cpu_reserved_capacity node_cpu_utilization node_filesystem_utilization node_memory_reserved_capacity node_memory_utilization node_network_total_bytes pod_cpu_reserved_capacity pod_cpu_utilization pod_cpu_utilization_over_pod_limit pod_memory_reserved_capacity pod_memory_utilization pod_memory_utilization_over_pod_limit pod_network_rx_bytes pod_network_tx_bytes

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
	Aplikasi Java	cluster_failed_node_count node_cpu_reserved_capacity node_cpu_utilization node_filesystem_utilization node_memory_reserved_capacity node_memory_utilization node_network_total_bytes pod_cpu_reserved_capacity pod_cpu_utilization pod_cpu_utilization_over_pod_limit pod_memory_reserved_capacity pod_memory_utilization pod_memory_utilization_over_pod_limit pod_network_rx_bytes pod_network_tx_bytes java_lang_threading_threadcount java_lang_classloading_loadedclasscount

Jenis komponen	Jenis beban kerja	Metrik yang direkomendasikan
		java_lang_memory_heapmemoryusage_used java_lang_memory_heapmemoryusage_committed java_lang_operatingsystem_freephysicalmemorysize java_lang_operatingsystem_freeswapspacesize

Tabel berikut mencantumkan proses dan metrik proses yang direkomendasikan untuk setiap jenis komponen. CloudWatch Application Insights tidak merekomendasikan pemantauan proses untuk proses yang tidak berjalan pada instance.

Jenis komponen	Jenis beban kerja	Proses yang direkomendasikan	Metrik yang direkomendasikan

Instans EC2 (server Windows)	Front-End Web Microsoft IIS/.NET	`w3wp`	`procstat cpu_usage ,`
			
			`procstat memory_rss ,`
			
			`procstat memory_vms ,`
			
			`procstat read_bytes ,`
			
			`procstat write_bytes`

	Tingkat Basis Data Server Microsoft SQL	`SQLAgent`	`procstat cpu_usage ,`

Jenis komponen	Jenis beban kerja	Proses yang direkomendasikan	Metrik yang direkomendasikan
			`procstat memory_rss` , `procstat memory_vms` , `procstat read_bytes` , `procstat write_bytes`
		`sqlservr`	`procstat cpu_usage` , `procstat memory_rss` , `procstat memory_vms` , `procstat read_bytes` , `procstat write_bytes`
		`sqlwriter`	`procstat cpu_usage` , `procstat memory_rss`

Jenis komponen	Jenis beban kerja	Proses yang direkomendasikan	Metrik yang direkomendasikan
		`Reporting ServicesS ervice`	`procstat cpu_usage ,` `procstat memory_rss`

		`MsDtsServr`	`procstat cpu_usage ,`
			`procstat memory_rss ,`
			`procstat memory_vms ,`
			`procstat read_bytes ,`
			`procstat write_bytes`

		`Msmdsrv`	`procstat cpu_usage ,`
			`procstat memory_rss ,`
			`procstat memory_vms ,`
			`procstat read_bytes ,`
			`procstat write_bytes`

Jenis komponen	Jenis beban kerja	Proses yang direkomendasikan	Metrik yang direkomendasikan
	.NET workerpool/ Tingkat Medium	`w3wp`	`procstat cpu_usage` , `procstat memory_rss` , `procstat memory_vms` , `procstat read_bytes` , `procstat write_bytes`
	Tingkat Inti .NET	`w3wp`	`procstat cpu_usage` , `procstat memory_rss` , `procstat memory_vms` , `procstat read_bytes` , `procstat write_bytes`

Metrik-metrik Penghitung Performa

Metrik-metrik Penghitung Performa direkomendasikan hanya untuk instans ketika serangkaian Penghitung Performa yang sesuai diinstal pada instans Windows.

Nama metrik Penghitung Performa	Nama rangkaian Penghitung Performa
Pengecualian .NET CLR # dari Exceps Thrown	Pengecualian .NET CLR
Pengecualian .NET CLR # dari Exceps Thrown/Detik	Pengecualian .NET CLR
Pengecualian .NET CLR # dari Penyaring/Detik	Pengecualian .NET CLR
Pengecualian .NET CLR # dari Finallys/Detik	Pengecualian .NET CLR
Pengecualian .NET CLR untuk Menelusuri Kedalaman/Detik	Pengecualian .NET CLR
Interop .NET CLR # dari CCW	Interop .NET CLR
Interop .NET CLR # dari Stub	Interop .NET CLR
Interop .NET CLR # ekspor TLB/Detik	Interop .NET CLR
Interop .NET CLR # impor TLB/Detik	Interop .NET CLR
Interop .NET CLR # Marshaling	Interop .NET CLR
.NET CLR Jit % Waktu di Jit	Jit .NET CLR
Kegagalan Jit Standar Jit .NET CLR	Jit .NET CLR
Persentase Waktu Pemuatan Pemuatan .NET CLR	Pemuatan .NET CLR
Tingkat Pemuatan Kegagalan Beban .NET CLR	Pemuatan .NET CLR
.NET CLR Tingkat LocksAndThreads Perbalahan/Detik	.NET CLR LocksAndThreads
.NET CLR Panjang LocksAndThreads Antrian/Detik	.NET CLR LocksAndThreads
Memori .NET CLR # Total Byte Berkomitmen	Memori .NET CLR

Nama metrik Penghitung Performa	Nama rangkaian Penghitung Performa
Persentase Waktu dalam GC Memori .NET CLR	Memori .NET CLR
.NET CLR Networking 4.0.0.0 Waktu HttpWebRequest Antrian Rata-rata	Jaringan 4.0.0.0 .NET CLR
.NET CLR Networking 4.0.0.0 HttpWebRequests Dibatarkan/Detik	Jaringan 4.0.0.0 .NET CLR
.NET CLR Networking 4.0.0.0 HttpWebRequests Gagal/Detik	Jaringan 4.0.0.0 .NET CLR
.NET CLR Networking 4.0.0.0 Antrean/Detik HttpWebRequests	Jaringan 4.0.0.0 .NET CLR
Kegagalan Ping Proses Pekerja Total APP_POOL_WAS	APP_POOL_WAS
Memulai Ulang Aplikasi ASP.NET	ASP.NET
Permintaan ASP.NET yang Ditolak	ASP.NET
Memulai Ulang Proses Pekerja ASP.NET	ASP.NET
Pembuangan API Cache Aplikasi ASP.NET	Aplikasi ASP.NET
Persentase Waktu Prosesor Terkelola Aplikasi ASP.NET (perkiraan)	Aplikasi ASP.NET
Total Kesalahan Aplikasi ASP.NET/Detik	Aplikasi ASP.NET
Kesalahan-kesalahan Aplikasi ASP.NET yang Tidak Ditangani Selama Eksekusi/Detik	Aplikasi ASP.NET
Permintaan Aplikasi ASP.NET dalam Antrean Aplikasi	Aplikasi ASP.NET
Permintaan Aplikasi ASP.NET/Detik	Aplikasi ASP.NET

Nama metrik Penghitung Performa	Nama rangkaian Penghitung Performa
Waktu Tunggu Permintaan ASP.NET	ASP.NET
Permintaan ASP.NET yang Masuk Antrean	ASP.NET
Database ==> Persentase Cache Basis Data Instans yang di-Hit	Basis data ==> Instans
Basis data ==> Latensi Rata-rata Pembacaan Basis Data I/O Instans	Basis data ==> Instans
Basis data ==> Pembacaan Basis Data I/O Instans/detik	Basis data ==> Instans
Basis data ==> Latensi Rata-rata Penulisan Log I/O Instans	Basis data ==> Instans
DirectoryServices Operasi Replikasi Tertunda DRA	DirectoryServices
DirectoryServices Sinkronisasi Replikasi Tertunda DRA	DirectoryServices
DirectoryServices Waktu Mengikat LDAP	DirectoryServices
Kueri Rekursif DNS/detik	DNS
Kegagalan Kueri Rekursif DNS/detik	DNS
Kueri DNS TCP Diterima/detik	DNS
Total Kueri DNS Diterima/detik	DNS
Respon Total DNS Dikirim/detik	DNS
Kueri UDP DNS Diterima/detik	DNS
Antrian Permintaan Layanan HTTP CurrentQueueSize	Antrean Permintaan Layanan HTTP

Nama metrik Penghitung Performa	Nama rangkaian Penghitung Performa
LogicalDisk % Ruang Bebas	LogicalDisk
LogicalDisk Rata-rata. Detik Cakram/Tulis	LogicalDisk
LogicalDisk Rata-rata. Detik Cakram/Baca	LogicalDisk
LogicalDisk Rata-rata. Panjang Antrean Cakram	LogicalDisk
Persentase Memori Bytes yang Dipesan dan Digunakan	Memori
Mbyte Memori yang Tersedia	Memori
Halaman Memori/Detik	Memori
Memori Jangka Panjang Rata-Rata Siaga Cache Seumur Hidup (dt)	Memori
Total Byte Antarmuka Jaringan/detik	Antarmuka Jaringan
Byte Antarmuka Jaringan yang Diterima/detik	Antarmuka Jaringan
Byte Antarmuka Jaringan yang Dikirim/detik	Antarmuka Jaringan
Bandwidth Saat Ini Antarmuka Jaringan	Antarmuka Jaringan
Persentase Penggunaan File Halaman	File Halaman
PhysicalDisk % Waktu Disk	PhysicalDisk
PhysicalDisk Rata-rata. Panjang Antrean Cakram	PhysicalDisk
PhysicalDisk Rata-rata. Detik Cakram/Baca	PhysicalDisk
PhysicalDisk Rata-rata. Detik Cakram/Tulis	PhysicalDisk
PhysicalDisk Byte Baca Disk/Detik	PhysicalDisk

Nama metrik Penghitung Performa	Nama rangkaian Penghitung Performa
PhysicalDisk Disk Membaca/Detik	PhysicalDisk
PhysicalDisk Byte Tulis Disk/Detik	PhysicalDisk
PhysicalDisk Disk Menulis/Detik	PhysicalDisk
Persentase Waktu Menganggur Prosesor	Prosesor
Persentase Waktu Interupsi Prosesor	Prosesor
Persentase Pemroses Processor Time	Prosesor
Persentase Waktu Pengguna Prosesor	Prosesor
SharePoint Rasio pengisian Cache Blob Cache Berbasis Disk	SharePoint Cache Berbasis Disk
SharePoint Cache Blob Berbasis Disk memerah /detik	SharePoint Cache Berbasis Disk
SharePoint Rasio hit cache Blob Berbasis Disk	SharePoint Cache Berbasis Disk
SharePoint Cache Berbasis Disk Jumlah total pemadatan cache	SharePoint Cache Berbasis Disk
SharePoint Yayasan Melaksanakan Waktu/Permintaan Halaman	SharePoint Yayasan
SharePoint Menerbitkan Cache Publishing cache flushes /second	SharePoint Menerbitkan Cache
Statistik Seluruh Sistem Keamanan Otentikasi Kerberos	Statistik Seluruh Sistem Keamanan
Statistik Seluruh Sistem Keamanan Otentikasi NTLM	Statistik Seluruh Sistem Keamanan
SQLServer:Metode Akses Catatan yang Diteruskan/Detik	SQLServer:Metode Akses

Nama metrik Penghitung Performa	Nama rangkaian Penghitung Performa
SQLServer:Metode Akses Pemindaian Penuh/Detik	SQLServer:Metode Akses
SQLServer:Metode Akses Pembagian Halaman/Detik	SQLServer:Metode Akses
SQLServer:Rasio Hit Cache Buffer Manajer Buffer	SQLServer:Manajer Buffer
SQLServer:Harapan Hidup Halaman Manajer Buffer	SQLServer:Manajer Buffer
SQLServer:Byte File Replika Basis Data yang Diterima/detik	SQLServer:Replika Basis Data
SQLServer:Byte Log Replika Basis Data yang Diterima/detik	SQLServer:Replika Basis Data
SQLServer:Log Replika Basis Data tersisa yang akan dibatalkan	SQLServer:Replika Basis Data
SQLServer:Antrean Kiriman Log Replika Basis Data	SQLServer:Replika Basis Data
SQLServer:Transaksi Tulis Tercermin Replika Basis Data/detik	SQLServer:Replika Basis Data
SQLServer:Antrean Pemulihan Replika Basis Data	SQLServer:Replika Basis Data
SQLServer:Byte Mengulang Replika Basis Data yang Tersisa	SQLServer:Replika Basis Data
SQLServer:Byte Mengulang Replika Basis Data/detik	SQLServer:Replika Basis Data
SQLServer:Log Total Replika Basis Data yang memerlukan pembatalan	SQLServer:Replika Basis Data

Nama metrik Penghitung Performa	Nama rangkaian Penghitung Performa
SQLServer:Keterlambatan Transaksi Replika Basis Data	SQLServer:Replika Basis Data
SQLServer:Proses Statistik Umum yang Diblokir	SQLServer:Statistik Umum
SQLServer:Koneksi Pengguna Statistik Umum	SQLServer:Statistik Umum
SQLServer:Waktu Tunggu Latch Rata-rata (ms)	SQLServer:Kait
SQLServer:Waktu Tunggu Rata-rata Kunci (ms)	SQLServer:Kunci
SQLServer:Kunci Batas Waktu Kunci/detik	SQLServer:Kunci
SQLServer:Kunci Kunci Menunggu/detik	SQLServer:Kunci
SQLServer:Jumlah Kunci Penguncian/Detik	SQLServer:Kunci
SQLServer:Pemberian Memori Manajer Memori Tertunda	SQLServer:Manajer Memori
SQLServer:Permintaan Batch Statistik SQL/detik	SQLServer:Statistik SQL
SQLServer:Statistik SQL Kompilasi SQL/Detik	SQLServer:Statistik SQL
SQLServer:Statistik SQL Re-Kompilasi SQL/Detik	SQLServer:Statistik SQL
Panjang Antrean Prosesor Sistem	Sistem
Koneksi TCPv4 Ditetapkan	TCPv4
Koneksi TCPv6 Ditetapkan	TCPv6
W3SVC_W3WP File Cache Terbilas	W3SVC_W3WP
W3SVC_W3WP File Cache Hilang	W3SVC_W3WP

Nama metrik Penghitung Performa	Nama rangkaian Penghitung Performa
W3SVC_W3WP Permintaan/Detik	W3SVC_W3WP
W3SVC_W3WP URI Cache Terbilas	W3SVC_W3WP
W3SVC_W3WP URI Cache Hilang	W3SVC_W3WP
Byte Layanan Web yang Diterima/Detik	Layanan Web
Byte Layanan Web yang Dikirim/Detik	Layanan Web
Upaya Koneksi Layanan Web/Detik	Layanan Web
Koneksi Saat Ini Layanan Web	Layanan Web
Permintaan Get Layanan Web/Detik	Layanan Web
Permintaan Post Layanan Web/Detik	Layanan Web

Menggunakan tampilan kondisi sumber daya yang ada di CloudWatch

Anda dapat menggunakan tampilan kondisi sumber daya untuk secara otomatis menemukan, mengelola, dan memvisualisasikan kondisi dan performa host di aplikasi mereka dalam satu tampilan. Anda dapat memvisualisasikan kondisi host mereka dengan sebuah dimensi performa seperti CPU atau memori, dan memotong ratusan host dalam satu tampilan tunggal dengan menggunakan filter. Anda dapat memfilter berdasarkan tag atau kasus penggunaan, seperti host di grup Auto Scaling yang sama atau host yang menggunakan penyeimbang beban yang sama,

Prasyarat

Untuk memastikan bahwa Anda mendapatkan manfaat penuh dari tampilan kondisi sumber daya tersebut, Anda perlu memeriksa apakah Anda telah memenuhi prasyarat berikut.

- Untuk melihat pemanfaatan memori host Anda dan menggunakannya sebagai sebuah filter, Anda harus melakukan instalasi agen CloudWatch di host Anda dan mengaturnya agar agen tersebut mengirimkan sebuah metrik memori ke CloudWatch di ruang nama `CWAgent` bawaan. Pada instans Linux dan macOS, agen CloudWatch tersebut harus mengirimkan metrik

`mem_used_percent`. Pada instans Windows, agen tersebut harus mengirimkan metrik `Memory % Committed Bytes In Use`. Metrik-metrik ini akan disertakan jika Anda menggunakan panduan untuk membuat file konfigurasi agen CloudWatch dan memilih salah satu set metrik yang telah ditentukan sebelumnya. Metrik-metrik yang dikumpulkan oleh agen CloudWatch yang ditagih sebagai metrik kustom. Untuk informasi selengkapnya, lihat Instalasi CloudWatch agen.

Bila Anda menggunakan agen CloudWatch untuk mengumpulkan metrik-metrik memori ini untuk Anda gunakan dengan tampilan kondisi sumber daya, Anda harus menyertakan bagian-bagian berikut dalam file konfigurasi agen CloudWatch. Bagian-bagian ini berisi pengaturan dimensi bawaan dan dibuat secara bawaan, jadi Anda tidak boleh mengubah bagian mana pun dari bagian ini menjadi sesuatu yang berbeda dari apa yang ditunjukkan dalam contoh berikut.

```
"append_dimensions": {
  "ImageId": "${aws:ImageId}",
  "InstanceId": "${aws:InstanceId}",
  "InstanceType": "${aws:InstanceType}",
  "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
},
```

- Untuk melihat semua informasi yang tersedia dalam tampilan kondisi sumber daya, Anda harus masuk ke akun yang memiliki izin-izin berikut ini. Jika Anda masuk dengan izin yang lebih sedikit, maka Anda masih akan dapat menggunakan tampilan kondisi sumber daya, tetapi akan ada beberapa data performa yang tidak akan ditampilkan.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "autoscaling:Describe*",
                "cloudwatch:Describe*",
                "cloudwatch:Describe*",
                "cloudwatch:Get*",
                "cloudwatch:List*",
                "logs:Get*",
                "logs:Describe*",
                "sns:Get*",
                "sns:List*",
                "ec2:DescribeInstances",
                "ec2:DescribeInstanceStatus",
                "ec2:DescribeRegions"
```

```
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Cara melihat kondisi sumber daya yang ada di akun Anda

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Pemantauan Infrastruktur, Kondisi Sumber Daya.

   Halaman kondisi sumber daya ditampilkan dan kemudian menyajikan kotak untuk setiap host yang ada di akun Anda. Setiap kotak memiliki warna berdasarkan status host saat ini, berdasarkan pengaturan untuk Warna berdasarkan. Kotak-kotak host dengan sebuah simbol alarm memiliki satu atau beberapa alarm yang saat ini berada dalam status ALARM.

   Anda dapat melihat hingga 500 host dalam satu tampilan. Jika Anda memiliki lebih banyak host yang ada dalam akun Anda, maka Anda harus menggunakan pengaturan filter pada langkah 6 prosedur ini.
3. Untuk mengubah kriteria apa yang akan Anda gunakan untuk menunjukkan kondisi dari masing-masing host, Anda bisa memilih pengaturan untuk Warna berdasarkan. Anda dapat memilih Pemanfaatan CPU, Pemanfaatan Memori, atau Pemeriksaan status. Metrik pemanfaatan memori hanya tersedia untuk host yang menjalankan agen CloudWatch dan mengonfigurasinya untuk mengumpulkan metrik-metrik memori dan mengirimkannya ke ruang nama `CWAgent` bawaan. Untuk informasi selengkapnya, lihat Kumpulkan metrik, log, dan jejak dengan agen CloudWatch .
4. Untuk mengubah ambang batas dan warna yang digunakan untuk indikator kondisi pada kisi tersebut, pilih ikon roda gigi yang terletak di atas kisi.
5. Untuk beralih apakah akan menampilkan alarm di kisi host, silakan pilih atau hapus Tampilkan alarm di semua metrik.
6. Untuk membagi host dalam peta menjadi beberapa grup, Anda harus memilih kriteria pengelompokan untuk Grup berdasarkan.
7. Untuk mempersempit tampilan dengan host yang lebih sedikit, pilih kriteria filter untuk Filter berdasarkan. Anda dapat melakukan filter berdasarkan tag dan pengelompokan sumber daya seperti grup Auto Scaling, tipe instans, grup keamanan, dan banyak lagi.

8. Untuk mengurutkan host, silakan pilih kriteria pengurutan untuk Urutkan berdasarkan. Anda dapat melakukan pengurutan berdasarkan hasil pemeriksaan status, status instans, pemanfaatan CPU atau memori, dan jumlah alarm yang sedang berada dalam status ALARM.
9. Untuk melihat informasi selengkapnya tentang host, pilih kotak yang mewakili host tersebut. Sebuah panel popup ditampilkan. Jika Anda ini menelusuri lebih dalam informasi tentang host tersebut, Anda bisa memilih Lihat dasbor atau Lihat di daftar.

CloudWatch observabilitas lintas akun

Dengan observabilitas CloudWatch lintas akun Amazon, Anda dapat memantau dan memecahkan masalah aplikasi yang menjangkau beberapa akun dalam suatu Wilayah. Cari, visualisasikan, dan analisis metrik, log, jejak, aplikasi Application Insights, dan monitor Internet Monitor Anda dengan mulus di salah satu akun tertaut tanpa batas akun.

Siapkan satu atau beberapa AWS akun sebagai akun pemantauan dan tautkan dengan beberapa akun sumber. Akun pemantauan adalah akun AWS pusat yang dapat melihat dan berinteraksi dengan data observabilitas yang dihasilkan dari akun sumber. Akun sumber adalah AWS akun individu yang menghasilkan data observabilitas untuk sumber daya yang berada di dalamnya. Akun sumber membagikan data observabilitas mereka dengan akun pemantauan. Data observabilitas bersama dapat mencakup jenis telemetri berikut:

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

Untuk membuat tautan antara akun pemantauan dan akun sumber, Anda dapat menggunakan CloudWatch konsol. Atau, gunakan perintah Observability Access Manager di API AWS CLI dan. Untuk informasi selengkapnya, silakan lihat Referensi API Manajer Akses Observabilitas.

Sink adalah sumber daya yang mewakili titik lampiran dalam akun pemantauan. Akun sumber dapat menautkan ke sink untuk berbagi data observabilitas. Setiap sink dikelola oleh akun pemantauan di mana ia berada. Tautan observabilitas adalah sumber daya yang mewakili tautan yang dibuat antara akun sumber dan akun pemantauan. Tautan-tautan itu dikelola oleh akun sumber.

Untuk demonstrasi video tentang pengaturan observabilitas CloudWatch lintas akun, lihat video berikut.

Topik berikutnya menjelaskan cara mengatur observabilitas CloudWatch lintas akun di akun pemantauan dan akun sumber. Untuk informasi tentang CloudWatch dasbor lintas akun Lintas wilayah, lihat. Konsol CloudWatch lintas akun lintas Wilayah

Gunakan Organizations untuk akun sumber

Ada dua opsi untuk menautkan akun sumber ke akun pemantauan Anda. Anda dapat menggunakan salah satu atau kedua opsi.

- Gunakan AWS Organizations untuk menautkan akun di organisasi atau unit organisasi ke akun pemantauan.
- Hubungkan AWS akun individual ke akun pemantauan.

Kami menyarankan Anda menggunakan Organizations sehingga AWS akun baru yang dibuat nanti di organisasi secara otomatis terhubung ke observabilitas lintas akun sebagai akun sumber.

Detail tentang menautkan akun pemantauan dan akun sumber

- Setiap akun pemantauan dapat ditautkan ke sebanyak 100.000 akun sumber.
- Setiap akun sumber dapat berbagi data dengan sebanyak lima akun pemantauan.
- Anda dapat mengatur satu akun sebagai akun pemantauan dan akun sumber. Jika Anda melakukan hal ini, akun ini hanya mengirimkan data observabilitas dari dirinya sendiri ke akun pemantauan yang ditautkan. Ini tidak menyampaikan data dari akun sumbernya.
- Akun pemantauan menentukan jenis telemetri mana yang dapat dibagikan dengannya. Akun sumber menentukan jenis telemetri mana yang ingin dibagikan.
  - Jika ada lebih banyak jenis telemetri yang dipilih di akun pemantauan daripada di akun sumber, akun tersebut ditautkan. Hanya tipe data yang dipilih di kedua akun yang dibagikan.
  - Jika ada lebih banyak jenis telemetri yang dipilih di akun sumber daripada di akun pemantauan, pembuatan tautan gagal dan tidak ada yang dibagikan.
- Untuk menghapus tautan antar akun, lakukan dari akun sumber.
- Untuk menghapus sink di akun pemantauan, Anda harus terlebih dahulu menghapus semua tautan ke akun pemantauan.

Penetapan Harga

Pengamatan lintas akun CloudWatch hadir tanpa biaya tambahan untuk log dan metrik, dan salinan jejak pertama gratis. Untuk informasi selengkapnya tentang harga, lihat CloudWatchHarga Amazon.

Daftar Isi

Tautkan akun pemantauan dengan akun sumber

Topik di bagian ini menjelaskan cara menyiapkan tautan antara akun pemantauan dan akun sumber.

Kami menyarankan Anda membuat AWS akun baru untuk berfungsi sebagai akun pemantauan untuk organisasi Anda.

Daftar Isi

Izin yang diperlukan

Untuk membuat tautan antara akun pemantauan dan akun sumber, Anda harus masuk dengan izin tertentu.

- Untuk menyiapkan akun pemantauan – Anda harus memiliki akses administrator penuh di akun pemantauan, atau Anda harus masuk ke akun tersebut dengan izin berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowSinkModification",
            "Effect": "Allow",
            "Action": [
                "oam:CreateSink",
                "oam:DeleteSink",
                "oam:PutSinkPolicy",
                "oam:TagResource"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadOnly",
            "Effect": "Allow",
            "Action": ["oam:Get*", "oam:List*"],
            "Resource": "*"
        }
    ]
}
```

- Akun sumber, dengan cakupan ke akun pemantauan tertentu – Untuk membuat, memperbarui, dan mengelola tautan hanya untuk satu akun pemantauan tertentu, Anda harus masuk ke akun dengan setidaknya izin berikut. Dalam contoh ini, akun pemantauan adalah 999999999999.

  Jika tautan tidak akan membagikan keempat jenis sumber daya (metrik, log, jejak, dan aplikasi Wawasan Aplikasi), Anda dapat menghilangkan,`cloudwatch:Link`, `logs:Link`, `xray:Link`, atau `applicationinsights:Link` sesuai kebutuhan.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Action": [
                "oam:CreateLink",
                "oam:UpdateLink",
                "oam:DeleteLink",
                "oam:GetLink",
                "oam:TagResource"
            ],
            "Effect": "Allow",
            "Resource": "arn:*:oam:*:*:link/*"
        },
        {
            "Action": [
                "oam:CreateLink",
                "oam:UpdateLink"
            ],
            "Effect": "Allow",
            "Resource": "arn:*:oam:*:*:sink/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": [
                        "999999999999"
                    ]
                }
            }
        },
        {
            "Action": "oam:ListLinks",
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "cloudwatch:Link",
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "logs:Link",
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "xray:Link",
            "Effect": "Allow",
```

```
            "Resource": "*"
        },
        {
             "Action": "applicationinsights:Link",
             "Effect": "Allow",
             "Resource": "*"
         }
    ]
}
```

- Akun sumber, dengan izin untuk menautkan ke akun pemantauan apa pun – Untuk membuat tautan ke sink akun pemantauan yang ada dan berbagi metrik, grup log, jejak, dan aplikasi Wawasan Aplikasi, Anda harus masuk ke akun sumber dengan izin administrator penuh atau masuk ke sana dengan izin berikut

  Jika tautan tidak akan membagikan keempat jenis sumber daya (metrik, log, jejak, dan aplikasi Wawasan Aplikasi), Anda dapat menghilangkan,`cloudwatch:Link`, `logs:Link`, `xray:Link`, atau `applicationinsights:Link` sesuai kebutuhan.

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "oam:CreateLink",
                "oam:UpdateLink"
            ],
            "Resource": [
                "arn:aws:oam:*:*:link/*",
                "arn:aws:oam:*:*:sink/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:List*",
                "oam:Get*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "oam:DeleteLink",
                "oam:GetLink",
                "oam:TagResource"
            ],
            "Resource": "arn:aws:oam:*:*:link/*"
        },
        {
            "Action": "cloudwatch:Link",
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "xray:Link",
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "logs:Link",
            "Effect": "Allow",
            "Resource": "*"
        },
        {
             "Action": "applicationinsights:Link",
             "Effect": "Allow",
             "Resource": "*"
        }
    ]
}
```

Gambaran umum pengaturan

Langkah-langkah tingkat tinggi berikut menunjukkan kepada Anda cara mengatur observabilitas CloudWatch lintas akun.

> **Note**
>
> Sebaiknya buat AWS akun baru untuk digunakan sebagai akun pemantauan organisasi Anda.

1. Menyiapkan akun pemantauan khusus.

2. (Opsional) Unduh AWS CloudFormation templat atau salin URL untuk menautkan akun sumber.
3. Menautkan akun sumber ke akun pemantauan.

Setelah menyelesaikan langkah-langkah ini, Anda dapat menggunakan akun pemantauan untuk melihat data observabilitas akun sumber.

Langkah 1: Menyiapkan akun pemantauan

Ikuti langkah-langkah di bagian ini untuk menyiapkan akun sebagai AWS akun pemantauan untuk pengamatan CloudWatch lintas akun.

Prasyarat

- Jika Anda menyiapkan akun di AWS Organizations organisasi sebagai akun sumber — Dapatkan jalur organisasi atau ID organisasi.
- Jika Anda tidak menggunakan Organizations untuk akun sumber – Dapatkan ID akun dari akun sumber.

Untuk mengatur akun sebagai akun pemantauan, Anda harus memiliki izin tertentu. Untuk informasi selengkapnya, lihat Izin yang diperlukan.

Untuk mengatur akun pemantauan

1. Masuk ke akun yang ingin Anda gunakan sebagai akun pemantauan.
2. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
3. Pada panel navigasi kiri, pilih Pengaturan.
4. Dengan Memantau konfigurasi akun, pilih Konfigurasi.
5. Untuk Memilih data, pilih apakah akun pemantauan ini akan dapat melihat data Log, Metrik, Jejak, dan Wawasan Aplikasi - aplikasi dari akun sumber yang ditautkan.
6. Untuk Membua daftar akun sumber, masukkan akun sumber yang akan ditampilkan oleh akun pemantauan ini. Untuk mengidentifikasi akun sumber, masukkan ID akun individual, jalur organisasi, atau ID organisasi. Jika Anda memasukkan jalur organisasi atau ID organisasi, akun pemantauan ini diizinkan untuk melihat data observabilitas dari semua akun terkait di organisasi tersebut.

   Pisahkan entri dalam daftar ini dengan koma.

> **Important**
>
> Saat Anda memasukkan jalur organisasi, ikuti format yang tepat. Ou-id harus diakhiri dengan / (karakter garis miring). Sebagai contoh: `o-a1b2c3d4e5/r-f6g7h8i9j0example/ou-def0-awsbbbb/`

7. Untuk Menentukan label untuk mengidentifikasi akun sumber Anda, tentukan apakah akan menggunakan nama akun atau alamat email untuk mengidentifikasi akun sumber saat Anda menggunakan akun pemantauan untuk melihatnya.
8. Pilih Konfigurasikan.

> **Important**
>
> Tautan antara pemantauan dan akun sumber tidak lengkap sampai Anda mengkonfigurasi akun sumber. Untuk informasi selengkapnya, silakan lihat bagian-bagian berikut ini.

Langkah 2: (Opsional) Unduh AWS CloudFormation templat atau URL

Untuk menautkan akun sumber ke akun pemantauan, sebaiknya gunakan template AWS CloudFormation atau URL.

- Jika Anda menautkan seluruh organisasi — CloudWatch berikan AWS CloudFormation templat.
- Jika Anda menautkan akun individual — Gunakan AWS CloudFormation templat atau URL yang CloudWatch menyediakan.

Untuk menggunakan AWS CloudFormation templat, Anda harus mengunduhnya selama langkah-langkah ini. Setelah Anda menautkan akun pemantauan dengan setidaknya satu akun sumber, AWS CloudFormation templat tidak lagi tersedia untuk diunduh.

Untuk mengunduh AWS CloudFormation templat atau menyalin URL untuk menautkan akun sumber ke akun pemantauan

1. Masuk ke akun yang ingin Anda gunakan sebagai akun pemantauan.
2. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
3. Pada panel navigasi kiri, pilih Pengaturan.

4. Dengan Memantau konfigurasi akun, pilih Sumber Daya untuk menautkan akun.
5. Lakukan salah satu langkah berikut:
   - Pilih organisasi AWS untuk mendapatkan template yang akan digunakan untuk menautkan akun di organisasi ke akun pemantauan ini.
   - Pilih Akun apa saja untuk mendapatkan template atau URL untuk menyiapkan akun individual sebagai akun sumber.
6. Lakukan salah satu hal berikut:
   - Jika Anda memilih AWS organisasi, pilih Unduh CloudFormation templat.
   - Jika Anda memilih Akun apa pun, pilih Unduh CloudFormation templat atau Salin URL.
7. (Opsional) Ulangi langkah 5-6 untuk mengunduh AWS CloudFormation templat dan URL.

Langkah 3: Menautkan akun sumber

Gunakan langkah-langkah di bagian ini untuk menautkan akun sumber ke akun pemantauan.

Untuk menautkan akun pemantauan dengan akun sumber, Anda harus memiliki izin tertentu. Untuk informasi selengkapnya, lihat Izin yang diperlukan.

Menggunakan template AWS CloudFormation untuk menyiapkan semua akun di organisasi atau unit organisasi sebagai akun sumber

Langkah-langkah ini mengasumsikan bahwa Anda sudah mengunduh AWS CloudFormation templat yang diperlukan dengan melakukan langkah-langkah masukLangkah 2: (Opsional) Unduh AWS CloudFormation templat atau URL.

Untuk menggunakan AWS CloudFormation templat untuk menautkan akun di organisasi atau unit organisasi ke akun pemantauan

1. Masuk ke akun manajemen organisasi.
2. Buka AWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation.
3. Di bilah navigasi kiri, pilih StackSets.
4. Periksa apakah Anda masuk ke Wilayah yang Anda inginkan, lalu pilih Buat StackSet.
5. Pilih Berikutnya.
6. Pilih Template sudah siap dan pilih Upload file template.

7. Pilih Pilih file, pilih template yang Anda unduh dari akun pemantauan, dan pilih Buka.
8. Pilih Berikutnya.
9. Untuk Tentukan StackSet detail, masukkan nama untuk StackSet dan pilih Berikutnya.
10. Untuk Menambahkan stack ke set stack, pilih Terapkan stack baru.
11. Untuk target Deployment, pilih apakah akan menyebarkan ke seluruh organisasi atau ke unit organisasi tertentu.
12. Untuk Menentukan wilayah, pilih Wilayah mana yang akan digunakan CloudWatch observabilitas lintas akun.
13. Pilih Berikutnya.
14. Pada halaman Ulasan, konfirmasikan opsi yang Anda pilih dan pilih Kirim.
15. Di tab Instans stack, segarkan layar hingga Anda melihat bahwa instance tumpukan Anda memiliki status CREATE_COMPLETE.

Gunakan template AWS CloudFormation untuk menyiapkan akun sumber individual

Langkah-langkah ini mengasumsikan bahwa Anda sudah mengunduh AWS CloudFormation templat yang diperlukan dengan melakukan langkah-langkah masukLangkah 2: (Opsional) Unduh AWS CloudFormation templat atau URL.

Untuk menggunakan AWS CloudFormation template untuk menyiapkan akun sumber individual untuk observabilitas CloudWatch lintas akun

1. Masuk ke akun sumber.
2. Buka AWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation.
3. Di bilah navigasi kiri, pilih Stack.
4. Periksa apakah Anda masuk ke Wilayah yang Anda inginkan, lalu pilih Buat tumpukan, Dengan sumber daya baru (standar).
5. Pilih Berikutnya.
6. Pilih Mengunggah file template.
7. Pilih Pilih file, pilih template yang Anda unduh dari akun pemantauan, dan pilih Buka.
8. Pilih Berikutnya.
9. Untuk Tentukan detail stack, masukkan nama untuk tumpukan dan pilih Berikutnya.
10. Pada halaman Konfigurasikan opsi stack, pilih Berikutnya.

11. Pada halaman Ulasan, pilih Kirim.
12. Pada halaman status untuk stack Anda, segarkan layar hingga Anda melihat bahwa stack Anda memiliki status CREATE_COMPLETE.
13. Untuk menggunakan template yang sama ini untuk menautkan lebih banyak akun sumber ke akun pemantauan ini, keluar dari akun ini dan masuk ke akun sumber berikutnya. Kemudian ulangi langkah 2-12.

Menggunakan URL untuk menyiapkan akun sumber individual

Langkah-langkah ini mengasumsikan bahwa Anda sudah menyalin URL yang diperlukan dengan melakukan langkah-langkah di Langkah 2: (Opsional) Unduh AWS CloudFormation templat atau URL.

Untuk menggunakan URL untuk menautkan akun sumber individual ke akun pemantauan

1. Masuk ke akun yang ingin Anda gunakan sebagai akun sumber.
2. Masukkan URL yang Anda salin dari akun pemantauan.

   Anda melihat halaman CloudWatch pengaturan, dengan beberapa informasi yang diisi.
3. Untuk Pilih data, pilih apakah akun sumber ini akan membagikan data Log, Metrik, Jejak, dan Wawasan Aplikasi - Aplikasi ke akun pemantauan ini.
4. Jangan mengubah ARN di Masukkan ARN konfigurasi akun pemantauan.
5. Bagian Tentukan label untuk mengidentifikasi akun sumber Anda sudah diisi sebelumnya dengan pilihan label dari akun pemantauan. Secara opsional, pilih Sunting untuk mengubahnya.
6. Pilih Tautkan.
7. Masukkan **`Confirm`** di kotak dan pilih Konfirmasi.
8. Untuk menggunakan URL yang sama ini untuk menautkan lebih banyak akun sumber ke akun pemantauan ini, keluar dari akun ini dan masuk ke akun sumber berikutnya. Kemudian ulangi langkah 2-7.

Mengelola akun pemantauan dan akun sumber

Setelah menyiapkan akun pemantauan dan akun sumber, Anda dapat menggunakan langkah-langkah di bagian ini untuk mengelolanya.

Daftar Isi

- Tautkan lebih banyak akun sumber ke akun pemantauan yang sudah ada
- Hapus tautan antara akun pemantauan dan akun sumber
- Melihat informasi tentang akun pemantauan

Tautkan lebih banyak akun sumber ke akun pemantauan yang sudah ada

Ikuti langkah-langkah di bagian ini untuk menambahkan tautan dari akun sumber tambahan ke akun pemantauan yang ada.

Setiap akun sumber dapat ditautkan ke sebanyak lima akun pemantauan. Setiap akun pemantauan dapat ditautkan ke sebanyak 100.000 akun sumber.

Untuk mengelola akun sumber, Anda harus memiliki izin tertentu. Untuk informasi selengkapnya, lihat Izin yang diperlukan.

Untuk menambahkan lebih banyak akun sumber ke akun pemantauan

1. Masuk ke akun pemantauan.
2. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
3. Pada panel navigasi kiri, pilih Pengaturan.
4. Dengan Memantau konfigurasi akun, pilih Kelola akun sumber.
5. Pilih tab Kebijakan konfigurasi.
6. Di kotak Kebijakan konfigurasi, tambahkan ID akun sumber baru di baris Pengguna Utama.

   Misalnya, baris Pengguna Utama saat ini adalah sebagai berikut:

   ```
   "Principal": {"AWS": ["111111111111", "222222222222"]}
   ```

   Untuk menambahkan 999999999999 sebagai akun sumber ketiga, sunting baris menjadi berikut:

   ```
   "Principal": {"AWS": ["111111111111", "222222222222", "999999999999"]}
   ```
7. Pilih Perbarui.
8. Pilih tab Detail konfigurasi.
9. Pilih ikon salin yang ada di sebelah ARN sink akun pemantauan.

10. Masuk ke akun yang ingin Anda gunakan sebagai akun sumber baru.
11. Rekatkan ARN wastafel akun pemantauan yang Anda salin di Langkah 9.

    Anda melihat halaman CloudWatch pengaturan, dengan beberapa informasi yang diisi.
12. Untuk Pilih data, pilih apakah akun sumber ini akan mengirim data Log, Metrik, Jejak, dan Wawasan Aplikasi - aplikasi ke akun pemantauan yang ditautkan.
13. Jangan mengubah ARN di Masukkan ARN konfigurasi akun pemantauan.
14. Bagian Tentukan label untuk mengidentifikasi akun sumber Anda sudah diisi sebelumnya dengan pilihan label dari akun pemantauan. Secara opsional, pilih Sunting untuk mengubahnya.
15. Pilih Tautkan.
16. Masukkan **`Confirm`** di kotak dan pilih Konfirmasi.

Hapus tautan antara akun pemantauan dan akun sumber

Ikuti langkah-langkah di bagian ini untuk menghentikan pengiriman data dari satu akun sumber ke akun pemantauan.

Anda harus memiliki izin yang diperlukan untuk mengelola akun sumber untuk menyelesaikan tugas ini. Untuk informasi selengkapnya, lihat Izin yang diperlukan.

Untuk menghapus tautan antara akun sumber dan akun pemantauan

1. Masuk ke akun sumber.
2. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
3. Pada panel navigasi kiri, pilih Pengaturan.
4. Berdasarkan informasi akun Sumber, pilih Lihat akun pemantauan.
5. Pilih kotak centang di samping akun pemantauan yang ingin Anda hentikan berbagi data.
6. Pilih Berhenti berbagi data, Konfirmasi.
7. Masuk ke akun pemantauan.
8. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
9. Pilih Pengaturan.
10. Dengan Memantau informasi akun, pilih Lihat konfigurasi.
11. Di kotak Kebijakan, hapus ID akun sumber dari baris Pengguna Utama dan pilih Perbarui.

Melihat informasi tentang akun pemantauan

Ikuti langkah-langkah di bagian ini untuk melihat pengaturan lintas akun untuk akun pemantauan.

Untuk mengelola akun pemantauan, Anda harus memiliki izin tertentu. Untuk informasi selengkapnya, lihat Izin yang diperlukan.

Untuk mengelola akun pemantauan

1. Masuk ke akun pemantauan.
2. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
3. Pada panel navigasi kiri, pilih Pengaturan.
4. Dengan Memantau konfigurasi akun, pilih Kelola akun sumber.
5. Untuk melihat kebijakan Pengelola Akses Observabilitas yang memungkinkan akun ini menjadi akun pemantauan, pilih tab Kebijakan konfigurasi.
6. Untuk melihat akun sumber yang ditautkan ke akun pemantauan ini, pilih tab Akun sumber terkait.
7. Untuk melihat ARN sink akun pemantauan, dan jenis data yang dapat dilihat akun pemantauan ini di akun sumber terkait, pilih tab Akun sumber terkait.

Metrik kueri dari sumber data lain

Anda dapat menggunakan CloudWatch kueri, memvisualisasikan, dan membuat alarm untuk metrik dari sumber data lain. Untuk melakukannya, Anda terhubung CloudWatch ke sumber data lain. Ini memberi Anda satu pengalaman pemantauan terkonsolidasi dalam CloudWatch konsol. Anda dapat memiliki sebuah tampilan terpadu dari infrastruktur dan metrik aplikasi Anda di mana pun data disimpan, membantu Anda mengidentifikasi dan menyelesaikan masalah dengan lebih cepat.

Setelah Anda terhubung ke sumber data menggunakan CloudWatch wizard, CloudWatch buat AWS CloudFormation tumpukan yang menyebarkan dan mengonfigurasi fungsi. AWS Lambda Fungsi Lambda ini berjalan sesuai permintaan setiap kali Anda menjalankan sumber data tersebut. Pembuat CloudWatch kueri menunjukkan kepada Anda secara real time daftar elemen yang dapat ditanyakan, seperti metrik, tabel, bidang, atau label. Saat Anda membuat pilihan, ppara embuat kueri akan mengisi kueri dalam bahasa asli dari sumber yang dipilih.

CloudWatch menyediakan wizard terpandu bagi Anda untuk terhubung ke sumber data berikut. Untuk sumber data ini, berikan informasi dasar untuk mengidentifikasi sumber data dan kredensialnya. Anda juga dapat membuat konektor secara manual ke sumber data lain dengan membuat fungsi Lambda Anda sendiri.

- OpenSearch Layanan Amazon- Dapatkan metrik dari log dan jejak OpenSearch Layanan Anda.
- Layanan Terkelola Amazon untuk Prometheus– Jalankan kueri metrik ini menggunakan PromQL.
- Amazon RDS for MySQL – Gunakan SQL untuk mengonversi data yang disimpan di tabel Amazon RDS Anda menjadi metrik.
- Amazon RDS for PostgreSQL – Gunakan SQL untuk mengonversi data yang disimpan di tabel Amazon RDS Anda menjadi metrik.
- Berkas CSV Amazon S3– Menampilkan data metrik dari sebuah berkas CSV yang disimpan di bucket Amazon S3.
- Monitor Microsoft Azure– Menjalankan kueri metrik dari akun Monitor Microsoft Azure Anda.
- Prometheus– Jalankan kueri metrik ini menggunakan PromQL.

Setelah Anda membuat konektor ke sumber data, silakan lihat Membuat sebuah grafik metrik dari sumber data lain untuk mendapat informasi tentang pembuatan grafik metrik dari sebuah sumber data. Untuk informasi tentang menyetel alarm pada metrik dari sebuah sumber data, silakan lihat Membuat sebuah alarm berdasarkan pada sumber data yang terhubung.

Topik

Mengelola akses ke sumber data

CloudWatch digunakan AWS CloudFormation untuk membuat sumber daya yang diperlukan di akun Anda. Kami menyarankan Anda menggunakan `cloudformation:TemplateUrl` kondisi ini untuk mengontrol akses ke AWS CloudFormation templat saat Anda memberikan `CreateStack` izin kepada pengguna IAM.

> **Warning**
>
> Setiap pengguna yang mendapatkan izin invokasi sumber data dari Anda dapat menjalankan kueri terhadap metrik dari sumber data tersebut meskipun pengguna tersebut tidak memiliki izin IAM langsung ke sumber data tersebut. Sebagai contoh, jika Anda memberikan izin `lambda:InvokeFunction` pada sebuah Layanan Terkelola Amazon untuk fungsi Lambda sumber data Prometheus kepada seorang pengguna, pengguna tersebut akan dapat menjalankan kueri terhadap metrik dari Layanan Terkelola Amazon yang sesuai untuk workspace Prometheus meskipun Anda tidak memberikan akses IAM langsung ke workspace tersebut kepada mereka.

Anda dapat menemukan URL templat untuk sumber data di halaman Buat tumpukan di Konsol CloudWatch Pengaturan.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect" : "Allow",
      "Action" : [ "cloudformation:CreateStack" ],
      "Resource" : "*",
```

```
      "Condition" : {
        "StringEquals" : {
          "cloudformation:TemplateUrl" : [ data-source-template-url ]
        }
      }
    }
  ]
}
```

Untuk informasi selengkapnya tentang mengontrol AWS CloudFormation akses, lihat Mengontrol akses dengan AWS Identity and Access Management

Hubungkan ke sumber data default dengan sebuah pemandu

Topik ini memberikan instruksi untuk menggunakan wizard untuk terhubung CloudWatch ke sumber data berikut.

- OpenSearch Layanan Amazon
- Layanan Terkelola Amazon untuk Prometheus
- Amazon RDS for MySQL
- Amazon RDS for PostgreSQL
- Berkas CSV Amazon S3
- Monitor Microsoft Azure
- Prometheus

Kemudian di bagian ini adalah subbagian dengan catatan tentang mengelola dan menjalankan kueri dengan masing-masing sumber data ini.

Untuk membuat sebuah konektor ke sumber data

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Pengaturan.
3. Pilih tab Sumber data metrik.
4. Pilih Buat sumber data.
5. Pilih sumber yang Anda inginkan, lalu pilih Berikutnya.

6. Masukkan sebuah nama untuk sumber data.
7. Masukkan informasi lain yang diperlukan, tergantung pada sumber data yang Anda pilih. Ini dapat mencakup kredensial untuk mengakses sumber data tersebut dan sumber data yang mengidentifikasi informasi seperti nama workspace Prometheus, nama basis data, atau nama bucket Amazon S3. Untuk AWS layanan, wizard menemukan sumber daya dan mengisinya ke dalam pilihan dropdown.

   Untuk catatan lebih lanjut tentang sumber data yang Anda gunakan, silakan lihat bagian setelah prosedur ini.
8. Untuk CloudWatch terhubung ke sumber data dalam VPC, pilih Gunakan VPC dan pilih VPC yang akan digunakan. Kemudian pilih subnet dan grup keamanan.
9. Pilih Saya mengakui yang AWS CloudFormation akan membuat sumber daya IAM. Sumber daya ini adalah peran eksekusi fungsi Lambda.
10. Pilih Buat sumber data.

    Sumber baru yang baru saja Anda tambahkan tidak muncul sampai AWS CloudFormation tumpukan selesai membuatnya. Untuk memeriksa kemajuan, Anda dapat memilih Lihat status CloudFormation tumpukan saya. Atau Anda dapat memilih ikon muat ulang untuk memperbarui daftar ini.

    Ketika sumber data baru Anda muncul dalam daftar ini, sumber data siap digunakan. Anda dapat memilih Kueri dari CloudWatch metrik untuk memulai kueri dengannya. Untuk informasi selengkapnya, lihat Membuat sebuah grafik metrik dari sumber data lain.

Layanan Terkelola Amazon untuk Prometheus

Memperbarui konfigurasi sumber data

- Anda dapat memperbarui sumber data Anda secara manual dengan melakukan hal berikut:
  - Untuk memperbarui ID workspace Layanan Terkelola Amazon untuk Prometheus, perbarui variabel lingkungan `AMAZON_PROMETHEUS_WORKSPACE_ID` untuk sumber data konektor fungsi Lambda.
  - Untuk memperbarui konfigurasi VPC, silakan lihat Mengonfigurasi akses VPC (konsol) untuk informasi selengkapnya.

Menjalankan kueri sumber data

- Saat menjalankan kueri Layanan Terkelola Amazon untuk Prometheus, setelah Anda memilih sumber datanya di tab kueri multi sumber dan memilih sebuah konektor Layanan Terkelola Amazon untuk Prometheus, Anda dapat menggunakan alat bantu Kueri untuk menemukan metrik dan melabelinya dan menyediakan kueri PromQL yang sederhana. Anda juga dapat menggunakan editor kueri PromQL untuk membangun sebuah kueri PromQL.
- Kueri multi-baris tidak didukung oleh konektor sumber CloudWatch data. Setiap umpan baris digantikan dengan sebuah ruang saat kueri dijalankan, atau saat Anda membuat sebuah alarm atau widget dasbor dengan kuerinya. Dalam beberapa kasus, ini mungkin membuat kueri Anda tidak valid. Sebagai contoh, jika kueri Anda berisikan sebuah komentar satu baris, itu akan menjadi tidak valid. Jika Anda mencoba membuat dasbor atau alarm dengan kueri multi-baris dari baris perintah atau Infrastruktur sebagai Kode, API akan menolak tindakan tersebut dengan kesalahan penguraian.

OpenSearch Layanan Amazon

Membuat sumber data

Anda harus memetakan peran eksekusi fungsi konektor Lambda ke pengguna di OpenSearch Layanan. Untuk informasi selengkapnya, lihat bagian Memetakan pengguna ke peran di Mengelola izin di dokumentasi OpenSearch Layanan.

Memperbarui sumber data

- Anda dapat memperbarui sumber data Anda secara manual dengan melakukan hal berikut:
  - Untuk memperbarui domain OpenSearch Layanan, perbarui variabel `AMAZON_OPENSEARCH_DOMAIN_NAME` lingkungan untuk fungsi Lambda konektor sumber data.
  - Untuk memperbarui konfigurasi VPC, silakan lihat Mengonfigurasi akses VPC (konsol) untuk informasi selengkapnya.

Menjalankan kueri sumber data

- Saat menanyakan OpenSearch Layanan, setelah Anda memilih sumber data di tab Kueri multi sumber, lakukan hal berikut:
  - Pilih Indeks untuk kueri.
  - Pilih Nama metrik (Bidang numerik apa pun dalam dokumen) dan Stat.
  - Pilih sumbu Waktu (Bidang tanggal apa pun dalam dokumen).

- Pilih Filter untuk diterapkan (Bidang String apa pun dalam dokumen).
- Pilih kueri Grafik.

Amazon RDS for PostgreSQL dan Amazon RDS for MySQL

Membuat sumber data

- Jika sumber data Anda hanya dapat diakses dalam sebuah VPC, Anda harus memasukkan konfigurasi VPC untuk konektor, seperti yang dijelaskan dalam Hubungkan ke sumber data default dengan sebuah pemandu. Jika sumber data dihubungkan ke VPC untuk kredensial, titik akhirnya harus terkonfigurasi di VPC. Untuk informasi selengkapnya, lihat Menggunakan titik AWS Secrets Manager akhir VPC.

Memperbarui sumber data

- Anda dapat memperbarui sumber data Anda secara manual dengan melakukan hal berikut:
  - Untuk memperbarui instans basis data, perbarui variabel lingkungan `RDS_INSTANCE` untuk sumber data konektor fungsi Lambda.
  - Untuk memperbarui nama pengguna dan kata sandi yang digunakan untuk terhubung ke Amazon RDS, gunakan AWS Secrets Manager. Anda dapat menemukan ARN dari yang rahasia yang digunakan untuk sumber data dalam variabel lingkungan `RDS_SECRET` pada sumber data fungsi Lambda. Untuk informasi selengkapnya tentang memperbarui rahasia di AWS Secrets Manager, silakan lihat Memodifikasi rahasia AWS Secrets Manager.
  - Untuk memperbarui konfigurasi VPC, silakan lihat Mengonfigurasi akses VPC (konsol) untuk informasi selengkapnya.

Menjalankan kueri sumber data

- Saat menjalankan kueri Amazon RDS, setelah memilih sumber data di tab Kueri multi sumber dan memilih sebuah konektor Amazon RDS, Anda dapat menggunakan penjelajah basis data untuk melihat basis data, tabel, dan kolom yang tersedia. Anda juga dapat menggunakan editor SQL untuk membuat sebuah kueri SQL.

  Anda dapat menggunakan variabel berikut dalam kueri:
  - `$start.iso` – Waktu mulai dalam format tanggal ISO
  - `$end.iso` – Waktu berakhir dalam format tanggal ISO

- `$period` – Periode yang dipilih dalam hitungan detik

  Misalnya, Anda dapat menjalankan kueri `SELECT value, timestamp FROM table WHERE timestamp BETWEEN $start.iso and $end.iso`

- Kueri multi-baris tidak didukung oleh konektor sumber CloudWatch data. Setiap umpan baris digantikan dengan sebuah ruang saat kueri dijalankan, atau saat Anda membuat sebuah alarm atau widget dasbor dengan kuerinya. Dalam beberapa kasus, ini mungkin membuat kueri Anda tidak valid. Sebagai contoh, jika kueri Anda berisikan sebuah komentar satu baris, itu akan menjadi tidak valid. Jika Anda mencoba membuat dasbor atau alarm dengan kueri multi-baris dari baris perintah atau Infrastruktur sebagai Kode, API akan menolak tindakan tersebut dengan kesalahan penguraian.

> **Note**
>
> Jika tidak ada bidang tanggal yang ditemukan dalam hasil, nilai untuk setiap bidang numerik akan dijumlahkan ke nilai tunggal dan diplot di seluruh rentang waktu yang tersedia. Jika stempel waktu tidak sejajar dengan periode yang dipilih CloudWatch, data secara otomatis digabungkan menggunakan `SUM` dan disejajarkan dengan periode di. CloudWatch

Berkas CSV Amazon S3

Menjalankan kueri sumber data

- Saat menjalankan kueri berkas CSV Amazon S3, setelah Anda memilih sumber data di tab Kueri multi sumber dan memilih sebuah konektor Amazon S3, pilih bucket dan kunci Amazon S3.

  File CSV harus diformat dengan cara berikut:

  - Cap waktu harus kolom pertama.
  - Tabel harus memiliki baris header. Header digunakan untuk memberi nama metrik Anda. Judul kolom stempel waktu akan diabaikan, hanya judul kolom metrik yang digunakan.
  - Perangko waktu harus dalam format tanggal ISO.
  - Metrik harus berupa bidang numerik.

```
Timestamp, Metric-1, Metric-2, ...
```

Berikut ini adalah contohnya:

stempel waktu	(%) CPU	Memori (%)	Penyimpanan (%)
2023-11-23T17:09:41+00:00	1	2	3
2023-11-23T17:04:41+00:00	4	5	6
2023-11-23T16:59:41+00:00	7	8	9
2023-11-23T16:54:41+00:00	10	11	12

> **Note**
>
> Jika tidak ada stempel waktu yang diberikan, nilai untuk setiap metrik akan dijumlahkan ke nilai-nilai tunggal dan diplot di seluruh rentang waktu yang disediakan. Jika stempel waktu tidak sejajar dengan periode yang dipilih CloudWatch, data secara otomatis digabungkan menggunakan `SUM` dan disejajarkan dengan periode di. CloudWatch

Monitor Microsoft Azure

Membuat sumber data

- Anda harus memberikan ID penyewa, ID klien, dan rahasia klien Anda untuk terhubung ke Monitor Microsoft Azure. Kredensialnya akan disimpan di. AWS Secrets Manager Untuk informasi selengkapnya, silakan lihat Membuat sebuah aplikasi Microsoft Entra dan pengguna utama layanan yang dapat mengakses sumber daya dalam dokumentasi Microsoft.

Memperbarui sumber data

- Anda dapat memperbarui sumber data Anda secara manual dengan melakukan hal berikut:

- Untuk memperbarui ID penyewa, ID klien, dan rahasia klien yang digunakan untuk terhubung ke Azure Monitor, Anda dapat menemukan ARN yang rahasianya digunakan untuk sumber data sebagai variabel lingkungan `AZURE_CLIENT_SECRET` pada sumber data fungsi Lambda. Untuk informasi selengkapnya tentang memperbarui rahasia AWS Secrets Manager, lihat Memodifikasi AWS Secrets Manager rahasia.

Menjalankan kueri sumber data

- Saat menjalankan kueri Azure Monitor, setelah Anda memilih sumber data di tab Kueri multi sumber dan memilih sebuah konektor Azure Monitor, tentukan langganan Azure, dan grup sumber daya serta sumber daya. Anda kemudian dapat memilih namespace metrik, metrik, dan agregasi, dan memfilter berdasarkan dimensi.

Prometheus

Membuat sumber data

- Anda harus memberikan titik akhir Prometheus dan pengguna serta kata sandi yang diperlukan untuk menjalankan kueri Prometheus. Kredensialnya akan disimpan di. AWS Secrets Manager
- Jika sumber data Anda hanya dapat diakses dalam sebuah VPC, Anda harus memasukkan konfigurasi VPC untuk konektor, seperti yang dijelaskan dalam Hubungkan ke sumber data default dengan sebuah pemandu. Jika sumber data adalah untuk menghubungkan untuk kredensial, titik akhir harus dikonfigurasi dalam VPC. Untuk informasi selengkapnya, lihat Menggunakan titik AWS Secrets Manager akhir VPC.

Memperbarui konfigurasi sumber data

- Anda dapat memperbarui sumber data Anda secara manual dengan melakukan hal berikut:
  - Untuk memperbarui titik akhir Prometheus, tentukan titik akhir yang baru sebagai variabel lingkungan `PROMETHEUS_API_ENDPOINT` pada sumber data fungsi Lambda.
  - Untuk memperbarui nama pengguna dan kata sandi yang digunakan untuk terhubung ke Prometheus, Anda dapat menemukan ARN dari rahasia yang digunakan untuk sumber data sebagai variabel lingkungan `PROMETHEUS_API_SECRET` pada sumber data fungsi Lambda. Untuk informasi selengkapnya tentang memperbarui rahasia AWS Secrets Manager, lihat Memodifikasi AWS Secrets Manager rahasia.

- Untuk memperbarui konfigurasi VPC, silakan lihat Mengonfigurasi akses VPC (konsol) untuk informasi selengkapnya.

Menjalankan kueri sumber data

> **Important**
>
> Jenis metrik Prometheus berbeda CloudWatch dari metrik dan banyak metrik yang tersedia melalui Prometheus bersifat kumulatif berdasarkan desain. Saat Anda menanyakan metrik Prometheus CloudWatch , tidak menerapkan transformasi tambahan apa pun ke data: jika Anda hanya menentukan nama atau label metrik, nilai yang ditampilkan akan bersifat kumulatif. Untuk informasi selengkapnya, silakan lihat Jenis-jenis metrik dalam dokumentasi Prometheus.
>
> Untuk melihat data metrik Prometheus sebagai nilai diskrit, CloudWatch seperti metrik, Anda perlu mengedit kueri sebelum menjalankannya. Misalnya, Anda mungkin perlu menambahkan sebuah panggilan ke fungsi tarif di atas nama metrik Prometheus Anda. Untuk dokumentasi tentang fungsi rate dan fungsi Prometheus lainnya, silakan lihat rate () dalam dokumentasi Prometheus.

Kueri multi-baris tidak didukung oleh konektor sumber CloudWatch data. Setiap umpan baris digantikan dengan sebuah ruang saat kueri dijalankan, atau saat Anda membuat sebuah alarm atau widget dasbor dengan kuerinya. Dalam beberapa kasus, ini mungkin membuat kueri Anda tidak valid. Sebagai contoh, jika kueri Anda berisikan sebuah komentar satu baris, itu akan menjadi tidak valid. Jika Anda mencoba membuat dasbor atau alarm dengan kueri multi-baris dari baris perintah atau Infrastruktur sebagai Kode, API akan menolak tindakan tersebut dengan kesalahan penguraian.

Notifikasi Pembaruan yang Tersedia

Dari waktu ke waktu, Amazon mungkin akan memberi tahu Anda bahwa kami menyarankan Anda memperbarui konektor Anda dengan versi yang lebih baru yang tersedia dan akan memberikan instruksi tentang cara melakukannya.

Buatlah sebuah konektor kustom ke sebuah sumber data

Untuk menghubungkan sumber data kustom ke CloudWatch, Anda memiliki dua opsi:

- Memulai dengan menggunakan contoh template yang CloudWatch menyediakan. Anda dapat menggunakan salah satu JavaScript atau Python dengan template ini. Templat ini mencakup contoh kode Lambda yang akan berguna bagi Anda saat Anda membuat fungsi Lambda Anda. Anda kemudian dapat memodifikasi fungsi Lambda dari template untuk terhubung ke sumber data kustom Anda.
- Buat AWS Lambda fungsi dari awal yang mengimplementasikan konektor sumber data, kueri data, dan persiapan deret waktu untuk digunakan oleh CloudWatch. Fungsi ini harus melakukan pra-agregat atau menggabungkan titik data jika diperlukan, dan juga menyelaraskan periode dan stempel waktu agar kompatibel dengannya. CloudWatch

Daftar Isi

Gunakan sebuah templat

Menggunakan sebuah template untuk membuat sebuah contoh fungsi Lambda, dan membantu Anda membuat konektor kustom Anda dibangun lebih cepat. Fungsi sampel ini menyediakan kode sampel untuk banyak skenario umum yang terlibat dengan pembuatan konektor khusus. Anda dapat memeriksa kode Lambda setelah Anda membuat sebuah konektor dengan templat, lalu memodifikasinya untuk digunakan untuk terhubung ke sumber data Anda.

Selain itu, jika Anda menggunakan template, CloudWatch berhati-hatilah dalam membuat kebijakan izin Lambda dan melampirkan tag sumber daya ke fungsi Lambda.

Gunakan template untuk membuat sebuah konektor ke sebuah sumber data kustom.

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Pengaturan.
3. Pilih tab Sumber data metrik.
4. Pilih Buat sumber data.
5. Pilih tombol radio untuk Kustom - memulai template dan kemudian pilih Berikutnya.
6. Masukkan sebuah nama untuk sumber data.
7. Pilih salah satu template yang terdaftar.
8. Pilih Node.js atau Python.
9. Pilih Buat sumber data.

   Sumber kustom baru yang baru saja Anda tambahkan tidak muncul sampai AWS CloudFormation tumpukan selesai membuatnya. Untuk memeriksa kemajuan, Anda dapat memilih Lihat status CloudFormation tumpukan saya. Atau Anda dapat memilih ikon muat ulang untuk memperbarui daftar ini.

   Ketika sumber data baru Anda muncul dalam daftar ini, maka siap untuk diujicoba di konsol dan dimodifikasi.
10. (Opsional) Untuk menjalankan kueri data pengujian dari sumber ini di konsol, ikuti instruksi di Membuat sebuah grafik metrik dari sumber data lain.
11. Ubah fungsi Lambda untuk kebutuhan Anda.

    a. Pada panel navigasi, silakan pilih Pengaturan.

    b. Pilih tab Sumber data metrik.

    c. Pilih Lihat di konsol Lambda untuk sumber yang ingin Anda ubah.

    Sekarang Anda dapat memodifikasi fungsi untuk mengakses sumber data Anda. Untuk informasi selengkapnya, lihat Langkah 1: Membuat fungsi .

    > **Note**
    >
    > Dengan menggunakan templat, ketika Anda menulis fungsi Lambda Anda, Anda tidak perlu mengikuti instruksi di Langkah 2: Membuat Kebijakan Izin Lambda atau Langkah

3: Melampirkan sebuah tanda sumber daya ke fungsi Lambda. Langkah-langkah ini dilakukan oleh CloudWatch karena Anda menggunakan template.

Buat sebuah sumber data kustom dari awal

Ikuti langkah-langkah di bagian ini untuk membuat fungsi Lambda yang terhubung CloudWatch ke sumber data.

Langkah 1: Membuat fungsi

Konektor sumber data khusus harus mendukung `GetMetricData` peristiwa dari CloudWatch. Secara opsional, Anda juga dapat mengimplementasikan `DescribeGetMetricData` acara untuk memberikan dokumentasi kepada pengguna di CloudWatch konsol tentang cara menggunakan konektor. `DescribeGetMetricData`Respons juga dapat digunakan untuk mengatur default yang digunakan dalam pembuat kueri CloudWatch kustom.

CloudWatch menyediakan cuplikan kode sebagai contoh untuk membantu Anda memulai. Untuk informasi lebih lanjut, lihat repositori sampel di https://github.com/aws-samples/. cloudwatch-data-source-samples

Kekurangan

- Respons dari Lambda harus lebih kecil dari 6 Mb. Jika respons melebihi 6 Mb, respons `GetMetricData` akan menandai fungsi Lambda sebagai `InternalError` dan tidak ada data yang dikembalikan.
- Fungsi Lambda harus menyelesaikan pelaksanaan dalam waktu 10 detik untuk tujuan visualisasi dan dasbor, atau dalam 4,5 detik untuk penggunaan alarm. Jika respons melebihi itu, respons `GetMetricData` akan menandai fungsi Lambda sebagai `InternalError` dan tidak ada data yang dikembalikan.
- Fungsi Lambda harus mengirimkan hasil akhirnya menggunakan stempel waktu dalam masa hitungan detik.
- Jika fungsi Lambda tidak mengambil sampel ulang data dan sebaliknya mengembalikan data yang tidak sesuai dengan waktu mulai dan panjang periode yang diminta oleh CloudWatch pengguna, data tersebut akan diabaikan oleh. CloudWatch Data tambahan dibuang dari visualisasi atau pengaturan alarm apapun. Data apa pun yang tidak berada di antara waktu mulai dan waktu akhir juga dibuang.

Sebagai contoh, jika seorang pengguna meminta data dari pukul 10:00 hingga 11:00 dengan jangka waktu 5 menit, maka "10:00:00 hingga 10:04:59" dan "10:05:00 hingga 10:09:59" adalah rentang waktu yang valid untuk data yang akan dikembalikan. Anda harus mengembalikan sebuah deret waktu yang mencakup `10:00 value1`, `10:05 value2`, dan sebagainya. Jika fungsi tersebut mengembalikan `10:03 valueX`, misalnya, maka fungsi akan jatuh karena 10:03 tidak sesuai dengan waktu mulai dan periode yang diminta.

- Kueri multi-baris tidak didukung oleh konektor sumber CloudWatch data. Setiap umpan baris digantikan dengan sebuah ruang saat kueri dijalankan, atau saat Anda membuat sebuah alarm atau widget dasbor dengan kuerinya. Dalam beberapa kasus, ini mungkin membuat kueri Anda tidak valid.

GetMetricData acara

Meminta muatan

Berikut ini adalah contoh muatan permintaan `GetMetricData` yang dikirimkan sebagai input ke fungsi Lambda.

```
{
  "EventType": "GetMetricData",
  "GetMetricDataRequest": {
    "StartTime": 1697060700,
    "EndTime": 1697061600,
    "Period": 300,
    "Arguments": ["serviceregistry_external_http_requests{host_cluster!=\"prod\"}"]
  }
}
```

- StartTime— Stempel waktu yang menentukan data paling awal untuk dikembalikan. Jenis adalah stempel waktu masa detik.
- EndTime— Stempel waktu yang menentukan data terbaru yang akan dikembalikan. Jenis adalah stempel waktu masa detik.
- Periode – Jumlah detik yang diwakili oleh setiap kumpulan data metrik. Minimal adalah 60 detik. Jenis masa adalah detik.
- Argumen – Serangkaian argumen yang akan diteruskan ke ekspresi matematika metrik Lambda. Untuk informasi selengkapnya tentang argumen yang diteruskan, silakan lihat Cara meneruskan argumen ke fungsi Lambda Anda.

Muatan respons

Berikut ini adalah contoh muatan respons `GetMetricData` yang dikembalikan oleh fungsi Lambda.

```
{
   "MetricDataResults": [
      {
         "StatusCode": "Complete",
         "Label": "CPUUtilization",
         "Timestamps": [ 1697060700, 1697061000, 1697061300 ],
         "Values": [ 15000, 14000, 16000 ]
      }
   ]
}
```

Muatan respons ini akan berisi bidang `MetricDataResults` atau bidang `Error`, tetapi tidak keduanya.

Bidang `MetricDataResults` adalah daftar bidang-bidang deret waktu jenis `MetricDataResult`. Masing-masing bidang deret waktu tersebut dapat mencakup bidang-bidang berikut ini.

- StatusCode— (Opsional) `Complete` menunjukkan bahwa semua titik data dalam rentang waktu yang diminta dikembalikan. `PartialData`berarti bahwa satu set titik data yang tidak lengkap dikembalikan. Jika ini dihilangkan, defaultnya adalah `Complete`.

  Nilai yang valid: `Complete` | `InternalError` | `PartialData` | `Forbidden`
- Pesan – Daftar pesan opsional dengan informasi tambahan tentang data yang dikembalikan.

  Jenis: Array MessageDataobjek dengan `Code` dan `Value` string.
- Label – Label yang dapat dibaca manusia yang terkait dengan data.

  Tipe: String
- Stempel waktu – Stempel waktu untuk titik data, yang diformat dalam waktu masa. Jumlah stempel waktu selalu cocok dengan jumlah nilai dan nilai untuk `Timestamps[x]` adalah `Values[x]`.

  Tipe: Array stempel waktu
- Nilai–Nilai – Nilai titik data untuk metrik, yang sesuai dengan `Timestamps`. Jumlah nilai selalu cocok dengan jumlah stempel waktu dan nilai untuk `Timestamps[x]` adalah `Values[x]`.

  Tipe: Array ganda

Untuk informasi selengkapnya tentang objek `Error`, silakan lihat bagian-bagian berikut.

Format respons kesalahan

Anda dapat secara opsional menggunakan respons kesalahan untuk memberikan informasi lebih lanjut tentang kesalahan. Kami menyarankan Anda mengembalikan kesalahan dengan Validasi Kode ketika kesalahan validasi terjadi, seperti ketika sebuah parameter hilang atau jenis parameter salah.

Berikut ini adalah contoh respon ketika fungsi Lambda ingin memunculkan pengecualian validasi `GetMetricData`.

```
{
   "Error": {
      "Code": "Validation",
      "Value": "Invalid Prometheus cluster"
   }
}
```

Berikut ini adalah contoh respons ketika fungsi Lambda mengindikasikn bahwa fungsi Lambda tidak dapat mengembalikan data karena masalah akses. Respons diterjemahkan ke dalam satu deret waktu dengan kode status `Forbidden`.

```
{
   "Error": {
      "Code": "Forbidden",
      "Value": "Unable to access ..."
   }
}
```

Berikut ini adalah sebuah contoh ketika fungsi Lambda memunculkan pengecualian `InternalError` keseluruhan, yang diterjemahkan ke dalam sebuah rangkaian waktu tunggal dengan kode status `InternalError` dan sebuah pesan. Setiap kali kode kesalahan memiliki nilai selain `Validation` atau`Forbidden`, CloudWatch mengasumsikan bahwa itu adalah kesalahan internal generik.

```
{
   "Error": {
      "Code": "PrometheusClusterUnreachable",
      "Value": "Unable to communicate with the cluster"
   }
```

```
}
```

DescribeGetMetricData acara

Meminta muatan

Berikut ini adalah contoh muatan permintaan `DescribeGetMetricData`.

```
{
  "EventType": "DescribeGetMetricData"
}
```

Muatan respons

Berikut ini adalah contoh muatan respons `DescribeGetMetricData`.

```
{
    "Description": "Data source connector",
    "ArgumentDefaults": [{
        Value: "default value"
     }]
}
```

- Deskripsi – Sebuah deskripsi tentang cara menggunakan konektor sumber data. Deskripsi ini akan muncul di CloudWatch konsol. Markdown didukung.

  Jenis: String
- ArgumentDefaults— Array opsional nilai default argumen yang digunakan pra-mengisi pembuat sumber data kustom.

  Jika`[{ Value: "default value 1"}, { Value: 10}]`, dikembalikan, pembuat kueri di CloudWatch konsol menampilkan dua input, yang pertama dengan “nilai default 1" dan yang kedua dengan 10.

  Jika `ArgumentDefaults` tidak tersedia, satu input tunggal akan ditampilkan dengan jenis default yang disetel ke `String`.

  Tipe: Array objek yang mengandung Nilai dan Tipe.
- Kesalahan – (Opsional) Sebuah bidang kesalahan dapat dimasukkan dalam respons apa pun. Anda dapat melihat contoh di GetMetricData acara.

Pertimbangan penting untuk alarm CloudWatch

Jika Anda akan menggunakan sumber data untuk mengatur CloudWatch alarm, Anda harus mengaturnya untuk melaporkan data dengan stempel waktu setiap menit. CloudWatch Untuk informasi selengkapnya dan pertimbangan lain untuk membuat alarm pada metrik dari sumber data yang terhubung, silakan lihat Membuat sebuah alarm berdasarkan pada sumber data yang terhubung.

(Opsional) Gunakan AWS Secrets Manager untuk menyimpan kredensil

Jika fungsi Lambda Anda perlu menggunakan kredensyal untuk mengakses sumber data, sebaiknya gunakan untuk menyimpan kredensyal ini alih-alih mengkodekannya AWS Secrets Manager ke dalam fungsi Lambda Anda. Untuk informasi selengkapnya tentang penggunaan AWS Secrets Manager dengan Lambda, lihat Menggunakan AWS Secrets Manager rahasia dalam AWS Lambda fungsi.

(Opsional) Hubungkan ke sebuah sumber data dalam sebuah VPC

Jika sumber data Anda berada dalam sebuah VPC yang dikelola oleh Cloud Privat Virtual Amazon, Anda harus mengonfigurasi fungsi Lambda Anda untuk mengaksesnya. Untuk informasi selengkapnya, silakan lihat Menghubungkan jaringan keluar ke sumber daya di VPC.

Anda mungkin juga perlu mengonfigurasi titik akhir layanan VPC untuk mengakses layanan seperti AWS Secrets Manager. Untuk informasi selengkapnya, lihat Mengakses AWS layanan menggunakan titik akhir VPC antarmuka.

Langkah 2: Membuat Kebijakan Izin Lambda

Anda harus menggunakan buat pernyataan kebijakan yang memberikan CloudWatch izin untuk menggunakan fungsi Lambda yang Anda buat. Anda dapat menggunakan AWS CLI atau konsol Lambda untuk membuat pernyataan kebijakan.

Untuk menggunakan AWS CLI untuk membuat pernyataan kebijakan

- Masukkan perintah berikut. Ganti *123456789012* dengan ID akun Anda, ganti *my-data-source-function*dengan nama fungsi Lambda Anda, dan ganti *MyDataSource-DataSourcePermission* 1234 dengan nilai unik yang sewenang-wenang.

```
aws lambda add-permission --function-name my-data-source-function --statement-
id MyDataSource-DataSourcePermission1234 --action lambda:InvokeFunction --principal
 lambda.datasource.cloudwatch.amazonaws.com --source-account 123456789012
```

Langkah 3: Melampirkan sebuah tanda sumber daya ke fungsi Lambda

CloudWatch Konsol menentukan fungsi Lambda mana yang merupakan konektor sumber data dengan menggunakan tag. Saat Anda membuat sumber data menggunakan salah satu wizard, tag secara otomatis diterapkan oleh AWS CloudFormation tumpukan yang mengonfigurasinya. Saat Anda membuat sumber data sendiri, maka Anda dapat menggunakan tanda berikut untuk fungsi Lambda Anda. Ini membuat konektor Anda muncul di dropdown sumber data di CloudWatch konsol saat Anda melakukan kueri metrik.

- Tanda dengan `cloudwatch:datasource` sebagai kunci dan `custom` sebagai nilai.

Gunakan sumber data kustom Anda

Setelah membuat sebuah sumber data, Anda dapat menggunakannya untuk menjalankan kueri data dari sumber tersebut untuk memvisualisasikannya dan mengatur alarm. Jika Anda menggunakan template untuk membuat konektor sumber data kustom atau menambahkan tanda yang tercantum di Langkah 3: Melampirkan sebuah tanda sumber daya ke fungsi Lambda, Anda dapat mengikuti langkah-langkahnya di Membuat sebuah grafik metrik dari sumber data lain untuk menjalankan kuerinya.

Anda juga dapat menggunakan fungsi matematika metrik `LAMBDA` untuk menjalankan kueri, sebagaimana yang dijelaskan di bagian berikut.

Untuk informasi tentang pengaturan alarm pada metrik dari sumber data Anda, silakan lihat Membuat sebuah alarm berdasarkan pada sumber data yang terhubung.

Cara meneruskan argumen ke fungsi Lambda Anda

Cara yang disarankan bagi Anda untuk meneruskan argumen ke sumber data kustom Anda adalah dengan menggunakan pembuat kueri di CloudWatch konsol saat Anda menanyakan sumber data.

Anda juga dapat menggunakan fungsi Lambda untuk mengambil data dari sumber data Anda dengan menggunakan `LAMBDA` ekspresi baru dalam CloudWatch matematika metrik.

```
LAMBDA("LambdaFunctionName" [, optional-arg]*)
```

`optional-arg` hingga 20 string, angka, atau Booleans. Sebagai contoh, `param 3.14` atau `true`.

> **Note**
>
> String multi-baris tidak didukung oleh konektor sumber CloudWatch data. Setiap umpan baris digantikan dengan sebuah ruang saat kueri dijalankan, atau saat Anda membuat sebuah alarm atau widget dasbor dengan kuerinya. Dalam beberapa kasus, ini mungkin membuat kueri Anda tidak valid.

Bila Anda menggunakan fungsi matematika metrik `LAMBDA`, Anda dapat memberikan nama fungsi (`"MyFunction"`). Jika kebijakan sumber daya Anda memungkinkan, Anda juga dapat menggunakan versi fungsi khusus (`"MyFunction:22"`), atau nama lain fungsi Lambda (`"MyFunction:MyAlias"`). Anda tidak dapat menggunakan `*`

Berikut ini adalah beberapa contoh calling fungsi `LAMBDA`.

```
LAMBDA("AmazonOpenSearchDataSource", "MyDomain", "some-query")
```

```
LAMBDA("MyCustomDataSource", true, "fuzzy", 99.9)
```

Fungsi matematika metrik `LAMBDA` akan mengembalikan sebuah daftar deret waktu yang dapat dikembalikan ke pemohon atau dikombinasikan dengan fungsi matematika metrik lainnya. Berikut ini adalah contoh penggabungan `LAMBDA` dengan fungsi matematika metrik lainnya.

```
FILL(LAMBDA("AmazonOpenSearchDataSource", "MyDomain", "some-query"), 0)
```

Menghapus sebuah konektor ke sumber data

Untuk menghapus sebuah konektor ke sumber data, ikuti petunjuk yang ada di bagian ini.

Cara menghapus sebuah konektor ke sumber data

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Pengaturan.
3. Pilih tab Sumber data metrik.
4. Pilih Kelola CloudFormation di baris sumber data yang ingin Anda hapus.

   Anda dibawa ke AWS CloudFormation konsol.

5. Di bagian dengan nama sumber data Anda, pilih Hapus.
6. Di pop-up konfirmasi, pilih Hapus.

Kumpulkan metrik, log, dan jejak dengan agen CloudWatch

CloudWatch Agen terpadu memungkinkan Anda melakukan hal berikut:

- Kumpulkan metrik tingkat sistem internal dari instans Amazon EC2 di seluruh sistem operasi. Metrik dapat mencakup metrik dalam-tamu, selain metrik untuk instans EC2. Metrik tambahan yang dapat dikumpulkan tercantum di Metrik yang dikumpulkan oleh agen CloudWatch.
- Kumpulkan metrik tingkat sistem dari server on-premise. Ini dapat mencakup server dalam lingkungan hybrid serta server yang tidak dikelola oleh AWS.
- Ambil metrik kustom dari aplikasi atau layanan Anda menggunakan protokol `StatsD` dan `collectd`. `StatsD` didukung pada server Linux dan server yang menjalankan Server Windows. `collectd` hanya didukung di server Linux.
- Kumpulkan log dari instans Amazon EC2 dan server on-premise, dengan menjalankan Server Linux atau Windows.

  > **Note**
  >
  > CloudWatch Agen tidak mendukung pengumpulan log dari pipa FIFO.

- Versi 1.300031.0 dan yang lebih baru dapat digunakan untuk mengaktifkan Sinyal Aplikasi. CloudWatch Untuk informasi selengkapnya, lihat Sinyal Aplikasi.
- Versi 1.300025.0 dan yang lebih baru dapat mengumpulkan jejak dari atau SDK klien OpenTelemetryX-Ray, dan mengirimkannya ke X-Ray.

  Menggunakan CloudWatch agen memungkinkan Anda mengumpulkan jejak tanpa perlu menjalankan daemon pengumpulan jejak terpisah, membantu mengurangi jumlah agen yang Anda jalankan dan kelola.

Anda dapat menyimpan dan melihat metrik yang Anda kumpulkan dengan CloudWatch agen CloudWatch seperti yang Anda bisa dengan CloudWatch metrik lainnya. Namespace default untuk metrik yang dikumpulkan oleh CloudWatch agen adalah`CWAgent`, meskipun Anda dapat menentukan namespace yang berbeda saat mengonfigurasi agen.

Log yang dikumpulkan oleh CloudWatch agen terpadu diproses dan disimpan di CloudWatch Log Amazon, seperti log yang dikumpulkan oleh agen CloudWatch Log lama. Untuk informasi tentang harga CloudWatch Log, lihat CloudWatch Harga Amazon.

Metrik yang dikumpulkan oleh CloudWatch agen ditagih sebagai metrik khusus. Untuk informasi selengkapnya tentang harga CloudWatch metrik, lihat CloudWatchHarga Amazon.

CloudWatch Agen adalah open-source di bawah lisensi MIT, dan di-host di GitHub. Jika Anda ingin membuat, menyesuaikan, atau berkontribusi pada CloudWatch agen, lihat GitHub repositori untuk petunjuk terbaru. Jika Anda merasa telah menemukan masalah keamanan potensial, jangan mempostingnya di GitHub atau forum publik mana pun. Sebagai gantinya, silakan ikuti instruksi di Pelaporan Kerentanan atau AWS keamanan email secara langsung.

Langkah-langkah di bagian ini menjelaskan cara menginstal CloudWatch agen terpadu di instans Amazon EC2 dan server lokal. Untuk informasi selengkapnya tentang metrik yang dapat dikumpulkan CloudWatch agen, lihatMetrik yang dikumpulkan oleh agen CloudWatch.

Sistem operasi yang didukung

CloudWatch Agen didukung pada arsitektur x86-64 pada sistem operasi berikut. Ini juga didukung pada semua pembaruan versi minor untuk masing-masing versi utama yang dicantumkan di sini.

- Amazon Linux 2023
- Amazon Linux 2
- Ubuntu Server versi 23.10, 22.04, 20.04, 18.04, 16.04, dan 14.04
- CentOS versi 9, 8, dan 7
- Red Hat Enterprise Linux (RHEL) versi 9, 8, dan 7
- Debian versi 12, 11 dan 10
- SUSE Linux Enterprise Server (SLES) versi 15 dan 12
- Oracle Linux versi 9, 8 dan 7
- AlmaLinux versi 9 dan 8
- Rocky Linux versi 9 dan 8
- Komputer macOS berikut: instans EC2 M1 Mac1, dan komputer yang menjalankan macOS 14 (Sonoma), macOS 13 (Ventura), dan macOS 12 (Monterey)
- Versi 64-bit Windows Server 2022, Windows Server 2019, dan Windows Server 2016
- Windows 10 64-bit

Agen ini didukung pada arsitektur ARM64 pada sistem operasi berikut ini: Ini juga didukung pada semua pembaruan versi minor untuk masing-masing versi utama yang dicantumkan di sini.

- Amazon Linux 2023
- Amazon Linux 2
- Ubuntu Server versi 23.10, 22.04, 20.04, 18.04, dan 16.04
- CentOS versi 9 dan 8
- Red Hat Enterprise Linux (RHEL) versi 9, 8, dan 7
- Debian versi 12, 11 dan 10
- SUSE Linux Enterprise Server 15
- Komputer macOS berikut: macOS 14 (Sonoma), macOS 13 (Ventura), dan macOS 12 (Monterey)

Gambaran umum proses instalasi

Anda dapat mengunduh dan menginstal CloudWatch agen secara manual menggunakan baris perintah, atau Anda dapat mengintegrasikannya dengan SSM. Alur umum pemasangan CloudWatch agen menggunakan salah satu metode adalah sebagai berikut:

1. Buat peran IAM atau pengguna yang memungkinkan agen mengumpulkan metrik dari server dan secara opsional untuk diintegrasikan. AWS Systems Manager
2. Unduh paket agen.
3. Ubah file konfigurasi CloudWatch agen dan tentukan metrik yang ingin Anda kumpulkan.
4. Pasang dan mulai agen di server Anda. Saat melakukan instalasi agen di instans EC2, Anda melampirkan peran IAM yang Anda buat di langkah 1. Saat melakukan instalasi agen di server on-premise, Anda menentukan profil yang diberi nama yang berisi kredensial pengguna IAM yang Anda buat di langkah 1.

Daftar Isi

Instalasi CloudWatch agen

CloudWatch Agen tersedia sebagai paket di Amazon Linux 2023 dan Amazon Linux 2. Jika Anda menggunakan salah satu sistem operasi ini, Anda dapat menginstal paket dengan memasukkan perintah berikut. Anda juga harus memastikan bahwa peran IAM yang dilampirkan pada instance memiliki CloudWatchAgentServerPolicyterlampir. Untuk informasi selengkapnya, silakan lihat Buat peran IAM untuk digunakan dengan CloudWatch agen di instans Amazon EC2.

```
sudo yum install amazon-cloudwatch-agent
```

Pada semua sistem operasi yang didukung termasuk Linux dan Windows Server, Anda dapat mengunduh dan menginstal CloudWatch agen menggunakan baris perintah dengan tautan unduhan Amazon S3, menggunakan Amazon EC2 Systems Manager, atau menggunakan templat. AWS CloudFormation Lihat bagian-bagian berikut untuk detailnya.

Daftar Isi

Menginstal CloudWatch agen menggunakan baris perintah

Gunakan topik berikut untuk mengunduh, mengonfigurasi, dan menginstal paket CloudWatch agen.

Topik

Unduh dan konfigurasikan CloudWatch agen menggunakan baris perintah

Gunakan langkah-langkah berikut untuk mengunduh paket CloudWatch agen, membuat peran IAM atau pengguna, dan secara opsional memodifikasi file konfigurasi umum.

Unduh paket CloudWatch agen

> **Note**
>
> Untuk mengunduh CloudWatch agen, koneksi Anda harus menggunakan TLS 1.2 atau yang lebih baru.

CloudWatch Agen tersedia sebagai paket di Amazon Linux 2023 dan Amazon Linux 2. Jika menggunakan sistem operasi ini, Anda dapat melakukan instalasi paket dengan memasukkan perintah berikut. Anda juga harus memastikan bahwa peran IAM yang dilampirkan pada instance memiliki CloudWatchAgentServerPolicyterlampir. Untuk informasi selengkapnya, silakan lihat Buat peran IAM dan pengguna untuk digunakan dengan agen CloudWatch .

```
sudo yum install amazon-cloudwatch-agent
```

Pada semua sistem operasi yang didukung, Anda dapat mengunduh dan menginstal CloudWatch agen menggunakan baris perintah.

Untuk setiap tautan unduhan, terdapat tautan umum serta tautan untuk setiap Wilayah. Misalnya, untuk Amazon Linux 2023 dan Amazon Linux 2 dan arsitektur x86-64, tiga tautan unduhan yang valid adalah:

- `https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm`
- `https://amazoncloudwatch-agent-us-east-1.s3.us-east-1.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm`
- `https://amazoncloudwatch-agent-eu-central-1.s3.eu-central-1.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm`

Anda juga dapat mengunduh file README tentang perubahan terbaru ke agen, dan file yang menunjukkan nomor versi yang tersedia untuk diunduh. File-file ini ada di lokasi-lokasi berikut ini:

- `https://amazoncloudwatch-agent.s3.amazonaws.com/info/latest/RELEASE_NOTES` atau `https://amazoncloudwatch-agent-`*`region`*`.s3.`*`region`*`.amazonaws.com/info/latest/RELEASE_NOTES`

- `https://amazoncloudwatch-agent.s3.amazonaws.com/info/latest/CWAGENT_VERSION` atau `https://amazoncloudwatch-agent-region.s3.region.amazonaws.com/amazoncloudwatch-agent-region/info/latest/CWAGENT_VERSION`

Arsitektur	Platform	Tautan unduhan	Tautan file tanda tangan
x86-64	Amazon Linux 2023 dan Amazon Linux 2	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/ amazon-cloudwatch-agent .rpm https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/amazon_linux/amd64/terbaru/* .rpm amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/amazon_linux/amd64/latest/* amazon-cloudwatch-agent .rpm.sig
x86-64	Centos	https://amazoncloudwatch-agent.s3.amazonaws.com/centos/amd64/latest/ amazon-cloudwatch-agent .rpm https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/centos/amd64/latest/* .rpm amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/centos/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/centos/amd64/latest/* amazon-cloudwatch-agent .rpm.sig
x86-64	Redhat	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/amd64/latest/ amazon-cloudwatch-agent .rpm	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/amd64/latest/ amazon-cloudwatch-agent .rpm.sig

Arsitektur	Platform	Tautan unduhan	Tautan file tanda tangan
		https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/redhat/amd64/latest/* .rpm amazon-cloudwatch-agent	https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/redhat/amd64/latest/* amazon-cloudwatch-agent .rpm.sig
x86-64	SEGAR	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/amd64/latest/ amazon-cloudwatch-agent .rpm https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/suse/amd64/latest/* .rpm amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/suse/amd64/latest/* amazon-cloudwatch-agent .rpm.sig
x86-64	Debian	https://amazoncloudwatch-agent.s3.amazonaws.com/debian/amd64/latest/ amazon-cloudwatch-agent .deb https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/debian/amd64/latest/* .deb amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/debian/amd64/latest/ amazon-cloudwatch-agent .deb.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/debian/amd64/latest/* amazon-cloudwatch-agent .deb.sig

Arsitektur	Platform	Tautan unduhan	Tautan file tanda tangan
x86-64	Ubuntu	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/amd64/latest/ amazon-cloudwatch-agent .deb https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/ubuntu/amd64/latest/* .deb amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/amd64/latest/ amazon-cloudwatch-agent .deb.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/ubuntu/amd64/latest/* amazon-cloudwatch-agent .deb.sig
x86-64	Oracle	https://amazoncloudwatch-agent.s3.amazonaws.com/oracle_linux/amd64/latest/ amazon-cloudwatch-agent .rpm https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/oracle_linux/amd64/latest/* .rpm amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/oracle_linux/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/oracle_linux/amd64/latest/* amazon-cloudwatch-agent .rpm.sig
x86-64	macOS	https://amazoncloudwatch-agent.s3.amazonaws.com/darwin/amd64/latest/ amazon-cloudwatch-agent .pkg https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/darwin/amd64/latest/* amazon-cloudwatch-agent .pkg	https://amazoncloudwatch-agent.s3.amazonaws.com/darwin/amd64/latest/ amazon-cloudwatch-agent .pkg.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/darwin/amd64/latest/* amazon-cloudwatch-agent .pkg.sig

Arsitektur	Platform	Tautan unduhan	Tautan file tanda tangan
x86-64	Jendela	https://amazoncloudwatch-agent.s3.amazonaws.com/windows/amd64/latest/ amazon-cloudwatch-agent .msi https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/windows/amd64/latest/* amazon-cloudwatch-agent .msi	https://amazoncloudwatch-agent.s3.amazonaws.com/windows/amd64/latest/ amazon-cloudwatch-agent .msi.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/windows/amd64/latest/* amazon-cloudwatch-agent .msi.sig
ARM64	Amazon Linux 2023 dan Amazon Linux 2	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/arm64/latest/ amazon-cloudwatch-agent .rpm https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/amazon_linux/arm64/latest/* .rpm amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/arm64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/amazon_linux/arm64/latest/* amazon-cloudwatch-agent .rpm.sig

Arsitektur	Platform	Tautan unduhan	Tautan file tanda tangan
ARM64	Redhat	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/arm64/latest/ amazon-cloudwatch-agent .rpm https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/redhat/arm64/latest/* .rpm amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/arm64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/redhat/arm64/latest/* amazon-cloudwatch-agent .rpm.sig
ARM64	Ubuntu	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/arm64/latest/ amazon-cloudwatch-agent .deb https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/ubuntu/arm64/latest/* .deb amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/arm64/latest/ amazon-cloudwatch-agent .deb.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/ubuntu/arm64/latest/* amazon-cloudwatch-agent .deb.sig
ARM64	SEGAR	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/arm64/latest/ amazon-cloudwatch-agent .rpm https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/suse/arm64/latest/* .rpm amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/arm64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/suse/arm64/latest/* amazon-cloudwatch-agent .rpm.sig

Arsitektur	Platform	Tautan unduhan	Tautan file tanda tangan
ARM64	macOS	https://amazoncloudwatch-agent.s3.amazonaws.com/darwin/arm64/latest/ amazon-cloudwatch-agent .pkg https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/darwin/arm64/latest/* amazon-cloudwatch-agent .pkg	https://amazoncloudwatch-agent.s3.amazonaws.com/darwin/arm64/latest/ amazon-cloudwatch-agent .pkg.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/darwin/arm64/latest/* amazon-cloudwatch-agent .pkg.sig

Untuk menggunakan baris perintah untuk mengunduh dan menginstal paket CloudWatch agen

1. Unduh CloudWatch agennya.

   Pada server Linux, masukkan yang berikut ini. Untuk *tautan-unduhan*, gunakan tautan unduhan yang sesuai dari tabel sebelumnya.

   ```
   wget download-link
   ```

   Di server yang menjalankan Windows Server, unduh file berikut:

   ```
   https://amazoncloudwatch-agent.s3.amazonaws.com/windows/amd64/latest/amazon-cloudwatch-agent.msi
   ```

2. Setelah mengunduh paket, Anda dapat secara opsional memverifikasi tanda tangan paket. Untuk informasi selengkapnya, lihat Memverifikasi tanda tangan paket CloudWatch agen.
3. Instal paket. Jika Anda mengunduh paket RPM di server Linux, ubah ke direktori yang berisi paket dan masukkan yang berikut:

   ```
   sudo rpm -U ./amazon-cloudwatch-agent.rpm
   ```

   Jika Anda mengunduh paket DEB di server Linux, ubah ke direktori yang berisi paket dan masukkan yang berikut:

```
sudo dpkg -i -E ./amazon-cloudwatch-agent.deb
```

Jika Anda mengunduh paket MSI di server yang menjalankan Windows Server, ubah ke direktori yang berisi paket dan masukkan yang berikut:

```
msiexec /i amazon-cloudwatch-agent.msi
```

Perintah ini juga berfungsi dari dalam PowerShell. Untuk informasi selengkapnya tentang opsi perintah MSI, silakan lihat Opsi Baris Perintah dalam dokumentasi Microsoft Windows.

Jika Anda mengunduh paket PKG di server macOS, ubah ke direktori yang berisi paket dan masukkan yang berikut:

```
sudo installer -pkg ./amazon-cloudwatch-agent.pkg -target /
```

Buat dan Ubah File Konfigurasi Agen

Setelah mengunduh CloudWatch agen, Anda harus membuat file konfigurasi sebelum memulai agen di server mana pun. Untuk informasi selengkapnya, lihat Buat file konfigurasi CloudWatch agen.

Buat peran IAM dan pengguna untuk digunakan dengan agen CloudWatch

Akses ke AWS sumber daya memerlukan izin. Anda membuat peran IAM, pengguna IAM, atau keduanya untuk memberikan izin yang diperlukan CloudWatch agen untuk menulis metrik. CloudWatch Jika Anda akan menggunakan agen pada instans Amazon EC2, Anda harus membuat peran IAM. Jika akan menggunakan agen di server on-premise, Anda harus membuat pengguna IAM.

> **Note**
>
> Kami baru-baru ini memodifikasi prosedur berikut dengan menggunakan `CloudWatchAgentServerPolicy` dan `CloudWatchAgentAdminPolicy` kebijakan yang dibuat oleh Amazon, alih-alih mengharuskan pelanggan untuk membuat sendiri kebijakan ini. Untuk menulis file ke dan mengunduh file dari Parameter Store, kebijakan yang dibuat oleh Amazon hanya mendukung file dengan nama yang dimulai dengan `AmazonCloudWatch-`. Jika Anda memiliki file konfigurasi CloudWatch agen dengan nama

file yang tidak dimulai`AmazonCloudWatch-`, kebijakan ini tidak dapat digunakan untuk menulis file ke Parameter Store atau mengunduhnya dari Parameter Store.

Jika Anda akan menjalankan CloudWatch agen di instans Amazon EC2, gunakan langkah-langkah berikut untuk membuat peran IAM yang diperlukan. Peran ini memberikan izin untuk membaca informasi dari instance dan menuliskannya ke CloudWatch.

Untuk membuat peran IAM yang diperlukan untuk menjalankan CloudWatch agen pada instans EC2

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi di sebelah kiri, pilih Peran dan kemudian Buat peran.
3. Pastikan bahwa AWS layanan dipilih di bawah jenis entitas Tepercaya.
4. Untuk kasus Penggunaan, pilih EC2 di bawah Kasus penggunaan umum,
5. Pilih Berikutnya.
6. Dalam daftar kebijakan, pilih kotak centang di sebelah CloudWatchAgentServerPolicy. Jika perlu, gunakan kotak pencarian untuk menemukan kebijakan.
7. (Opsional) Jika agen mengirim jejak ke X-Ray, Anda juga perlu memberikan peran AWSXRayDaemonWriteAccesskebijakan tersebut. Untuk melakukan hal ini, Anda harus menemukan kebijakan tersebut dalam daftar dan kemudian pilih kotak centang yang ada di sebelahnya.
8. Pilih Berikutnya.
9. Di Nama peran, masukkan nama untuk peran tersebut, seperti *`CloudWatchAgentServerRole`*. Secara opsional, berikan deskripsi. Lalu pilih Buat peran.

   Peran ini sekarang dibuat.
10. (Opsional) Jika agen akan mengirim CloudWatch log ke Log dan Anda ingin agen dapat mengatur kebijakan retensi untuk grup log ini, Anda perlu menambahkan `logs:PutRetentionPolicy` izin ke peran tersebut. Untuk informasi selengkapnya, lihat Mengizinkan CloudWatch agen menyetel kebijakan retensi log.

Jika Anda akan menjalankan CloudWatch agen di server lokal, gunakan langkah-langkah berikut untuk membuat pengguna IAM yang diperlukan.

> **Warning**
>
> Skenario ini mengharuskan pengguna IAM dengan akses terprogram dan kredensil jangka panjang, yang menghadirkan risiko keamanan. Untuk membantu mengurangi risiko ini, kami menyarankan agar Anda memberikan pengguna ini hanya izin yang mereka perlukan untuk melakukan tugas dan menghapus pengguna ini ketika mereka tidak lagi diperlukan. Kunci akses dapat diperbarui jika perlu. Untuk informasi selengkapnya, lihat Memperbarui kunci akses di Panduan Pengguna IAM.

Untuk membuat pengguna IAM yang diperlukan agar CloudWatch agen dapat berjalan di server lokal

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi di sebelah kiri, pilih Pengguna dan kemudian Tambahkan pengguna.
3. Masukkan nama pengguna untuk pengguna baru.
4. Pilih Kunci akses - Akses terprogram dan pilih Berikutnya: Izin.
5. Pilih Lampirkan kebijakan yang sudah ada secara langsung.
6. Dalam daftar kebijakan, pilih kotak centang di sebelah CloudWatchAgentServerPolicy. Jika perlu, gunakan kotak pencarian untuk menemukan kebijakan.
7. (Opsional) Jika agen akan melacak ke X-Ray, Anda juga perlu memberikan peran AWSXRayDaemonWriteAccesskebijakan. Untuk melakukan hal ini, Anda harus menemukan kebijakan tersebut dalam daftar dan kemudian pilih kotak centang yang ada di sebelahnya.
8. Pilih Berikutnya: Tanda.
9. Secara opsional buat tanda untuk pengguna IAM baru, lalu pilih Berikutnya:Review.
10. Konfirmasikan bahwa kebijakan yang benar telah dicantumkan, dan kemudian pilih Buat pengguna.
11. Di samping nama pengguna baru, pilih Tampilkan. Salin kunci akses dan kunci rahasia ke file sehingga Anda dapat menggunakannya saat melakukan instalasi agen. Pilih Tutup

Mengizinkan CloudWatch agen menyetel kebijakan retensi log

Anda dapat mengonfigurasi CloudWatch agen untuk menyetel kebijakan penyimpanan untuk grup log tempat ia mengirim peristiwa log. Jika melakukan ini, Anda harus memberikan

`logs:PutRetentionPolicy` kepada peran IAM atau pengguna yang digunakan agen. Agen menggunakan peran IAM untuk berjalan di instans Amazon EC2, dan menggunakan pengguna IAM untuk server on-premise.

Untuk memberikan izin peran IAM CloudWatch agen untuk menetapkan kebijakan penyimpanan log

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi sebelah kiri, pilih Peran.
3. Di kotak pencarian, Ketik awal nama peran IAM CloudWatch agen. Anda memilih nama ini ketika Anda membuat peran tersebut. Itu bisa diberi nama`CloudWatchAgentServerRole`.

   Ketika Anda melihat peran, pilih nama peran.
4. Di tab Izin, pilih Tambahkan izin, Buat kebijakan tidak terpisah.
5. Pilih tab JSON dan kemudian salin kebijakan berikut ke dalam kotak, yang menggantikan JSON default dalam di kotak:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "logs:PutRetentionPolicy",
      "Resource": "*"
    }
  ]
}
```

6. Pilih Tinjau kebijakan.
7. Untuk Nama, masukkan **`CloudWatchAgentPutLogsRetention`** atau yang serupa, kemudian pilih Buat kebijakan.

Untuk memberikan izin kepada pengguna IAM CloudWatch agen untuk menetapkan kebijakan penyimpanan log

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi kiri, pilih Pengguna.

3. Di kotak pencarian, Ketik awal nama pengguna IAM CloudWatch agen. Anda memilih nama ini ketika Anda membuat pengguna.

   Saat Anda melihat pengguna, pilih nama pengguna.

4. Pada tab Izin, pilih Tambahkan kebijakan tak terpisahkan.

5. Pilih tab JSON dan kemudian salin kebijakan berikut ke dalam kotak, yang menggantikan JSON default dalam di kotak:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "logs:PutRetentionPolicy",
      "Resource": "*"
    }
  ]
}
```

6. Pilih Tinjau kebijakan.

7. Untuk Nama, masukkan **`CloudWatchAgentPutLogsRetention`** atau yang serupa, kemudian pilih Buat kebijakan.

Menginstal dan menjalankan CloudWatch agen di server Anda

Setelah Anda membuat file konfigurasi agen yang Anda inginkan dan membuat peran IAM atau pengguna IAM, gunakan langkah-langkah berikut untuk melakukan instalasi dan menjalankan agen di server Anda, dengan menggunakan konfigurasi tersebut. Pertama, lampirkan peran IAM atau pengguna IAM ke server yang akan menjalankan agen. Kemudian, pada server tersebut, unduh paket agen dan mulai menggunakan konfigurasi agen yang Anda buat.

Unduh paket CloudWatch agen menggunakan tautan unduhan S3

> **Note**
>
> Untuk mengunduh CloudWatch agen, koneksi Anda harus menggunakan TLS 1.2 atau yang lebih baru.

Anda perlu melakukan instalasi agen di setiap server tempat Anda akan menjalankan agen.

Amazon Linux 2

CloudWatch Agen tersedia sebagai paket di Amazon Linux 2023 dan Amazon Linux 2. Jika menggunakan sistem operasi ini, Anda dapat melakukan instalasi paket dengan memasukkan perintah berikut. Anda juga harus memastikan bahwa peran IAM yang dilampirkan pada instance memiliki CloudWatchAgentServerPolicyterlampir. Untuk informasi selengkapnya, silakan lihat Buat peran IAM untuk digunakan dengan CloudWatch agen di instans Amazon EC2.

```
sudo yum install amazon-cloudwatch-agent
```

Semua sistem operasi

Pada semua sistem operasi yang didukung, Anda dapat mengunduh dan menginstal CloudWatch agen menggunakan baris perintah dengan tautan unduhan Amazon S3 seperti yang dijelaskan dalam langkah-langkah berikut.

Untuk setiap tautan unduhan, terdapat tautan umum serta tautan untuk setiap Wilayah. Misalnya, untuk Amazon Linux 2023 dan Amazon Linux 2 dan arsitektur x86-64, tiga tautan unduhan yang valid adalah:

- `https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm`
- `https://amazoncloudwatch-agent-us-east-1.s3.us-east-1.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm`
- `https://amazoncloudwatch-agent-eu-central-1.s3.eu-central-1.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm`

Arsitektur	Platform	Tautan unduhan	Tautan file tanda tangan
x86-64	Amazon Linux 2023 dan Amazon Linux 2	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/ amazon-cloudwatch-agent .rpm	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/ amazon-cloudwatch-agent .rpm.sig

Arsitektur	Platform	Tautan unduhan	Tautan file tanda tangan
		https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/amazon_linux/amd64/terbaru/* .rpm amazon-cloudwatch-agent	https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/amazon_linux/amd64/latest/* amazon-cloudwatch-agent .rpm.sig
x86-64	Centos	https://amazoncloudwatch-agent.s3.amazonaws.com/centos/amd64/latest/ amazon-cloudwatch-agent .rpm https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/centos/amd64/latest/* .rpm amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/centos/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/centos/amd64/latest/* amazon-cloudwatch-agent .rpm.sig
x86-64	Redhat	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/amd64/latest/ amazon-cloudwatch-agent .rpm https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/redhat/amd64/latest/* .rpm amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/redhat/amd64/latest/* amazon-cloudwatch-agent .rpm.sig

Arsitektur	Platform	Tautan unduhan	Tautan file tanda tangan
x86-64	SEGAR	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/amd64/latest/ amazon-cloudwatch-agent .rpm https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/suse/amd64/latest/* .rpm amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/suse/amd64/latest/* amazon-cloudwatch-agent .rpm.sig
x86-64	Debian	https://amazoncloudwatch-agent.s3.amazonaws.com/debian/amd64/latest/ amazon-cloudwatch-agent .deb https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/debian/amd64/latest/* .deb amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/debian/amd64/latest/ amazon-cloudwatch-agent .deb.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/debian/amd64/latest/* amazon-cloudwatch-agent .deb.sig
x86-64	Ubuntu	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/amd64/latest/ amazon-cloudwatch-agent .deb https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/ubuntu/amd64/latest/* .deb amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/amd64/latest/ amazon-cloudwatch-agent .deb.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/ubuntu/amd64/latest/* amazon-cloudwatch-agent .deb.sig

Arsitektur	Platform	Tautan unduhan	Tautan file tanda tangan
x86-64	Oracle	https://amazoncloudwatch-agent.s3.amazonaws.com/oracle_linux/amd64/latest/ amazon-cloudwatch-agent .rpm https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/oracle_linux/amd64/latest/* .rpm amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/oracle_linux/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/oracle_linux/amd64/latest/* amazon-cloudwatch-agent .rpm.sig
x86-64	macOS	https://amazoncloudwatch-agent.s3.amazonaws.com/darwin/amd64/latest/ amazon-cloudwatch-agent .pkg https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/darwin/amd64/latest/* amazon-cloudwatch-agent .pkg	https://amazoncloudwatch-agent.s3.amazonaws.com/darwin/amd64/latest/ amazon-cloudwatch-agent .pkg.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/darwin/amd64/latest/* amazon-cloudwatch-agent .pkg.sig
x86-64	Jendela	https://amazoncloudwatch-agent.s3.amazonaws.com/windows/amd64/latest/ amazon-cloudwatch-agent .msi https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/windows/amd64/latest/* amazon-cloudwatch-agent .msi	https://amazoncloudwatch-agent.s3.amazonaws.com/windows/amd64/latest/ amazon-cloudwatch-agent .msi.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/windows/amd64/latest/* amazon-cloudwatch-agent .msi.sig

Arsitektur	Platform	Tautan unduhan	Tautan file tanda tangan
ARM64	Amazon Linux 2023 dan Amazon Linux 2	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/arm64/latest/ amazon-cloudwatch-agent .rpm https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/amazon_linux/arm64/latest/* .rpm amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/arm64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/amazon_linux/arm64/latest/* amazon-cloudwatch-agent .rpm.sig
ARM64	Redhat	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/arm64/latest/ amazon-cloudwatch-agent .rpm https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/redhat/arm64/latest/* .rpm amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/arm64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/redhat/arm64/latest/* amazon-cloudwatch-agent .rpm.sig
ARM64	Ubuntu	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/arm64/latest/ amazon-cloudwatch-agent .deb https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/ubuntu/arm64/latest/* .deb amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/arm64/latest/ amazon-cloudwatch-agent .deb.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/ubuntu/arm64/latest/* amazon-cloudwatch-agent .deb.sig

Arsitektur	Platform	Tautan unduhan	Tautan file tanda tangan
ARM64	SEGAR	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/arm64/latest/ amazon-cloudwatch-agent .rpm https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/suse/arm64/latest/* .rpm amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/arm64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/suse/arm64/latest/* amazon-cloudwatch-agent .rpm.sig

Untuk menggunakan baris perintah untuk menginstal CloudWatch agen pada instans Amazon EC2

1. Unduh CloudWatch agennya. Untuk server Linux, masukkan yang berikut ini. Untuk *tautan-unduhan*, gunakan tautan unduhan yang sesuai dari tabel sebelumnya.

   ```
   wget download-link
   ```

   Untuk server yang menjalankan Windows Server, unduh file berikut:

   ```
   https://amazoncloudwatch-agent.s3.amazonaws.com/windows/amd64/latest/amazon-cloudwatch-agent.msi
   ```

2. Setelah mengunduh paket, Anda dapat secara opsional memverifikasi tanda tangan paket. Untuk informasi selengkapnya, lihat Memverifikasi tanda tangan paket CloudWatch agen.
3. Instal paket. Jika Anda mengunduh paket RPM di server Linux, ubah ke direktori yang berisi paket dan masukkan yang berikut:

   ```
   sudo rpm -U ./amazon-cloudwatch-agent.rpm
   ```

   Jika Anda mengunduh paket DEB di server Linux, ubah ke direktori yang berisi paket dan masukkan yang berikut:

   ```
   sudo dpkg -i -E ./amazon-cloudwatch-agent.deb
   ```

Jika Anda mengunduh paket MSI di server yang menjalankan Windows Server, ubah ke direktori yang berisi paket dan masukkan yang berikut:

```
msiexec /i amazon-cloudwatch-agent.msi
```

Perintah ini juga berfungsi dari dalam PowerShell. Untuk informasi selengkapnya tentang opsi perintah MSI, silakan lihat Opsi Baris Perintah dalam dokumentasi Microsoft Windows.

(Instal pada Instans EC2) Melampirkan Peran IAM

Untuk mengaktifkan CloudWatch agen mengirim data dari instance, Anda harus melampirkan peran IAM ke instance. Peran untuk dilampirkan adalah CloudWatchAgentServerRole. Anda seharusnya membuat peran ini sebelumnya. Untuk informasi selengkapnya, silakan lihat Buat peran IAM dan pengguna untuk digunakan dengan agen CloudWatch .

Untuk informasi selengkapnya tentang melampirkan peran IAM ke contohnya, silakan lihat Melampirkan Peran IAM ke Instans dalam Panduan Pengguna Amazon EC2 untuk Instans Windows.

(Menginstal di server lokal) Tentukan kredensyal IAM dan Wilayah AWS

Untuk mengaktifkan CloudWatch agen mengirim data dari server lokal, Anda harus menentukan kunci akses dan kunci rahasia pengguna IAM yang Anda buat sebelumnya. Untuk informasi selengkapnya tentang membuat pengguna ini, silakan lihat Buat peran IAM dan pengguna untuk digunakan dengan agen CloudWatch .

Anda juga harus menentukan AWS Wilayah untuk mengirim metrik ke, menggunakan `region` bidang di `[AmazonCloudWatchAgent]` bagian file AWS konfigurasi, seperti pada contoh berikut.

```
[profile AmazonCloudWatchAgent]
region = us-west-1
```

Berikut ini adalah contoh penggunaan `aws configure` perintah untuk membuat profil bernama untuk CloudWatch agen. Contoh ini mengasumsikan Anda sedang menggunakan nama profil bawaan dari `AmazonCloudWatchAgent`.

Untuk membuat AmazonCloudWatchAgent profil untuk CloudWatch agen

1. Jika Anda belum melakukannya, instal AWS Command Line Interface di server. Untuk informasi selengkapnya, silakan lihat Menginstal AWS CLI.

2. Pada server Linux, masukkan perintah berikut dan ikuti petunjuknya:

```
sudo aws configure --profile AmazonCloudWatchAgent
```

Di Windows Server, buka PowerShell sebagai administrator, masukkan perintah berikut, dan ikuti petunjuknya.

```
aws configure --profile AmazonCloudWatchAgent
```

Memverifikasi Akses Internet

Instans Amazon EC2 Anda harus memiliki akses internet keluar untuk mengirim data ke atau Log. CloudWatch CloudWatch Untuk informasi selengkapnya tentang cara mengonfigurasi akses internet, silakan lihat Gateway Internet dalam Panduan Pengguna VPC Amazon.

Titik akhir dan port yang harus dikonfigurasikan pada proksi Anda adalah sebagai berikut:

- Jika Anda menggunakan agen untuk mengumpulkan metrik, Anda harus menambahkan CloudWatch titik akhir untuk Wilayah yang sesuai ke daftar izin. Titik akhir ini tercantum dalam CloudWatch titik akhir dan kuota Amazon.
- Jika Anda menggunakan agen untuk mengumpulkan log, Anda harus menambahkan titik akhir CloudWatch Log untuk Wilayah yang sesuai ke daftar izin. Titik akhir ini tercantum dalam titik akhir dan CloudWatch kuota Amazon Logs.
- Jika Anda menggunakan Systems Manager untuk melakukan instalasi agen atau Parameter Store untuk menyimpan file konfigurasi Anda, Anda harus menambahkan titik akhir Systems Manager untuk Wilayah yang sesuai ke daftar izin. Titik akhir ini tercantum dalam AWS Systems Manager titik akhir dankuota.

(Opsional) Ubah Konfigurasi Umum untuk Informasi Proxy atau Wilayah

CloudWatch Agen menyertakan file konfigurasi yang disebut`common-config.toml`. Secara opsional, Anda dapat menggunakan file ini sebagai pilihan untuk menentukan informasi proksi dan Wilayah.

Di server yang menjalankan Linux, file ini ada di `/opt/aws/amazon-cloudwatch-agent/etc` direktori. Pada server yang menjalankan Server Windows, file ini berada di direktori `C:\ProgramData\Amazon\AmazonCloudWatchAgent`.

Standarnya `common-config.toml` adalah sebagai berikut.

```
# This common-config is used to configure items used for both ssm and cloudwatch access


## Configuration for shared credential.
## Default credential strategy will be used if it is absent here:
##            Instance role is used for EC2 case by default.
##            AmazonCloudWatchAgent profile is used for the on-premises case by
 default.
# [credentials]
#    shared_credential_profile = "{profile_name}"
#    shared_credential_file= "{file_name}"

## Configuration for proxy.
## System-wide environment-variable will be read if it is absent here.
## i.e. HTTP_PROXY/http_proxy; HTTPS_PROXY/https_proxy; NO_PROXY/no_proxy
## Note: system-wide environment-variable is not accessible when using ssm run-command.
## Absent in both here and environment-variable means no proxy will be used.
# [proxy]
#    http_proxy = "{http_url}"
#    https_proxy = "{https_url}"
#    no_proxy = "{domain}"
```

Semua baris pada awalnya berkomentar. Untuk mengatur pengaturan profil kredensial atau proksi, hapus # dari baris tersebut dan tentukan nilai. Anda dapat mengedit file ini secara manual atau dengan menggunakan `RunShellScript` Jalankan Perintah dalam Manajer Sistem:

- `shared_credential_profile`— Untuk server lokal, baris ini menentukan profil kredensi pengguna IAM yang akan digunakan untuk mengirim data. CloudWatch Jika Anda tetap baris ini berkomentar, `AmazonCloudWatchAgent` digunakan. Untuk informasi selengkapnya tentang membuat profil ini, silakan lihat (Menginstal di server lokal) Tentukan kredensyal IAM dan Wilayah AWS.

  Pada instans EC2, Anda dapat menggunakan baris ini untuk meminta CloudWatch agen mengirim data dari instance ini ke CloudWatch AWS Wilayah yang berbeda. Untuk melakukan hal itu, tentukan profil dengan nama yang mencakup `region` bidang yang menetapkan nama Wilayah yang akan dikirim.

  Jika menentukan `shared_credential_profile`, Anda juga harus menghapus # dari awal `[credentials]` yang sesuai.

- `shared_credential_file` – Untuk meminta agen mencari kredensial dalam file yang terletak di jalur selain jalur default, tentukan jalur dan nama file lengkap di sini. Jalur default adalah `/root/.aws` di Linux dan `C:\\Users\\Administrator\\.aws` pada Server Windows.

  Contoh pertama di bawah ini menunjukkan sintaks baris `shared_credential_file` yang valid untuk server Linux, dan contoh kedua valid untuk Server Windows. Pada Server Windows, Anda harus menghindari karakter \.

  ```
  shared_credential_file= "/usr/username/credentials"
  ```

  ```
  shared_credential_file= "C:\\Documents and Settings\\username\\.aws\\credentials"
  ```

  Jika menentukan `shared_credential_file`, Anda juga harus menghapus # dari awal `[credentials]` yang sesuai.

- Pengaturan proxy – Jika server Anda menggunakan proxy HTTP atau HTTPS untuk menghubungi layanan AWS , tentukan proxy tersebut di bidang `http_proxy` dan `https_proxy`. Jika ada URL yang harus dikecualikan dari proxy, tentukan di bidang `no_proxy`, dipisahkan dengan koma.

Mulai CloudWatch agen menggunakan baris perintah

Ikuti langkah-langkah ini untuk menggunakan baris perintah untuk memulai CloudWatch agen di server.

Untuk menggunakan baris perintah untuk memulai CloudWatch agen di server

1. Salin file konfigurasi agen yang ingin Anda gunakan ke server di mana Anda akan menjalankan agen. Catat nama lokasi tempat Anda menyalinnya.
2. Dalam perintah ini, `-a fetch-config` menyebabkan agen memuat versi terbaru dari file konfigurasi CloudWatch agen, dan `-s` memulai agen.

   Gunakan salah satu perintah berikut ini. Ganti *`configuration-file-path`*dengan path ke file konfigurasi agen. File ini disebut `config.json` Jika kau membuatnya dengan wizard, dan bisa dipanggil `amazon-cloudwatch-agent.json` jika Anda membuatnya secara manual.

   Pada instans EC2 yang menjalankan Linux, masukkan perintah berikut.

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-
config -m ec2 -s -c file:configuration-file-path
```

Di server on-premise yang menjalankan Linux, masukkan berikut ini:

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-
config -m onPremise -s -c file:configuration-file-path
```

Pada instans EC2 yang menjalankan Windows Server, masukkan yang berikut dari PowerShell konsol:

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -
a fetch-config -m ec2 -s -c file:configuration-file-path
```

Di server lokal yang menjalankan Windows Server, masukkan yang berikut dari PowerShell konsol:

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -
a fetch-config -m onPremise -s -c file:configuration-file-path
```

Instalasi CloudWatch agen menggunakan AWS Systems Manager

Gunakan topik berikut untuk menginstal dan menjalankan CloudWatch agen menggunakan AWS Systems Manager.

Topik

- Buat peran IAM dan pengguna untuk digunakan dengan agen CloudWatch
- Unduh dan konfigurasikan CloudWatch agen
- Menginstal CloudWatch agen pada instans EC2 menggunakan konfigurasi agen Anda
- Menginstal CloudWatch agen di server lokal

Buat peran IAM dan pengguna untuk digunakan dengan agen CloudWatch

Akses ke AWS sumber daya memerlukan izin. Anda dapat membuat peran IAM dan pengguna yang menyertakan izin yang Anda perlukan bagi CloudWatch agen untuk menulis metrik CloudWatch dan

agar CloudWatch agen dapat berkomunikasi dengan Amazon EC2 dan. AWS Systems Manager Anda menggunakan peran IAM di instans Amazon EC2, dan Anda menggunakan pengguna IAM dengan server on-premise.

Satu peran atau pengguna memungkinkan CloudWatch agen untuk diinstal pada server dan mengirim metrik ke CloudWatch. Peran atau pengguna lain diperlukan untuk menyimpan konfigurasi CloudWatch agen Anda di Systems Manager Parameter Store. Parameter Store memungkinkan beberapa server untuk menggunakan satu konfigurasi CloudWatch agen.

Kemampuan menulis ke Parameter Store adalah izin yang luas dan kuat. Anda harus menggunakannya hanya saat memerlukannya, dan tidak boleh dilampirkan ke beberapa instans dalam deployment Anda. Jika Anda menyimpan konfigurasi CloudWatch agen Anda di Parameter Store, kami merekomendasikan hal berikut:

- Atur satu instans tempat Anda melakukan konfigurasi ini.
- Gunakan peran IAM dengan izin untuk menulis ke Parameter Store hanya pada instans ini.
- Gunakan peran IAM dengan izin untuk menulis ke Parameter Store hanya saat Anda bekerja dengan dan menyimpan file konfigurasi CloudWatch agen.

> **Note**
>
> Kami baru-baru ini memodifikasi prosedur berikut dengan menggunakan kebijakan `CloudWatchAgentServerPolicy` dan kebijakan `CloudWatchAgentAdminPolicy` baru yang dibuat oleh Amazon, alih-alih mengharuskan pelanggan untuk membuat sendiri kebijakan ini. Untuk menggunakan kebijakan ini untuk menulis file konfigurasi agen ke Parameter Store lalu mengunduhnya dari Parameter Store, file konfigurasi agen Anda harus memiliki nama yang diawali dengan `AmazonCloudWatch-`. Jika Anda memiliki file konfigurasi CloudWatch agen dengan nama file yang tidak dimulai`AmazonCloudWatch-`, kebijakan ini tidak dapat digunakan untuk menulis file ke Parameter Store atau untuk mengunduh file dari Parameter Store.

Buat peran IAM untuk digunakan dengan CloudWatch agen di instans Amazon EC2

Prosedur pertama membuat peran IAM yang harus Anda lampirkan ke setiap instans Amazon EC2 yang menjalankan CloudWatch agen. Peran ini memberikan izin untuk membaca informasi dari instance dan menuliskannya ke CloudWatch.

Prosedur kedua membuat peran IAM yang harus Anda lampirkan ke instans Amazon EC2 yang digunakan untuk membuat file konfigurasi CloudWatch agen. Langkah ini diperlukan jika Anda akan menyimpan file ini di Systems Manager Parameter Store sehingga server-server yang lain dapat menggunakannya. Peran ini memberikan izin untuk menulis ke Parameter Store, selain izin untuk membaca informasi dari instance dan menulisnya. CloudWatch Peran ini mencakup izin yang cukup untuk menjalankan CloudWatch agen serta menulis ke Parameter Store.

> **Note**
>
> Parameter Store mendukung parameter-parameter yang ada di tingkat Standar dan Tingkat Lanjut. Tingkatan parameter ini tidak terkait dengan tingkat detail Dasar, Standar, dan Lanjutan yang tersedia dengan set metrik CloudWatch Agen yang telah ditentukan sebelumnya.

Untuk membuat peran IAM yang diperlukan untuk setiap server untuk menjalankan agen CloudWatch

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi, silakan pilih Peran lalu pilih Buat peran.
3. Di bawah Pilih jenis entitas terpercaya, pilih AWS layanan.
4. Segera di bawah Kasus penggunaan umum, pilih EC2, lalu pilih Berikutnya: Izin.
5. Dalam daftar kebijakan, pilih kotak centang di sebelah CloudWatchAgentServerPolicy. Jika perlu, gunakan kotak pencarian untuk menemukan kebijakan.
6. Untuk menggunakan Systems Manager untuk menginstal atau mengkonfigurasi CloudWatch agen, pilih kotak di sebelah AmazonSSM ManagedInstanceCore. Kebijakan AWS terkelola ini memungkinkan instance untuk menggunakan fungsionalitas inti layanan Systems Manager. Jika perlu, gunakan kotak pencarian untuk menemukan kebijakan. Kebijakan ini tidak diperlukan jika Anda memulai dan mengonfigurasi agen hanya melalui baris perintah.
7. Pilih Berikutnya: Tag
8. (Opsional) Tambahkan satu atau beberapa pasangan nilai kunci tanda untuk mengatur, melacak, atau mengontrol akses untuk peran ini, lalu pilih Berikutnya: Tinjauan.
9. Untuk Nama peran, masukkan nama untuk peran baru Anda, seperti **`CloudWatchAgentServerRole`** atau nama lain yang Anda inginkan.
10. (Opsional) Untuk Deskripsi peran, masukkan deskripsi.

11. Konfirmasikan itu CloudWatchAgentServerPolicydan secara opsional AmazonSSM ManagedInstanceCore muncul di sebelah Kebijakan.
12. Pilih Buat peran.

    Peran ini sekarang dibuat.

Prosedur berikut membuat peran IAM yang juga dapat menulis ke Parameter Store. Anda dapat menggunakan peran ini untuk menyimpan file konfigurasi agen di Parameter Store sehingga server lain dapat mengambilnya.

Izin untuk menulis ke Parameter Store memberikan akses luas. Peran ini seharusnya tidak dilampirkan ke semua server Anda, dan hanya administrator yang boleh menggunakannya. Setelah Anda membuat file konfigurasi agen dan menyalinnya ke Parameter Store, Anda harus melepaskan peran ini dari instans dan sebaliknya menggunakan `CloudWatchAgentServerRole`.

Untuk membuat peran IAM bagi administrator untuk menulis ke Parameter Store

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi, silakan pilih Peran lalu pilih Buat peran.
3. Di bawah Pilih jenis entitas terpercaya, pilih AWS layanan.
4. Segera di bawah Pilih layanan yang akan menggunakan peran ini, pilih EC2, lalu pilih Berikutnya: Izin.
5. Dalam daftar kebijakan, pilih kotak centang di sebelah CloudWatchAgentAdminPolicy. Jika perlu, gunakan kotak pencarian untuk menemukan kebijakan.
6. Untuk menggunakan Systems Manager untuk menginstal atau mengkonfigurasi CloudWatch agen, pilih kotak di sebelah AmazonSSM ManagedInstanceCore. Kebijakan AWS terkelola ini memungkinkan instance untuk menggunakan fungsionalitas inti layanan Systems Manager. Jika perlu, gunakan kotak pencarian untuk menemukan kebijakan. Kebijakan ini tidak diperlukan jika Anda memulai dan mengonfigurasi agen hanya melalui baris perintah.
7. Pilih Berikutnya: Tag
8. (Opsional) Tambahkan satu atau beberapa pasangan nilai kunci tanda untuk mengatur, melacak, atau mengontrol akses untuk peran ini, lalu pilih Berikutnya: Tinjauan.
9. Untuk Nama peran, masukkan nama untuk peran baru Anda, seperti **`CloudWatchAgentAdminRole`** atau nama lain yang Anda inginkan.
10. (Opsional) Untuk Deskripsi peran, masukkan deskripsi.

11. Konfirmasikan itu CloudWatchAgentAdminPolicydan secara opsional AmazonSSM ManagedInstanceCore muncul di sebelah Kebijakan.
12. Pilih Buat peran.

    Peran ini sekarang dibuat.

Buat pengguna IAM untuk digunakan dengan CloudWatch agen di server lokal

Prosedur pertama menciptakan pengguna IAM yang Anda butuhkan untuk menjalankan CloudWatch agen. Pengguna ini memberikan izin untuk mengirim data ke CloudWatch.

Prosedur kedua membuat pengguna IAM yang dapat Anda gunakan saat membuat file konfigurasi CloudWatch agen. Gunakan prosedur ini untuk menyimpan file ini di Systems Manager Parameter Store sehingga server lain dapat menggunakannya. Pengguna ini memberikan izin untuk menulis ke Parameter Store, selain izin untuk menulis data ke. CloudWatch

> **Note**
>
> Parameter Store mendukung parameter-parameter yang ada di tingkat Standar dan Tingkat Lanjut. Tingkatan parameter ini tidak terkait dengan tingkat detail Dasar, Standar, dan Lanjutan yang tersedia dengan set metrik CloudWatch Agen yang telah ditentukan sebelumnya.

Untuk membuat pengguna IAM diperlukan bagi CloudWatch agen untuk menulis data CloudWatch

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi, silakan pilih Pengguna, lalu pilih Tambahkan pengguna.
3. Masukkan nama pengguna untuk pengguna baru.
4. Untuk Jenis akses, pilih Akses terprogram, lalu pilih Berikutnya: Izin.
5. Untuk Atur izin, pilih Lampirkan kebijakan yang ada secara langsung.
6. Dalam daftar kebijakan, pilih kotak centang di sebelah CloudWatchAgentServerPolicy. Jika perlu, gunakan kotak pencarian untuk menemukan kebijakan.
7. Untuk menggunakan Systems Manager untuk menginstal atau mengkonfigurasi CloudWatch agen, pilih kotak di sebelah AmazonSSM ManagedInstanceCore. Kebijakan AWS terkelola ini memungkinkan instance untuk menggunakan fungsionalitas inti layanan Systems Manager. (Jika

perlu, gunakan kotak pencarian untuk menemukan kebijakan. Kebijakan ini tidak diperlukan jika Anda memulai dan mengonfigurasi agen hanya melalui baris perintah.)

8. Pilih Selanjutnya: Tanda.
9. (Opsional) Tambahkan satu atau beberapa pasangan nilai kunci tanda untuk mengatur, melacak, atau mengontrol akses untuk peran ini, lalu pilih Berikutnya: Tinjauan.
10. Konfirmasikan bahwa kebijakan yang benar dicantumkan, lalu pilih Buat pengguna.
11. Di baris untuk pengguna baru, pilih Tampilkan. Salin kunci akses dan kunci rahasia ke file sehingga Anda dapat menggunakannya saat melakukan instalasi agen. Pilih Tutup

Prosedur berikut membuat pengguna IAM yang juga dapat menulis ke Parameter Store. Jika ingin menyimpan file konfigurasi agen di Parameter Store sehingga server lain dapat menggunakannya, Anda perlu menggunakan pengguna IAM ini. Pengguna IAM ini memberikan izin untuk menulis ke Parameter Store. Pengguna ini juga memberikan izin untuk membaca informasi dari instance dan menuliskannya ke CloudWatch. Izin untuk menulis ke Systems Manager Parameter Store menyediakan akses luas. Pengguna IAM ini seharusnya tidak dilampirkan ke semua server Anda, dan hanya administrator yang boleh menggunakannya. Anda harus menggunakan pengguna IAM ini hanya ketika Anda menyimpan file konfigurasi agen di Parameter Store.

Untuk membuat pengguna IAM diperlukan untuk menyimpan file konfigurasi di Parameter Store dan mengirim informasi ke CloudWatch

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi, silakan pilih Pengguna, lalu pilih Tambahkan pengguna.
3. Masukkan nama pengguna untuk pengguna baru.
4. Untuk Jenis akses, pilih Akses terprogram, lalu pilih Berikutnya: Izin.
5. Untuk Atur izin, pilih Lampirkan kebijakan yang ada secara langsung.
6. Dalam daftar kebijakan, pilih kotak centang di sebelah CloudWatchAgentAdminPolicy. Jika perlu, gunakan kotak pencarian untuk menemukan kebijakan.
7. Untuk menggunakan Systems Manager untuk menginstal atau mengkonfigurasi CloudWatch agen, pilih kotak centang di sebelah AmazonSSM ManagedInstanceCore. Kebijakan AWS terkelola ini memungkinkan instance untuk menggunakan fungsionalitas inti layanan Systems Manager. (Jika perlu, gunakan kotak pencarian untuk menemukan kebijakan. Kebijakan ini tidak diperlukan jika Anda memulai dan mengonfigurasi agen hanya melalui baris perintah.)
8. Pilih Selanjutnya: Tanda.

9. (Opsional) Tambahkan satu atau beberapa pasangan nilai kunci tanda untuk mengatur, melacak, atau mengontrol akses untuk peran ini, lalu pilih Berikutnya: Tinjauan.
10. Konfirmasikan bahwa kebijakan yang benar dicantumkan, lalu pilih Buat pengguna.
11. Di baris untuk pengguna baru, pilih Tampilkan. Salin kunci akses dan kunci rahasia ke file sehingga Anda dapat menggunakannya saat melakukan instalasi agen. Pilih Tutup

Unduh dan konfigurasikan CloudWatch agen

Bagian ini menjelaskan cara menggunakan System Manager untuk mengunduh agen dan kemudian cara membuat file konfigurasi agen Anda. Sebelum dapat menggunakan Manajer Sistem untuk mengunduh agen, Anda harus memastikan bahwa instans dikonfigurasi dengan benar untuk Manajer Sistem.

Menginstal atau memperbarui Agen SSM

Pada instans Amazon EC2, CloudWatch agen mengharuskan instans menjalankan versi 2.2.93.0 atau yang lebih baru. Sebelum Anda menginstal CloudWatch agen, perbarui atau instal Agen SSM pada instance jika Anda belum melakukannya.

Untuk informasi tentang melakukan instalasi atau memperbarui Agen SSM pada instans yang menjalankan Linux, silakan lihat Memasang dan Mengonfigurasi Agen SSM di Instans Linux di Panduan Pengguna AWS Systems Manager .

Untuk informasi tentang menginstal atau memperbarui Agen SSM, silakan lihat Bekerja dengan Agen SSM di Panduan Pengguna AWS Systems Manager .

(Opsional) Verifikasikan Prasyarat Manajer Sistem

Sebelum Anda menggunakan Systems Manager Run Command untuk menginstal dan mengonfigurasi CloudWatch agen, verifikasi bahwa instans Anda memenuhi persyaratan minimum Systems Manager. Untuk informasi selengkapnya, silakan lihat Prasyarat Systems Manager di Panduan Pengguna AWS Systems Manager .

Memverifikasi Akses Internet

Instans Amazon EC2 Anda harus memiliki akses internet keluar untuk mengirim data ke atau Log. CloudWatch CloudWatch Untuk informasi selengkapnya tentang cara mengonfigurasi akses internet, silakan lihat Gateway Internet dalam Panduan Pengguna VPC Amazon.

Titik akhir dan port yang harus dikonfigurasikan pada proksi Anda adalah sebagai berikut:

- Jika Anda menggunakan agen untuk mengumpulkan metrik, Anda harus mengizinkan daftar CloudWatch titik akhir untuk Wilayah yang sesuai. Titik akhir ini terdaftar di Amazon CloudWatch di. Referensi Umum Amazon Web Services
- Jika Anda menggunakan agen untuk mengumpulkan log, Anda harus mengizinkan daftar titik akhir CloudWatch Log untuk Wilayah yang sesuai. Titik akhir ini tercantum di CloudWatch Log Amazon di. Referensi Umum Amazon Web Services
- Jika menggunakan Systems Manager untuk melakukan instalasi agen atau Parameter Store untuk menyimpan file konfigurasi Anda, Anda harus mengizinkan daftar titik akhir Systems Manager untuk Wilayah yang sesuai. Titik akhir ini tercantum dalam AWS Systems Manager di Referensi Umum Amazon Web Services.

Gunakan langkah-langkah berikut untuk mengunduh paket CloudWatch agen menggunakan Systems Manager.

Untuk mengunduh CloudWatch agen menggunakan Systems Manager

1. Buka konsol Systems Manager di https://console.aws.amazon.com/systems-manager/.
2. Pada panel navigasi, silakan pilih Perintah Eksekusi.

   -atau-

   Jika AWS Systems Manager halaman beranda terbuka, gulir ke bawah dan pilih Jelajahi Jalankan Perintah.
3. Pilih Jalankan perintah.
4. Dalam daftar dokumen Command, pilih AWS-configure AWSPackage.
5. Di area Target, pilih instance untuk menginstal CloudWatch agen. Jika Anda tidak melihat instans tertentu, itu mungkin tidak dikonfigurasi sebagai contoh terkelola untuk digunakan dengan Manajer Sistem. Untuk informasi selengkapnya, lihat AWS Systems Manager Menyiapkan Lingkungan Hybrid di Panduan AWS Systems Manager Pengguna.
6. Di daftar Tindakan pilih, Instal.
7. Di bidang Nama, masukkan *AmazonCloudWatchAgent*.
8. Simpan Versi diatur ke terbaru untuk melakukan instalasi versi terbaru agen.
9. Pilih Jalankan.
10. Secara opsional, dalam Target dan keluaran, pilih tombol yang ada di samping nama instans dan pilih Lihat output. Manajer Sistem harus menunjukkan bahwa agen berhasil diinstal.

Buat dan Ubah File Konfigurasi Agen

Setelah mengunduh CloudWatch agen, Anda harus membuat file konfigurasi sebelum memulai agen di server mana pun.

Jika akan menyimpan file konfigurasi agen di Systems Manager Parameter Store, Anda harus menggunakan instans EC2 untuk menyimpan ke Parameter Store. Selain itu, Anda harus terlebih dahulu melampirkan Peran IAM `CloudWatchAgentAdminRole` ke instans tersebut. Untuk informasi selengkapnya tentang melampirkan peran, silakan lihat Melampirkan Peran IAM ke Instans dalam Panduan Pengguna Amazon EC2 untuk Instans Windows.

Untuk informasi selengkapnya tentang membuat file konfigurasi CloudWatch agen, lihatBuat file konfigurasi CloudWatch agen.

Menginstal CloudWatch agen pada instans EC2 menggunakan konfigurasi agen Anda

Setelah Anda memiliki konfigurasi CloudWatch agen yang disimpan di Parameter Store, Anda dapat menggunakannya ketika Anda menginstal agen di server lain.

Topik

- Melampirkan peran IAM ke instans
- Unduh paket CloudWatch agen pada instans Amazon EC2
- (Opsional) Ubah konfigurasi umum dan profil bernama untuk CloudWatch agen
- Mulai CloudWatch agen

Melampirkan peran IAM ke instans

Anda harus melampirkan peran CloudWatchAgentServerRoleIAM ke instans EC2 untuk dapat menjalankan CloudWatch agen pada instance. Peran ini memungkinkan CloudWatch agen untuk melakukan tindakan pada instance. Anda seharusnya membuat peran ini sebelumnya. Untuk informasi selengkapnya, silakan lihat Buat peran IAM dan pengguna untuk digunakan dengan agen CloudWatch .

Untuk informasi selengkapnya, silakan lihat Melampirkan Peran IAM ke Instans dalam Panduan Pengguna Amazon EC2 untuk Instans Windows.

Unduh paket CloudWatch agen pada instans Amazon EC2

Anda perlu melakukan instalasi agen di setiap server tempat Anda akan menjalankan agen. CloudWatch Agen tersedia sebagai paket di Amazon Linux 2023 dan Amazon Linux 2. Jika menggunakan sistem operasi ini, Anda dapat melakukan instalasi paket dengan memasukkan perintah berikut. Anda juga harus memastikan bahwa peran IAM yang dilampirkan pada instance memiliki CloudWatchAgentServerPolicyterlampir. Untuk informasi selengkapnya, silakan lihat Buat peran IAM untuk digunakan dengan CloudWatch agen di instans Amazon EC2.

```
sudo yum install amazon-cloudwatch-agent
```

Pada semua sistem operasi yang didukung, Anda dapat mengunduh paket CloudWatch agen menggunakan Systems Manager Run Command atau tautan unduhan Amazon S3. Untuk informasi tentang penggunaan tautan unduhan Amazon S3, silakan lihat Unduh paket CloudWatch agen.

> **Note**
>
> Saat Anda menginstal atau memperbarui CloudWatch agen, hanya opsi Uninstall dan instal ulang yang didukung. Anda tidak dapat menggunakan opsi Pembaruan di tempat.

Unduh CloudWatch agen di instans Amazon EC2 Menggunakan Systems Manager

Sebelum Anda dapat menggunakan Systems Manager untuk menginstal CloudWatch agen, Anda harus memastikan bahwa instance dikonfigurasi dengan benar untuk Systems Manager.

Menginstal atau memperbarui Agen SSM

Pada instans Amazon EC2, CloudWatch agen mengharuskan instans menjalankan versi 2.2.93.0 atau yang lebih baru. Sebelum Anda menginstal CloudWatch agen, perbarui atau instal Agen SSM pada instance jika Anda belum melakukannya.

Untuk informasi tentang melakukan instalasi atau memperbarui Agen SSM pada instans yang menjalankan Linux, silakan lihat Memasang dan Mengonfigurasi Agen SSM pada Instans Linux di AWS Systems Manager Panduan Pengguna.

Untuk informasi tentang melakukan instalasi atau memperbarui Agen SSM pada instans yang menjalankan Windows Server, silakan lihat Memasang dan Mengonfigurasi Agen SSM pada Instans Windows di Panduan Pengguna AWS Systems Manager .

(Opsional) Verifikasikan Prasyarat Manajer Sistem

Sebelum Anda menggunakan Systems Manager Run Command untuk menginstal dan mengonfigurasi CloudWatch agen, verifikasi bahwa instans Anda memenuhi persyaratan minimum Systems Manager. Untuk informasi selengkapnya, silakan lihat Pengaturan AWS Systems Manager dalam AWS Systems Manager Panduan Pengguna.

Memverifikasi Akses Internet

Instans Amazon EC2 Anda harus memiliki akses internet keluar untuk mengirim data ke atau Log. CloudWatch CloudWatch Untuk informasi selengkapnya tentang cara mengonfigurasi akses internet, silakan lihat Gateway Internet dalam Panduan Pengguna VPC Amazon.

Unduh paket CloudWatch agen

Perintah Operasi Manajer Sistem memungkinkan Anda mengelola konfigurasi instans Anda. Anda menentukan dokumen Manajer Sistem, menentukan parameter, dan mengeksekusi perintah pada satu atau beberapa instans. SSM Agent pada instans memproses perintah dan mengonfigurasikan proses seperti yang ditentukan.

Untuk mengunduh CloudWatch agen menggunakan Run Command

1. Buka konsol Systems Manager di https://console.aws.amazon.com/systems-manager/.
2. Pada panel navigasi, silakan pilih Perintah Eksekusi.

   -atau-

   Jika AWS Systems Manager halaman beranda terbuka, gulir ke bawah dan pilih Jelajahi Jalankan Perintah.
3. Pilih Jalankan perintah.
4. Dalam daftar dokumen Command, pilih AWS-configure AWSPackage.
5. Di area Target, pilih instance untuk menginstal CloudWatch agen. Jika Anda tidak melihat instans khusus, maka proses ini mungkin tidak dikonfigurasi untuk Perintah Proses. Untuk informasi selengkapnya, silakan lihat Mengatur AWS Systems Manager untuk Lingkungan Hibrid di Panduan Pengguna AWS Systems Manager .
6. Di daftar Tindakan pilih, Instal.
7. Di kotak Nama, masukkan `AmazonCloudWatchAgent`.
8. Simpan Versi diatur ke terbaru untuk melakukan instalasi versi terbaru agen.

9. Pilih Jalankan.
10. Secara opsional, dalam Target dan keluaran, pilih tombol yang ada di samping nama instans dan pilih Lihat output. Manajer Sistem harus menunjukkan bahwa agen berhasil diinstal.

(Opsional) Ubah konfigurasi umum dan profil bernama untuk CloudWatch agen

CloudWatch Agen menyertakan file konfigurasi yang disebut`common-config.toml`. Anda dapat menggunakan file ini, secara opsional, sebagai pilihan untuk menentukan informasi proksi dan Wilayah.

Di server yang menjalankan Linux, file ini ada di direktori `/opt/aws/amazon-cloudwatch-agent/etc`. Pada server yang menjalankan Server Windows, file ini berada di direktori `C:\ProgramData\Amazon\AmazonCloudWatchAgent`.

`common-config.toml` bawaan adalah sebagai berikut:

```
# This common-config is used to configure items used for both ssm and cloudwatch access


## Configuration for shared credential.
## Default credential strategy will be used if it is absent here:
##            Instance role is used for EC2 case by default.
##            AmazonCloudWatchAgent profile is used for onPremise case by default.
# [credentials]
#    shared_credential_profile = "{profile_name}"
#    shared_credential_file= "{file_name}"

## Configuration for proxy.
## System-wide environment-variable will be read if it is absent here.
## i.e. HTTP_PROXY/http_proxy; HTTPS_PROXY/https_proxy; NO_PROXY/no_proxy
## Note: system-wide environment-variable is not accessible when using ssm run-command.
## Absent in both here and environment-variable means no proxy will be used.
# [proxy]
#    http_proxy = "{http_url}"
#    https_proxy = "{https_url}"
#    no_proxy = "{domain}"
```

Semua baris pada awalnya berkomentar. Untuk mengatur pengaturan profil kredensial atau proksi, hapus # dari baris tersebut dan tentukan nilai. Anda dapat mengedit file ini secara manual, atau menggunakan Perintah Eksekusi `RunShellScript` dalam Systems Manager:

- `shared_credential_profile`— Untuk server lokal, baris ini menentukan profil kredensi pengguna IAM yang akan digunakan untuk mengirim data. CloudWatch Jika Anda tetap baris ini berkomentar, `AmazonCloudWatchAgent` digunakan.

  Pada instans EC2, Anda dapat menggunakan baris ini untuk meminta CloudWatch agen mengirim data dari instance ini ke CloudWatch AWS Wilayah yang berbeda. Untuk melakukan hal itu, tentukan profil dengan nama yang mencakup `region` bidang yang menetapkan nama Wilayah yang akan dikirim.

  Jika menentukan `shared_credential_profile`, Anda juga harus menghapus # dari awal `[credentials]` yang sesuai.

- `shared_credential_file` – Untuk meminta agen mencari kredensial dalam file yang terletak di jalur selain jalur default, tentukan jalur dan nama file lengkap di sini. Jalur default adalah `/root/.aws` di Linux dan `C:\\Users\\Administrator\\.aws` pada Server Windows.

  Contoh pertama di bawah ini menunjukkan sintaks baris `shared_credential_file` yang valid untuk server Linux, dan contoh kedua valid untuk Server Windows. Pada Server Windows, Anda harus menghindari karakter \.

  ```
  shared_credential_file= "/usr/username/credentials"
  ```

  ```
  shared_credential_file= "C:\\Documents and Settings\\username\\.aws\\credentials"
  ```

  Jika menentukan `shared_credential_file`, Anda juga harus menghapus # dari awal `[credentials]` yang sesuai.

- Pengaturan proxy – Jika server Anda menggunakan proxy HTTP atau HTTPS untuk menghubungi layanan AWS , tentukan proxy tersebut di bidang `http_proxy` dan `https_proxy`. Jika ada URL yang harus dikecualikan dari proxy, tentukan di bidang `no_proxy`, dipisahkan dengan koma.

Mulai CloudWatch agen

Anda dapat memulai agen menggunakan Perintah Operasi Manajer Sistem atau baris perintah.

Mulai CloudWatch agen menggunakan Systems Manager Run Command

Ikuti langkah-langkah ini untuk memulai agen menggunakan Perintah Proses Manajer Sistem.

Untuk memulai CloudWatch agen menggunakan Run Command

1. Buka konsol Systems Manager di https://console.aws.amazon.com/systems-manager/.
2. Pada panel navigasi, silakan pilih Perintah Eksekusi.

   -atau-

   Jika AWS Systems Manager halaman beranda terbuka, gulir ke bawah dan pilih Jelajahi Jalankan Perintah.
3. Pilih Jalankan perintah.
4. Dalam daftar dokumen Command, pilih AmazonCloudWatch- ManageAgent.
5. Di area Target, pilih instance tempat Anda menginstal CloudWatch agen.
6. Di daftar Tindakan pilih, konfigurasi.
7. Di daftar Sumber Konfigurasi Opsional pilih, ssm.
8. Di kotak Lokasi Konfigurasi Opsional, masukkan nama nama parameter Systems Manager dari file konfigurasi agen yang Anda buat dan simpan ke Systems Manager Parameter Store, seperti yang dijelaskan dalamBuat file konfigurasi CloudWatch agen.
9. Di daftar Mulai Ulang Opsional pilih, ya untuk memulai agen setelah Anda menyelesaikan langkah-langkah ini.
10. Pilih Jalankan.
11. Secara opsional, dalam Target dan keluaran, pilih tombol yang ada di samping nama instans dan pilih Lihat output. Manajer Sistem harus menunjukkan bahwa agen telah berhasil dimulai.

Mulai CloudWatch agen pada instans Amazon EC2 menggunakan baris perintah

Ikuti langkah-langkah ini untuk menggunakan baris perintah untuk menginstal CloudWatch agen pada instans Amazon EC2.

Untuk menggunakan baris perintah untuk memulai CloudWatch agen pada instans Amazon EC2

- Dalam perintah ini, `-a fetch-config` menyebabkan agen memuat versi terbaru dari file konfigurasi CloudWatch agen, dan `-s` memulai agen.

  Linux dan macOS: Jika Anda menyimpan file konfigurasi di Systems Manager Parameter Store, masukkan yang berikut ini:

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-
config -m ec2 -s -c ssm:configuration-parameter-store-name
```

Linux dan macOS: Jika Anda menyimpan file konfigurasi di komputer lokal, masukkan perintah berikut. Ganti *configuration-file-path*dengan path ke file konfigurasi agen. File ini disebut `config.json` Jika Anda membuatnya dengan pemandu, dan bisa disebut `amazon-cloudwatch-agent.json` jika Anda membuatnya secara manual.

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-
config -m ec2 -s -c file:configuration-file-path
```

Windows Server: Jika Anda menyimpan file konfigurasi agen di Systems Manager Parameter Store, masukkan yang berikut dari PowerShell konsol:

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -
a fetch-config -m ec2 -s -c ssm:configuration-parameter-store-name
```

Windows Server: Jika Anda menyimpan file konfigurasi agen di komputer lokal, masukkan yang berikut dari PowerShell konsol:

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1"
 -a fetch-config -m ec2 -s -c file:"C:\Program Files\Amazon\AmazonCloudWatchAgent
\config.json"
```

Menginstal CloudWatch agen di server lokal

Jika Anda telah mengunduh CloudWatch agen di satu komputer dan membuat file konfigurasi agen yang Anda inginkan, Anda dapat menggunakan file konfigurasi tersebut untuk menginstal agen di server lokal lainnya.

Unduh CloudWatch agen di server lokal

Anda dapat mengunduh paket CloudWatch agen menggunakan Systems Manager Run Command atau tautan unduhan Amazon S3. Untuk informasi tentang penggunaan tautan unduhan Amazon S3, silakan lihat Unduh paket CloudWatch agen.

Unduh Menggunakan Systems Manager

Untuk menggunakan Perintah Eksekusi Systems Manager, Anda harus mendaftarkan server on-premise Anda dengan Amazon EC2 Systems Manager. Untuk informasi selengkapnya, silakan lihat Menyiapkan Systems Manager dalam Lingkungan Hibrid dalam AWS Systems Manager Panduan Pengguna.

Jika Anda sudah mendaftarkan server Anda, perbarui Agen SSM ke versi terbaru.

Untuk informasi tentang memperbarui SSM Agent di server yang menjalankan Linux, silakan lihat Instal Agen SSM untuk Lingkungan Hibrid (Linux) di AWS Systems Manager Panduan Pengguna.

Untuk informasi tentang memperbarui Agen SSM di server yang menjalankan Windows Server, silakan lihat Instal Agen SSM untuk Lingkungan Hibrid (Windows) di AWS Systems Manager Panduan Pengguna .

Untuk menggunakan Agen SSM untuk mengunduh paket CloudWatch agen di server lokal

1. Buka konsol Systems Manager di https://console.aws.amazon.com/systems-manager/.
2. Pada panel navigasi, silakan pilih Perintah Eksekusi.

   -atau-

   Jika AWS Systems Manager halaman beranda terbuka, gulir ke bawah dan pilih Jelajahi Jalankan Perintah.
3. Pilih Jalankan perintah.
4. Dalam daftar dokumen Command, pilih tombol di sebelah AWS-configure AWSPackage.
5. Di area Target, pilih server untuk menginstal CloudWatch agen. Jika Anda tidak melihat server tertentu, server tersebut mungkin tidak dikonfigurasikan untuk Perintah Proses. Untuk informasi selengkapnya, silakan lihat Mengatur AWS Systems Manager untuk Lingkungan Hibrid di Panduan Pengguna AWS Systems Manager .
6. Di daftar Tindakan pilih, Instal.
7. Di kotak Nama, masukkan `AmazonCloudWatchAgent`.
8. Simpan Versi kosong untuk melakukan instalasi versi terbaru agen.
9. Pilih Jalankan.

   Paket agen diunduh, dan langkah selanjutnya adalah mengonfigurasi dan memulainya.

(Menginstal di server lokal) Tentukan kredensional IAM dan Wilayah AWS

Untuk mengaktifkan CloudWatch agen mengirim data dari server lokal, Anda harus menentukan kunci akses dan kunci rahasia pengguna IAM yang Anda buat sebelumnya. Untuk informasi selengkapnya tentang membuat pengguna ini, silakan lihat Buat peran IAM dan pengguna untuk digunakan dengan agen CloudWatch .

Anda juga harus menentukan AWS Wilayah untuk mengirim metrik ke, menggunakan `region` bidang.

Berikut ini adalah contoh file ini.

```
[AmazonCloudWatchAgent]
aws_access_key_id=my_access_key
aws_secret_access_key=my_secret_key
region = us-west-1
```

Untuk *my_access_key* dan *my_secret_key*, gunakan kunci dari pengguna IAM yang tidak memiliki izin untuk menulis ke Systems Manager Parameter Store. Untuk informasi selengkapnya tentang pengguna IAM yang diperlukan untuk CloudWatch agen, lihatBuat pengguna IAM untuk digunakan dengan CloudWatch agen di server lokal.

Jika Anda namai profil ini `AmazonCloudWatchAgent`, Anda tidak perlu melakukan apa pun lagi. Atau, Anda dapat memberikan nama yang berbeda dan menentukan nama tersebut sebagai nilai untuk `shared_credential_profile` dalam `common-config.toml` file, yang dijelaskan di bagian berikut.

Berikut ini adalah contoh menggunakan aws configure perintah untuk membuat profil bernama untuk CloudWatch agen. Contoh ini mengasumsikan bahwa Anda sedang menggunakan nama profil bawaan dari `AmazonCloudWatchAgent`.

Untuk membuat AmazonCloudWatchAgent profil untuk CloudWatch agen

1. Jika Anda belum melakukannya, instal AWS Command Line Interface di server. Untuk informasi selengkapnya, silakan lihat Menginstal AWS CLI.
2. Pada server Linux, masukkan perintah berikut dan ikuti petunjuknya:

   ```
   sudo aws configure --profile AmazonCloudWatchAgent
   ```

Di Windows Server, buka PowerShell sebagai administrator, masukkan perintah berikut, dan ikuti petunjuknya.

```
aws configure --profile AmazonCloudWatchAgent
```

(Opsional) Memodifikasi konfigurasi umum dan profil bernama untuk agen CloudWatch

CloudWatch Agen menyertakan file konfigurasi yang disebut`common-config.toml`. Secara opsional, Anda dapat menggunakan file ini sebagai pilihan untuk menentukan informasi proksi dan Wilayah.

Di server yang menjalankan Linux, file ini ada di `/opt/aws/amazon-cloudwatch-agent/etc` direktori. Pada server yang menjalankan Server Windows, file ini berada di direktori `C:\ProgramData\Amazon\AmazonCloudWatchAgent`.

`common-config.toml` bawaan adalah sebagai berikut:

```
# This common-config is used to configure items used for both ssm and cloudwatch access


## Configuration for shared credential.
## Default credential strategy will be used if it is absent here:
##            Instance role is used for EC2 case by default.
##            AmazonCloudWatchAgent profile is used for onPremise case by default.
# [credentials]
#    shared_credential_profile = "{profile_name}"
#    shared_credential_file= "{file_name}"

## Configuration for proxy.
## System-wide environment-variable will be read if it is absent here.
## i.e. HTTP_PROXY/http_proxy; HTTPS_PROXY/https_proxy; NO_PROXY/no_proxy
## Note: system-wide environment-variable is not accessible when using ssm run-command.
## Absent in both here and environment-variable means no proxy will be used.
# [proxy]
#    http_proxy = "{http_url}"
#    https_proxy = "{https_url}"
#    no_proxy = "{domain}"
```

Semua baris pada awalnya berkomentar. Untuk mengatur pengaturan profil kredensial atau proksi, hapus # dari baris tersebut dan tentukan nilai. Anda dapat mengedit file ini secara manual, atau menggunakan Perintah Eksekusi `RunShellScript` dalam Systems Manager:

- `shared_credential_profile`— Untuk server lokal, baris ini menentukan profil kredensi pengguna IAM yang akan digunakan untuk mengirim data. CloudWatch Jika Anda tetap baris ini berkomentar, `AmazonCloudWatchAgent` digunakan. Untuk informasi selengkapnya tentang membuat profil ini, silakan lihat (Menginstal di server lokal) Tentukan kredensional IAM dan Wilayah AWS.

  Pada instans EC2, Anda dapat menggunakan baris ini untuk meminta CloudWatch agen mengirim data dari instance ini ke CloudWatch AWS Wilayah yang berbeda. Untuk melakukan hal itu, tentukan profil dengan nama yang mencakup `region` bidang yang menetapkan nama Wilayah yang akan dikirim.

  Jika menentukan `shared_credential_profile`, Anda juga harus menghapus # dari awal `[credentials]` yang sesuai.
- `shared_credential_file` – Untuk meminta agen mencari kredensial dalam file yang terletak di jalur selain jalur default, tentukan jalur dan nama file lengkap di sini. Jalur default adalah `/root/.aws` di Linux dan `C:\\Users\\Administrator\\.aws` pada Server Windows.

  Contoh pertama di bawah ini menunjukkan sintaks baris `shared_credential_file` yang valid untuk server Linux, dan contoh kedua valid untuk Server Windows. Pada Server Windows, Anda harus menghindari karakter \.

  ```
  shared_credential_file= "/usr/username/credentials"
  ```

  ```
  shared_credential_file= "C:\\Documents and Settings\\username\\.aws\\credentials"
  ```

  Jika menentukan `shared_credential_file`, Anda juga harus menghapus # dari awal `[credentials]` yang sesuai.
- Pengaturan proxy – Jika server Anda menggunakan proxy HTTP atau HTTPS untuk menghubungi layanan AWS , tentukan proxy tersebut di bidang `http_proxy` dan `https_proxy`. Jika ada URL yang harus dikecualikan dari proxy, tentukan di bidang `no_proxy`, dipisahkan dengan koma.

Memulai CloudWatch agen

Anda dapat memulai CloudWatch agen menggunakan Systems Manager Run Command atau command line.

Menggunakan Agen SSM untuk memulai CloudWatch agen di server lokal

1. Buka konsol Systems Manager di https://console.aws.amazon.com/systems-manager/.
2. Pada panel navigasi, silakan pilih Perintah Eksekusi.

   -atau-

   Jika AWS Systems Manager halaman beranda terbuka, gulir ke bawah dan pilih Jelajahi Jalankan Perintah.
3. Pilih Jalankan perintah.
4. Dalam daftar dokumen Perintah, pilih tombol di sebelah AmazonCloudWatch- ManageAgent.
5. Di area Target, pilih contoh tempat Anda melakukan instalasi agen.
6. Di daftar Tindakan pilih, konfigurasi.
7. Di daftar Mode, pilih onPremise.
8. Pada Optional Configuration Location masukkan nama file konfigurasi agen yang Anda buat dengan pemandu dan simpan di Parameter Store.
9. Pilih Jalankan.

   Agen memulai dengan konfigurasi yang Anda tentukan dalam file konfigurasi.

Untuk menggunakan baris perintah untuk memulai CloudWatch agen di server lokal

- Dalam perintah ini, `-a fetch-config` menyebabkan agen memuat versi terbaru dari file konfigurasi CloudWatch agen, dan `-s` memulai agen.

  Linux: Jika Anda menyimpan file konfigurasi di Systems Manager Parameter Store, masukkan yang berikut ini:

  ```
  sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-
  config -m onPremise -s -c ssm:configuration-parameter-store-name
  ```

  Linux: Jika Anda menyimpan file konfigurasi di komputer lokal, masukkan perintah berikut. Ganti *configuration-file-path*dengan path ke file konfigurasi agen. File ini disebut

`config.json` Jika Anda membuatnya dengan pemandu, dan bisa disebut `amazon-cloudwatch-agent.json` jika Anda membuatnya secara manual.

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m onPremise -s -c file:configuration-file-path
```

Windows Server: Jika Anda menyimpan file konfigurasi agen di Systems Manager Parameter Store, masukkan yang berikut dari PowerShell konsol:

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -a fetch-config -m onPremise -s -c ssm:configuration-parameter-store-name
```

Windows Server: Jika Anda menyimpan file konfigurasi agen di komputer lokal, masukkan yang berikut ini dari PowerShell konsol. Ganti *configuration-file-path*dengan path ke file konfigurasi agen. File ini disebut `config.json` Jika Anda membuatnya dengan pemandu, dan bisa disebut `amazon-cloudwatch-agent.json` jika Anda membuatnya secara manual.

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -a fetch-config -m onPremise -s -c file:configuration-file-path
```

Menginstal CloudWatch agen pada instance baru menggunakan AWS CloudFormation

Amazon telah mengunggah beberapa AWS CloudFormation templat GitHub untuk membantu Anda menginstal dan memperbarui CloudWatch agen pada instans Amazon EC2 baru. Untuk informasi selengkapnya tentang penggunaan AWS CloudFormation, lihat Apa itu AWS CloudFormation? .

Lokasi template adalah Menyebarkan CloudWatch agen Amazon ke instans EC2 menggunakan. AWS CloudFormation Lokasi ini mencakup direktori `inline` dan `ssm`. Setiap direktori ini berisi template untuk instans Linux dan Windows.

- Template dalam `inline` direktori memiliki konfigurasi CloudWatch agen yang disematkan ke dalam AWS CloudFormation template. Secara bawaan, template Linux mengumpulkan metrik `mem_used_percent` dan `swap_used_percent`, dan template Windows mengumpulkan `Memory % Committed Bytes In Use` dan `Paging File % Usage`.

Untuk mengubah template ini guna mengumpulkan metrik yang berbeda, ubah bagian template berikut. Contoh berikut berasal dari template untuk server Linux. Ikuti format dan sintaksis file konfigurasi agen untuk membuat perubahan ini. Untuk informasi selengkapnya, lihat Buat atau edit file konfigurasi CloudWatch agen secara manual.

```
{
   "metrics":{
      "append_dimensions":{
         "AutoScalingGroupName":"${!aws:AutoScalingGroupName}",
         "ImageId":"${!aws:ImageId}",
         "InstanceId":"${!aws:InstanceId}",
         "InstanceType":"${!aws:InstanceType}"
      },
      "metrics_collected":{
         "mem":{
            "measurement":[
               "mem_used_percent"
            ]
         },
         "swap":{
            "measurement":[
               "swap_used_percent"
            ]
         }
      }
   }
}
```

> **Note**
>
> Pada template inline, semua variabel placeholder harus memiliki tanda seru (!) sebelum mereka sebagai karakter lolos. Anda dapat melihat ini di template contoh. Jika Anda menambahkan variabel placeholder lainnya, pastikan menambahkan tanda seru di depan nama.

- Templat dalam `ssm` direktori memuat file konfigurasi agen dari Parameter Store. Untuk menggunakan template ini, Anda harus terlebih dahulu membuat file konfigurasi dan mengunggahnya ke Parameter Store. Kemudian, Anda memberikan Parameter Store nama file

dalam templat. Anda dapat membuat file konfigurasi secara manual atau dengan menggunakan pemandu. Untuk informasi selengkapnya, lihat Buat file konfigurasi CloudWatch agen.

Anda dapat menggunakan kedua jenis templat untuk menginstal CloudWatch agen dan untuk memperbarui konfigurasi agen.

Tutorial: Instal dan konfigurasikan CloudWatch agen menggunakan AWS CloudFormation template inline

Tutorial ini memandu Anda menggunakan AWS CloudFormation untuk menginstal CloudWatch agen pada instans Amazon EC2 baru. Tutorial ini melakukan instalasi pada instans baru yang menjalankan Amazon Linux 2 menggunakan template inline, yang tidak memerlukan penggunaan file konfigurasi JSON atau Parameter Store. Templat inline mencakup konfigurasi agen dalam templat. Dalam tutorial ini, Anda menggunakan konfigurasi agen default yang terkandung dalam template.

Setelah prosedur untuk memasang agen, tutorial berlanjut dengan cara memperbarui agen.

Untuk digunakan AWS CloudFormation untuk menginstal CloudWatch agen pada instance baru

1. Unduh template dari GitHub. Dalam tutorial ini, unduh template inline untuk Amazon Linux 2 sebagai berikut:

   ```
   curl -O https://raw.githubusercontent.com/awslabs/aws-cloudformation-templates/
   master/aws/solutions/AmazonCloudWatchAgent/inline/amazon_linux.template
   ```

2. Buka AWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation.
3. Pilih Create stack.
4. Untuk Pilih template, pilih Unggah template ke Amazon S3, pilih template yang diunduh, dan pilih Berikutnya.
5. Pada halaman Tentukan Detail, isilah parameter-parameter berikut dan kemudian pilih Berikutnya:

   - Nama tumpukan: Pilih nama tumpukan untuk AWS CloudFormation tumpukan Anda.
   - IamRole: Pilih peran IAM yang memiliki izin untuk menulis CloudWatch metrik, log, dan jejak. Untuk informasi selengkapnya, lihat Buat peran IAM untuk digunakan dengan CloudWatch agen di instans Amazon EC2.
   - InstanceAMI: Pilih AMI yang valid di Wilayah tempat Anda akan meluncurkan tumpukan Anda.
   - InstanceType: Pilih jenis instance yang valid.

- KeyName: Untuk mengaktifkan akses SSH ke instans baru, pilih key pair Amazon EC2 yang sudah ada. Jika Anda belum memiliki pasangan kunci Amazon EC2, maka Anda dapat membuatnya di AWS Management Console. Untuk informasi selengkapnya, silakan lihat Pasangan Kunci Amazon EC2 dalam Panduan Pengguna Amazon EC2 untuk Instans Linux.
- SSHLocation: Menentukan rentang alamat IP yang dapat digunakan untuk terhubung ke instans menggunakan SSH. Standar tersebut memungkinkan akses dari alamat IP mana pun.

6. Di halaman Opsi Anda dapat memilih untuk menandai sumber daya tumpukan Anda. Pilih Berikutnya.
7. Pada halamanPeninjauan, periksa informasi Anda, akui bahwa tumpukan mungkin membuat sumber daya IAM, kemudian pilih Buat.

   Jika menyegarkan konsol, Anda melihat bahwa tumpukan baru memiliki `CREATE_IN_PROGRESS` status.
8. Saat instans dibuat, Anda dapat melihatnya di konsol Amazon EC2. Atau, Anda dapat terhubung ke host dan memeriksa kemajuannya.

   Gunakan perintah berikut untuk mengonfirmasi bahwa agen diinstal:

   ```
   rpm -qa amazon-cloudwatch-agent
   ```

   Gunakan perintah berikut untuk mengonfirmasi bahwa agen menjalankan:

   ```
   ps aux | grep amazon-cloudwatch-agent
   ```

Prosedur selanjutnya menunjukkan penggunaan AWS CloudFormation untuk memperbarui CloudWatch agen menggunakan template inline. Templat inline bawaan mengumpulkan `mem_used_percent` metrik. Dalam tutorial ini, Anda mengubah konfigurasi agen untuk berhenti mengumpulkan metrik tersebut.

Untuk digunakan AWS CloudFormation untuk memperbarui CloudWatch agen

1. Dalam template yang Anda unduh dalam prosedur sebelumnya, hapus baris berikut lalu simpan template:

   ```
   "mem": {
   ```

```
    "measurement": [
        "mem_used_percent"
      ]
},
```

2. Buka AWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation.
3. Di AWS CloudFormation dasbor, pilih tumpukan yang Anda buat dan pilih Update Stack.
4. Untuk Pilih Template, pilih Unggah template ke Amazon S3, pilih template yang Anda ubah, dan pilih Berikutnya.
5. Pada halaman Option, pilih Berikutnya dan kemudian Berikutnya.
6. Pada halaman Review, periksa informasi Anda dan pilih Update.

   Setelah beberapa saat, Anda akan melihat `UPDATE_COMPLETE`.

Tutorial: Instal CloudWatch agen menggunakan AWS CloudFormation dan Parameter Store

Tutorial ini memandu Anda menggunakan AWS CloudFormation untuk menginstal CloudWatch agen pada instans Amazon EC2 baru. Tutorial ini melakukan instalasi di instans baru yang menjalankan Amazon Linux 2 menggunakan file konfigurasi agen yang Anda buat dan simpan di Parameter Store.

Setelah prosedur untuk menginstal agen, tutorial berlanjut dengan cara memperbarui agen.

Untuk menggunakan AWS CloudFormation untuk menginstal CloudWatch agen pada instance baru menggunakan konfigurasi dari Parameter Store

1. Jika Anda belum melakukannya, unduh paket CloudWatch agen ke salah satu komputer Anda sehingga Anda dapat membuat file konfigurasi agen. Untuk informasi selengkapnya dan mengunduh agen menggunakan Parameter Store, silakan lihat Unduh dan konfigurasikan CloudWatch agen. Untuk informasi selengkapnya tentang mengunduh paket menggunakan baris perintah, silakan lihat Unduh dan konfigurasikan CloudWatch agen menggunakan baris perintah.
2. Buat file konfigurasi agen dan simpan di Parameter Store. Untuk informasi selengkapnya, lihat Buat file konfigurasi CloudWatch agen.
3. Unduh template dari GitHub sebagai berikut:

```
curl -O https://raw.githubusercontent.com/awslabs/aws-cloudformation-templates/
master/aws/solutions/AmazonCloudWatchAgent/ssm/amazon_linux.template
```

4. Buka AWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation.
5. Pilih Create stack.
6. Untuk Pilih template, pilih Unggah template ke Amazon S3, pilih template yang Anda unduh, lalu pilih Berikutnya.
7. Pada halaman Tentukan Detail, isilah parameter-parameter berikut dengan sesuai dan pilih Berikutnya:

   - Nama tumpukan: Pilih nama tumpukan untuk AWS CloudFormation tumpukan Anda.
   - IamRole: Pilih peran IAM yang memiliki izin untuk menulis CloudWatch metrik, log, dan jejak. Untuk informasi selengkapnya, lihat Buat peran IAM untuk digunakan dengan CloudWatch agen di instans Amazon EC2.
   - InstanceAMI: Pilih AMI yang valid di Wilayah tempat Anda akan meluncurkan tumpukan Anda.
   - InstanceType: Pilih jenis instance yang valid.
   - KeyName: Untuk mengaktifkan akses SSH ke instans baru, pilih key pair Amazon EC2 yang sudah ada. Jika Anda belum memiliki pasangan kunci Amazon EC2, maka Anda dapat membuatnya di AWS Management Console. Untuk informasi selengkapnya, silakan lihat Pasangan Kunci Amazon EC2 dalam Panduan Pengguna Amazon EC2 untuk Instans Linux.
   - SSHLocation: Menentukan rentang alamat IP yang dapat digunakan untuk terhubung ke instans menggunakan SSH. Standar tersebut memungkinkan akses dari alamat IP mana pun.
   - SSMKey: Menentukan file konfigurasi agen yang Anda buat dan simpan di Parameter Store.
8. Di halaman Opsi Anda dapat memilih untuk menandai sumber daya tumpukan Anda. Pilih Berikutnya.
9. Pada halamanPeninjauan, periksa informasi Anda, akui bahwa tumpukan mungkin membuat sumber daya IAM, kemudian pilih Buat.

   Jika menyegarkan konsol, Anda melihat bahwa tumpukan baru memiliki `CREATE_IN_PROGRESS` status.
10. Saat instans dibuat, Anda dapat melihatnya di konsol Amazon EC2. Atau, Anda dapat terhubung ke host dan memeriksa kemajuannya.

    Gunakan perintah berikut untuk mengonfirmasi bahwa agen diinstal:

    ```
    rpm -qa amazon-cloudwatch-agent
    ```

    Gunakan perintah berikut untuk mengonfirmasi bahwa agen menjalankan:

```
ps aux | grep amazon-cloudwatch-agent
```

Prosedur selanjutnya menunjukkan penggunaan AWS CloudFormation untuk memperbarui CloudWatch agen, menggunakan konfigurasi agen yang Anda simpan di Parameter Store.

Untuk digunakan AWS CloudFormation untuk memperbarui CloudWatch agen menggunakan konfigurasi di Parameter Store

1. Ubah file konfigurasi agen yang disimpan di Parameter Store ke konfigurasi baru yang Anda inginkan.
2. Di AWS CloudFormation templat yang Anda unduh dalam the section called “Tutorial: Instal CloudWatch agen menggunakan AWS CloudFormation dan Parameter Store” topik, ubah nomor versi. Misalnya, Anda mungkin mengubah `VERSION=1.0` untuk `VERSION=2.0`.
3. Buka AWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation.
4. Di AWS CloudFormation dasbor, pilih tumpukan yang Anda buat dan pilih Update Stack.
5. Untuk Memilih Template, pilih Unggah template ke Amazon S3, pilih template yang baru saja Anda modifikasi, dan pilih Berikutnya.
6. Pada halaman Option, pilih Berikutnya dan kemudian Berikutnya.
7. Pada halaman Review, periksa informasi Anda dan pilih Update.

   Setelah beberapa saat, Anda akan melihat `UPDATE_COMPLETE`.

Memecahkan masalah pemasangan agen dengan CloudWatch AWS CloudFormation

Bagian ini membantu Anda memecahkan masalah dengan menginstal dan memperbarui CloudWatch agen menggunakan. AWS CloudFormation

Mendeteksi saat ada pembaruan yang mengalami kegagalan

Jika Anda menggunakannya AWS CloudFormation untuk memperbarui konfigurasi CloudWatch agen, dan menggunakan konfigurasi yang tidak valid, agen akan berhenti mengirim metrik apa pun. CloudWatch Cara cepat untuk memeriksa apakah pembaruan konfigurasi agen berhasil adalah dengan melihat file `cfn-init-cmd.log`. Pada server Linux, file tersebut terletak di `/var/log/cfn-init-cmd.log`. Pada instans Windows, file tersebut terletak di `C:\cfn\log\cfn-init-cmd.log`.

Metrik tidak ada

Jika Anda tidak melihat metrik yang Anda harapkan untuk dilihat setelah melakukan instalasi atau memperbarui agen, pastikan bahwa agen dikonfigurasi untuk mengumpulkan metrik tersebut. Untuk melakukan hal itu, periksa `amazon-cloudwatch-agent.json` untuk memastikan bahwa metrik dicantumkan, dan periksa bahwa Anda mencari di namespace metrik yang benar. Untuk informasi selengkapnya, lihat CloudWatch file dan lokasi agen.

Memverifikasi tanda tangan paket CloudWatch agen

File tanda tangan GPG disertakan untuk paket CloudWatch agen di server Linux. Anda dapat menggunakan sebuah kunci publik untuk memverifikasi apakah file unduhan agen adalah asli dan tidak dimodifikasi.

Untuk Server Windows, Anda dapat menggunakan MSI untuk memverifikasi tanda tangan tersebut.

Untuk komputer macOS, tanda tangan disertakan dalam paket unduhan agen.

Untuk menemukan file tanda tangan yang benar, silakan lihat tabel berikut. Untuk setiap arsitektur dan sistem operasi, terdapat tautan umum serta tautan untuk setiap Wilayah. Misalnya, untuk Amazon Linux 2023 dan Amazon Linux 2 dan arsitektur x86-64, tiga tautan yang valid adalah:

- `https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig`
- `https://amazoncloudwatch-agent-us-east-1.s3.us-east-1.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm`
- `https://amazoncloudwatch-agent-eu-central-1.s3.eu-central-1.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm`

> **Note**
>
> Untuk mengunduh CloudWatch agen, koneksi Anda harus menggunakan TLS 1.2 atau yang lebih baru.

Arsitektur	Platform	Tautan unduhan	Tautan file tanda tangan
x86-64	Amazon Linux 2023 dan Amazon Linux 2	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/ amazon-cloudwatch-agent .rpm https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/amazon_linux/amd64/terbaru/* .rpm amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/amazon_linux/amd64/latest/* amazon-cloudwatch-agent .rpm.sig
x86-64	Centos	https://amazoncloudwatch-agent.s3.amazonaws.com/centos/amd64/latest/ amazon-cloudwatch-agent .rpm https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/centos/amd64/latest/* .rpm amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/centos/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/centos/amd64/latest/* amazon-cloudwatch-agent .rpm.sig
x86-64	Redhat	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/amd64/latest/ amazon-cloudwatch-agent .rpm https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/redhat/amd64*	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/redhat/amd64/*

Arsitektur	Platform	Tautan unduhan	Tautan file tanda tangan
		/latest/ .rpm amazon-cloudwatch-agent	*latest/* amazon-cloudwatch-agent .rpm.sig
x86-64	SEGAR	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/amd64/latest/ amazon-cloudwatch-agent .rpm https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/suse/amd64/latest/* .rpm amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/suse/amd64/latest/* amazon-cloudwatch-agent .rpm.sig
x86-64	Debian	https://amazoncloudwatch-agent.s3.amazonaws.com/debian/amd64/latest/ amazon-cloudwatch-agent .deb https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/debian/amd64/latest/* .deb amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/debian/amd64/latest/ amazon-cloudwatch-agent .deb.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/debian/amd64/latest/* amazon-cloudwatch-agent .deb.sig

Arsitektur	Platform	Tautan unduhan	Tautan file tanda tangan
x86-64	Ubuntu	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/amd64/latest/ amazon-cloudwatch-agent .deb https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/ubuntu/amd64/latest/* .deb amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/amd64/latest/ amazon-cloudwatch-agent .deb.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/ubuntu/amd64/latest/* amazon-cloudwatch-agent .deb.sig
x86-64	Oracle	https://amazoncloudwatch-agent.s3.amazonaws.com/oracle_linux/amd64/latest/ amazon-cloudwatch-agent .rpm https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/oracle_linux/amd64/latest/* .rpm amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/oracle_linux/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/oracle_linux/amd64/latest/* amazon-cloudwatch-agent .rpm.sig
x86-64	macOS	https://amazoncloudwatch-agent.s3.amazonaws.com/darwin/amd64/latest/ amazon-cloudwatch-agent .pkg https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/darwin/amd64/latest/* amazon-cloudwatch-agent .pkg	https://amazoncloudwatch-agent.s3.amazonaws.com/darwin/amd64/latest/ amazon-cloudwatch-agent .pkg.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/darwin/amd64/latest/* amazon-cloudwatch-agent .pkg.sig

Arsitektur	Platform	Tautan unduhan	Tautan file tanda tangan
x86-64	Jendela	https://amazoncloudwatch-agent.s3.amazonaws.com/windows/amd64/latest/ amazon-cloudwatch-agent .msi https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/windows/amd64/latest/* amazon-cloudwatch-agent .msi	https://amazoncloudwatch-agent.s3.amazonaws.com/windows/amd64/latest/ amazon-cloudwatch-agent .msi.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/windows/amd64/latest/* amazon-cloudwatch-agent .msi.sig
ARM64	Amazon Linux 2023 dan Amazon Linux 2	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/arm64/latest/ amazon-cloudwatch-agent .rpm https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/amazon_linux/arm64/latest/* .rpm amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/arm64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/amazon_linux/arm64/latest/* amazon-cloudwatch-agent .rpm.sig

Arsitektur	Platform	Tautan unduhan	Tautan file tanda tangan
ARM64	Redhat	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/arm64/latest/ amazon-cloudwatch-agent .rpm https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/redhat/arm64/latest/* .rpm amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/arm64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/redhat/arm64/latest/* amazon-cloudwatch-agent .rpm.sig
ARM64	Ubuntu	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/arm64/latest/ amazon-cloudwatch-agent .deb https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/ubuntu/arm64/latest/* .deb amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/arm64/latest/ amazon-cloudwatch-agent .deb.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/ubuntu/arm64/latest/* amazon-cloudwatch-agent .deb.sig
ARM64	SEGAR	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/arm64/latest/ amazon-cloudwatch-agent .rpm https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/suse/arm64/latest/* .rpm amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/arm64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent - *wilayah* .s3. *wilayah .amazonaws.com/suse/arm64/latest/* amazon-cloudwatch-agent .rpm.sig

Untuk memverifikasi paket CloudWatch agen di server Linux

1. Unduh kunci publik.

```
shell$ wget https://amazoncloudwatch-agent.s3.amazonaws.com/assets/amazon-
cloudwatch-agent.gpg
```

2. Impor kunci publik ke dalam keyring Anda.

```
shell$  gpg --import amazon-cloudwatch-agent.gpg
gpg: key 3B789C72: public key "Amazon CloudWatch Agent" imported
gpg: Total number processed: 1
gpg: imported: 1 (RSA: 1)
```

   Catat nilai utama, saat Anda membutuhkannya pada langkah berikutnya. Dalam contoh sebelumnya, nilai utama adalah 3B789C72.

3. Verifikasi sidik jari dengan menjalankan perintah berikut, mengganti *key-value* dengan nilai dari langkah sebelumnya:

```
shell$  gpg --fingerprint key-value
pub   2048R/3B789C72 2017-11-14
      Key fingerprint = 9376 16F3 450B 7D80 6CBD  9725 D581 6730 3B78 9C72
uid                  Amazon CloudWatch Agent
```

   String sidik jari harus sama dengan berikut ini:

   ```
   9376 16F3 450B 7D80 6CBD 9725 D581 6730 3B78 9C72
   ```

   Jika string sidik jari tidak cocok, jangan instal agen. Hubungi Layanan Web Amazon.

   Setelah Anda memverifikasi sidik jari, Anda dapat menggunakannya untuk memverifikasi tanda tangan paket CloudWatch agen.

4. Unduh file tanda tangan paket menggunakan wget. Untuk menentukan file tanda tangan yang benar, silakan lihat tabel sebelumnya.

```
wget Signature File Link
```

5. Untuk memverifikasi tanda tangan, jalankan gpg --verify.

```
shell$ gpg --verify signature-filename agent-download-filename
```

```
gpg: Signature made Wed 29 Nov 2017 03:00:59 PM PST using RSA key ID 3B789C72
gpg: Good signature from "Amazon CloudWatch Agent"
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: 9376 16F3 450B 7D80 6CBD  9725 D581 6730 3B78 9C72
```

Jika output mencakup frasa `BAD signature`, periksa apakah Anda melakukan prosedur dengan benar. Jika Anda terus mendapatkan respons ini, hubungi Layanan Web Amazon dan hindari menggunakan file yang diunduh.

Catat peringatan tentang kepercayaan. Kunci hanya dapat dipercaya jika Anda atau seseorang yang Anda percaya telah menandatanganinya. Ini tidak berarti bahwa tanda pembeda Anda tidak valid, hanya saja Anda belum memverifikasi kunci publiknya.

Untuk memverifikasi paket CloudWatch agen pada server yang menjalankan Windows Server

1. Unduh dan instal GnuPG untuk Windows dari https://gnupg.org/download/. Saat menginstal, sertakan opsi Shell Extension (GpgEx).

   Anda dapat melakukan langkah-langkah yang tersisa di Windows PowerShell.

2. Unduh kunci publik.

   ```
   PS> wget https://amazoncloudwatch-agent.s3.amazonaws.com/assets/amazon-cloudwatch-
   agent.gpg -OutFile amazon-cloudwatch-agent.gpg
   ```

3. Impor kunci publik ke dalam keyring Anda.

   ```
   PS>  gpg --import amazon-cloudwatch-agent.gpg
   gpg: key 3B789C72: public key "Amazon CloudWatch Agent" imported
   gpg: Total number processed: 1
   gpg: imported: 1 (RSA: 1)
   ```

   Catat nilai kunci karena Anda membutuhkannya di langkah berikutnya. Dalam contoh sebelumnya, nilai utama adalah `3B789C72`.

4. Verifikasi sidik jari dengan menjalankan perintah berikut, mengganti *`key-value`* dengan nilai dari langkah sebelumnya:

   ```
   PS>  gpg --fingerprint key-value
   pub   rsa2048 2017-11-14 [SC]
   ```

```
      9376 16F3 450B 7D80 6CBD  9725 D581 6730 3B78 9C72
uid           [ unknown] Amazon CloudWatch Agent
```

String sidik jari harus sama dengan berikut ini:

```
9376 16F3 450B 7D80 6CBD 9725 D581 6730 3B78 9C72
```

Jika string sidik jari tidak cocok, jangan instal agen. Hubungi Layanan Web Amazon.

Setelah Anda memverifikasi sidik jari, Anda dapat menggunakannya untuk memverifikasi tanda tangan paket CloudWatch agen.

5. Unduh file tanda tangan paket menggunakan wget. Untuk menentukan file tanda tangan yang benar, lihat Tautan Unduhan CloudWatch Agen.
6. Untuk memverifikasi tanda tangan, jalankan gpg --verify.

```
PS> gpg --verify sig-filename agent-download-filename
gpg: Signature made 11/29/17 23:00:45 Coordinated Universal Time
gpg:                using RSA key D58167303B789C72
gpg: Good signature from "Amazon CloudWatch Agent" [unknown]
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: 9376 16F3 450B 7D80 6CBD  9725 D581 6730 3B78 9C72
```

Jika output mencakup frasa `BAD signature`, periksa apakah Anda melakukan prosedur dengan benar. Jika Anda terus mendapatkan respons ini, hubungi Layanan Web Amazon dan hindari menggunakan file yang diunduh.

Catat peringatan tentang kepercayaan. Kunci hanya dapat dipercaya jika Anda atau seseorang yang Anda percaya telah menandatanganinya. Ini tidak berarti bahwa tanda pembeda Anda tidak valid, hanya saja Anda belum memverifikasi kunci publiknya.

Untuk memverifikasi paket CloudWatch agen di komputer macOS

- Ada dua metode untuk verifikasi tanda tangan di macOS.
  - Verifikasi sidik jari dengan menjalankan perintah berikut.

```
pkgutil --check-signature amazon-cloudwatch-agent.pkg
```

Anda akan melihat hasil yang mirip dengan berikut ini.

```
Package "amazon-cloudwatch-agent.pkg":
       Status: signed by a developer certificate issued by Apple for
 distribution
       Signed with a trusted timestamp on: 2020-10-02 18:13:24 +0000
       Certificate Chain:
       1. Developer ID Installer: AMZN Mobile LLC (94KV3E626L)
       Expires: 2024-10-18 22:31:30 +0000
       SHA256 Fingerprint:
       81 B4 6F AF 1C CA E1 E8 3C 6F FB 9E 52 5E 84 02 6E 7F 17 21 8E FB
       0C 40 79 13 66 8D 9F 1F 10 1C

 ------------------------------------------------------------------------
       2. Developer ID Certification Authority
       Expires: 2027-02-01 22:12:15 +0000
       SHA256 Fingerprint:
       7A FC 9D 01 A6 2F 03 A2 DE 96 37 93 6D 4A FE 68 09 0D 2D E1 8D 03
       F2 9C 88 CF B0 B1 BA 63 58 7F

 ------------------------------------------------------------------------
       3. Apple Root CA
       Expires: 2035-02-09 21:40:36 +0000
       SHA256 Fingerprint:
       B0 B1 73 0E CB C7 FF 45 05 14 2C 49 F1 29 5E 6E DA 6B CA ED 7E 2C
       68 C5 BE 91 B5 A1 10 01 F0 24
```

- Atau, download dan gunakan file .sig untuk menggunakan metode ini, ikuti langkah-langkah berikut.
  - Pasang aplikasi GPG ke host macOS Anda dengan memasukkan perintah berikut.

    ```
    brew install GnuPG
    ```

- Unduh file tanda tangan paket menggunakan curl. Untuk menentukan file tanda tangan yang benar, lihat Tautan Unduhan CloudWatch Agen.
- Untuk memverifikasi tanda tangan, jalankan gpg --verify.

```
PS> gpg --verify sig-filename agent-download-filename
gpg: Signature made 11/29/17 23:00:45 Coordinated Universal Time
gpg:                using RSA key D58167303B789C72
```

```
gpg: Good signature from "Amazon CloudWatch Agent" [unknown]
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: 9376 16F3 450B 7D80 6CBD  9725 D581 6730 3B78 9C72
```

Jika output mencakup frasa `BAD signature`, periksa apakah Anda melakukan prosedur dengan benar. Jika Anda terus mendapatkan respons ini, hubungi Layanan Web Amazon dan hindari menggunakan file yang diunduh.

Catat peringatan tentang kepercayaan. Kunci hanya dapat dipercaya jika Anda atau seseorang yang Anda percaya telah menandatanganinya. Ini tidak berarti bahwa tanda pembeda Anda tidak valid, hanya saja Anda belum memverifikasi kunci publiknya.

Buat file konfigurasi CloudWatch agen

Sebelum menjalankan CloudWatch agen di server apa pun, Anda harus membuat satu atau lebih file konfigurasi CloudWatch agen.

File konfigurasi agen adalah file JSON yang menentukan metrik, log, dan jejak yang akan dikumpulkan agen, termasuk metrik kustom. Anda dapat membuatnya dengan menggunakan pemandu atau dengan membuatnya sendiri dari awal. Anda juga dapat menggunakan pemandu untuk membuat file konfigurasi di awal lalu mengubahnya secara manual. Jika membuat atau memodifikasi file secara manual, prosesnya lebih rumit, tetapi Anda memiliki kontrol lebih banyak atas metrik yang dikumpulkan dan dapat menentukan metrik yang tidak tersedia melalui pemandu.

Setiap kali mengubah file konfigurasi agen, Anda kemudian harus memulai ulang agen agar perubahan berlaku. Untuk memulai ulang agen, ikuti petunjuk di Mulai CloudWatch agen.

Setelah membuat file konfigurasi, Anda dapat menyimpannya secara manual sebagai file JSON lalu menggunakan file ini saat melakukan instalasi agen di server Anda. Atau, Anda dapat menyimpannya di System Manager Parameter Store jika Anda akan menggunakan System Manager saat melakukan instalasi agen di server.

CloudWatch Agen mendukung penggunaan beberapa file konfigurasi. Untuk informasi selengkapnya, lihat Beberapa file konfigurasi CloudWatch agen.

Metrik, log, dan jejak yang dikumpulkan oleh CloudWatch agen menimbulkan biaya. Untuk informasi selengkapnya tentang harga, lihat CloudWatch Harga Amazon.

Daftar Isi

- Buat file konfigurasi CloudWatch agen dengan wizard
- Buat atau edit file konfigurasi CloudWatch agen secara manual

Buat file konfigurasi CloudWatch agen dengan wizard

Wizard file konfigurasi agen`amazon-cloudwatch-agent-config-wizard`,, mengajukan serangkaian pertanyaan untuk membantu Anda mengonfigurasi CloudWatch agen untuk kebutuhan Anda.

Kredensi yang diperlukan

Wizard dapat mendeteksi secara otomatis kredensil dan AWS Wilayah untuk digunakan jika Anda memiliki AWS kredensyal dan file konfigurasi sebelum Anda memulai wizard. Untuk informasi selengkapnya tentang file ini, silakan lihat File Konfigurasi dan Kredensial di AWS Systems Manager Panduan Pengguna .

Dalam file AWS kredensial, wizard memeriksa kredensi default dan juga mencari `AmazonCloudWatchAgent` bagian seperti berikut ini:

```
[AmazonCloudWatchAgent]
aws_access_key_id = my_access_key
aws_secret_access_key = my_secret_key
```

Pemandu menampilkan kredensial default, kredensial dari `AmazonCloudWatchAgent`, dan `Others` pilihan yang tepat. Anda dapat memilih kredensial mana yang akan digunakan. Jika memilih `Others`, Anda dapat memasukkan kredensial.

Untuk *`my_access_key`* dan *`my_secret_key`*, gunakan kunci dari pengguna IAM yang memiliki izin untuk menulis ke Systems Manager Parameter Store. Untuk informasi selengkapnya tentang pengguna IAM yang dibutuhkan untuk CloudWatch agen, lihatBuat pengguna IAM untuk digunakan dengan CloudWatch agen di server lokal.

Dalam file AWS konfigurasi, Anda dapat menentukan Wilayah tempat agen mengirimkan metrik jika berbeda dari `[default]` bagian tersebut. Bawaannya adalah menerbitkan metrik ke Wilayah tempat instans Amazon EC2 berada. Jika metrik harus dipublikasikan ke Wilayah yang berbeda, tentukan Wilayah di sini. Pada contoh berikut, metrik diterbitkan untuk `us-west-1` Wilayah.

```
[AmazonCloudWatchAgent]
```

```
region = us-west-1
```

Jalankan wizard konfigurasi CloudWatch agen

Untuk membuat file konfigurasi CloudWatch agen

1. Mulai wizard konfigurasi CloudWatch agen dengan memasukkan yang berikut:

   ```
   sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-config-wizard
   ```

   Di server yang menjalankan Windows Server, jalankan perintah berikut untuk meluncurkan pemandu:

   ```
   cd "C:\Program Files\Amazon\AmazonCloudWatchAgent"
   ```

   ```
   .\amazon-cloudwatch-agent-config-wizard.exe
   ```

2. Jawab pertanyaan untuk menyesuaikan file konfigurasi untuk server Anda.
3. Jika Anda menyimpan file konfigurasi secara lokal, file konfigurasi `config.json` disimpan dalam `/opt/aws/amazon-cloudwatch-agent/bin/` di server Linux, dan disimpan di `C:\Program Files\Amazon\AmazonCloudWatchAgent` pada Server Windows. Anda kemudian dapat menyalin file ini ke server lain tempat Anda ingin melakukan instalasi agen.

   Jika Anda akan menggunakan System Manager untuk melakukan instalasi dan mengonfigurasi agen, pastikan untuk menjawab Ya ketika diminta untuk menyimpan file di Systems Manager Parameter Store. Anda juga dapat memilih untuk menyimpan file di Parameter Store bahkan jika Anda tidak menggunakan Agen SSM untuk menginstal CloudWatch agen. Untuk dapat menyimpan file di Parameter Store, Anda harus menggunakan peran IAM dengan izin yang memadai. Untuk informasi selengkapnya, lihat Buat peran IAM dan pengguna untuk digunakan dengan agen CloudWatch .

CloudWatch set metrik agen yang telah ditentukan

Pemandu dikonfigurasi dengan rangkaian metrik yang telah ditentukan sebelumnya, dengan tingkat pedetail yang berbeda. Rangkaian metrik ini ditunjukkan dalam tabel berikut. Untuk informasi selengkapnya tentang metrik ini, silakan lihat Metrik yang dikumpulkan oleh agen CloudWatch.

> **Note**
>
> Parameter Store mendukung parameter-parameter yang ada di tingkat Standar dan Tingkat Lanjut. Tingkat-tingkat parameter ini tidak terkait dengan detail tingkat Dasar, Standar, dan Tingkat Lanjut dari metrik yang dijelaskan dalam tabel ini.

Instans Amazon EC2 yang menjalankan Linux

Tingkat detail	Termasuk metrik
Basic	Mem: mem_used_percent Disk: disk_used_percent Metrik-metrik `disk` seperti `disk_used_percent` memiliki sebuah dimensi untuk `Partition`, yang berarti bahwa jumlah metrik kustom yang dihasilkan tergantung pada jumlah partisi yang dikaitkan dengan instans Anda. Jumlah partisi disk yang Anda miliki tergantung pada AMI mana yang Anda gunakan dan jumlah volume EBS Amazon yang Anda lampirkan di server.
Standar	CPU: `cpu_usage_idle`, `cpu_usage_iowait`, `cpu_usage_user`, `cpu_usage_system` Disk: `disk_used_percent`, `disk_inodes_free` Diskio: `diskio_io_time` Mem: `mem_used_percent` Swap: `swap_used_percent`
Advanced	CPU: `cpu_usage_idle`, `cpu_usage_iowait`, `cpu_usage_user`, `cpu_usage_system` Disk: `disk_used_percent`, `disk_inodes_free` Diskio: `diskio_io_time`, `diskio_write_bytes`, `diskio_read_bytes`, `diskio_writes`, `diskio_reads`

Tingkat detail	Termasuk metrik
	Mem: `mem_used_percent` Netstat: `netstat_tcp_established` , `netstat_tcp_time_wait` Swap: `swap_used_percent`

Server on-premise yang menjalankan Linux

Tingkat detail	Termasuk metrik
Basic	Disk: `disk_used_percent` Diskio: `diskio_write_bytes` , `diskio_read_bytes` , `diskio_writes` , `diskio_reads` Mem: `mem_used_percent` Net: `net_bytes_sent` , `net_bytes_recv` , `net_packets_sent` , `net_packets_recv` Swap: `swap_used_percent`
Standar	CPU: `cpu_usage_idle` , `cpu_usage_iowait` Disk: `disk_used_percent` , `disk_inodes_free` Diskio: `diskio_io_time` , `diskio_write_bytes` , `diskio_read_bytes` , `diskio_writes` , `diskio_reads` Mem: `mem_used_percent` Net: `net_bytes_sent` , `net_bytes_recv` , `net_packets_sent` , `net_packets_recv` Swap: `swap_used_percent`
Advanced	CPU: `cpu_usage_guest` , `cpu_usage_idle` , `cpu_usage_iowait` , `cpu_usage_steal` , `cpu_usage_user` , `cpu_usage_system`

Tingkat detail	Termasuk metrik
	Disk: `disk_used_percent` , `disk_inodes_free` Diskio: `diskio_io_time` , `diskio_write_bytes` , `diskio_read_bytes` , `diskio_writes` , `diskio_reads` Mem: `mem_used_percent` Net: `net_bytes_sent` , `net_bytes_recv` , `net_packets_sent` , `net_packets_recv` Netstat: `netstat_tcp_established` , `netstat_tcp_time_wait` Swap: `swap_used_percent`

Instans Amazon EC2 yang menjalankan Server Windows

> **Note**
>
> Nama metrik yang tercantum dalam tabel ini menampilkan bagaimana metrik muncul saat dilihat di konsol. Nama metrik sebenarnya mungkin tidak termasuk kata pertama. Sebagai contoh, nama metrik `LogicalDisk % Free Space` sebenarnya untuk hanya`% Free Space`.

Tingkat detail	Termasuk metrik
Basic	Memori: `Memory % Committed Bytes In Use` LogicalDisk: `LogicalDisk % Free Space`
Standar	Memory: `Memory % Committed Bytes In Use` Paging: `Paging File % Usage` Processor: `Processor % Idle Time`, `Processor % Interrupt Time`, `Processor % User Time` PhysicalDisk: `PhysicalDisk % Disk Time`

Tingkat detail	Termasuk metrik
	LogicalDisk: `LogicalDisk % Free Space`
Advanced	Memory: `Memory % Committed Bytes In Use` Paging: `Paging File % Usage` Processor: `Processor % Idle Time`, `Processor % Interrupt Time`, `Processor % User Time` LogicalDisk: `LogicalDisk % Free Space` PhysicalDisk: `PhysicalDisk % Disk Time` , `PhysicalDisk Disk Write Bytes/sec` , `PhysicalDisk Disk Read Bytes/sec` , `PhysicalDisk Disk Writes/sec` , `PhysicalDisk Disk Reads/sec` TCP: `TCPv4 Connections Established` , `TCPv6 Connections Established`

Server on-premise yang menjalankan Server Windows

> **Note**
>
> Nama metrik yang tercantum dalam tabel ini menampilkan bagaimana metrik muncul saat dilihat di konsol. Nama metrik sebenarnya mungkin tidak termasuk kata pertama. Sebagai contoh, nama metrik `LogicalDisk % Free Space` sebenarnya untuk hanya`% Free Space`.

Tingkat detail	Termasuk metrik
Basic	Paging: `Paging File % Usage` Processor: `Processor % Processor Time` LogicalDisk:`LogicalDisk % Free Space`

Tingkat detail	Termasuk metrik
	PhysicalDisk: `PhysicalDisk Disk Write Bytes/sec` , `PhysicalDisk Disk Read Bytes/sec` , `PhysicalDisk Disk Writes/sec` , `PhysicalDisk Disk Reads/sec` Memory: `Memory % Committed Bytes In Use` Network Interface: `Network Interface Bytes Sent/sec`, `Network Interface Bytes Received/sec` , `Network Interface Packets Sent/sec`, `Network Interface Packets Received/sec`
Standar	Paging: `Paging File % Usage` Processor: `Processor % Processor Time`, `Processor % Idle Time`, `Processor % Interrupt Time` LogicalDisk: `LogicalDisk % Free Space` PhysicalDisk: `PhysicalDisk % Disk Time` , `PhysicalDisk Disk Write Bytes/sec` , `PhysicalDisk Disk Read Bytes/sec` , `PhysicalDisk Disk Writes/sec` , `PhysicalDisk Disk Reads/sec` Memory: `Memory % Committed Bytes In Use` Network Interface: `Network Interface Bytes Sent/sec`, `Network Interface Bytes Received/sec` , `Network Interface Packets Sent/sec`, `Network Interface Packets Received/sec`

Tingkat detail	Termasuk metrik
Advanced	Paging:`Paging File % Usage` Processor: `Processor % Processor Time`, `Processor % Idle Time`, `Processor % Interrupt Time`, `Processor % User Time` LogicalDisk: `LogicalDisk % Free Space` PhysicalDisk: `PhysicalDisk % Disk Time` , `PhysicalDisk Disk Write Bytes/sec` , `PhysicalDisk Disk Read Bytes/sec` , `PhysicalDisk Disk Writes/sec` , `PhysicalDisk Disk Reads/sec` Memory: `Memory % Committed Bytes In Use` Network Interface: `Network Interface Bytes Sent/sec`, `Network Interface Bytes Received/sec` , `Network Interface Packets Sent/sec`, `Network Interface Packets Received/sec` TCP: `TCPv4 Connections Established` , `TCPv6 Connections Established`

Buat atau edit file konfigurasi CloudWatch agen secara manual

File konfigurasi CloudWatch agen adalah file JSON dengan empat bagian,,`agent`,`metrics`, dan `logstraces`, dijelaskan sebagai berikut:

- `agent` bagian ini mencakup kolom untuk konfigurasi agen secara keseluruhan.
- `metrics`Bagian ini menentukan metrik kustom untuk pengumpulan dan penerbitan. CloudWatch Jika menggunakan agen hanya untuk mengumpulkan log, Anda dapat menghilangkan bagian `metrics` dari file.
- `logs`Bagian ini menentukan file log apa yang dipublikasikan ke CloudWatch Log. Ini dapat mencakup kejadian dari Windows Event Log (Log Kejadian) jika server menjalankan Server Windows.
- `traces`Bagian ini menentukan sumber untuk jejak yang dikumpulkan dan dikirim ke AWS X-Ray.

Bagian berikut menjelaskan struktur dan bidang file JSON ini. Anda juga dapat melihat definisi skema untuk file konfigurasi ini. Definisi skema berada di *installation-directory*/doc/amazon-cloudwatch-agent-schema.json di server Linux, dan di *installation-directory*/amazon-cloudwatch-agent-schema.json pada server yang menjalankan Server Windows.

Jika membuat atau mengedit file konfigurasi agen secara manual, Anda dapat memberinya nama apa pun. Untuk kemudahan dalam pemecahan masalah, kami sarankan Anda menamainya /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json di server Linux dan $Env:ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.json pada server yang menjalankan Server Windows. Setelah membuat file, Anda dapat menyalinnya ke server lain tempat Anda ingin melakukan instalasi agen.

> **Note**
>
> Metrik, log, dan jejak yang dikumpulkan oleh CloudWatch agen menimbulkan biaya. Untuk informasi selengkapnya tentang harga, lihat CloudWatch Harga Amazon.

CloudWatch file konfigurasi agen: Bagian Agen

Bagian agent dapat mencakup bidang-bidang berikut. Pemandu tidak membuat agent bagian. Sebaliknya, pemandu menghilangkannya dan menggunakan nilai default untuk semua bidang di bagian ini.

- metrics_collection_interval – Opsional. Menentukan seberapa sering semua metrik yang ditentukan dalam file konfigurasi ini akan dikumpulkan. Anda dapat mengganti nilai ini untuk jenis metrik tertentu.

  Nilai ini ditentukan dalam detik. Misalnya, menentukan 10 metrik yang akan dikumpulkan setiap 10 detik, dan menetapkannya menjadi 300 metrik kustom untuk dikumpulkan setiap 5 menit.

  Jika Anda mengatur nilai ini di bawah 60 detik, setiap metrik dikumpulkan sebagai metrik resolusi tinggi. Untuk informasi selengkapnya tentang metrik resolusi tinggi, silakan lihat Metrik resolusi tinggi.

  Nilai bawaannya adalah 60.
- region— Menentukan Wilayah yang akan digunakan untuk CloudWatch titik akhir saat instans Amazon EC2 sedang dipantau. Metrik yang dikumpulkan dikirim ke Wilayah ini, seperti us-

`west-1`. Jika Anda menghilangkan bidang ini, agen akan mengirimkan metrik ke Wilayah tempat instans Amazon EC2 berada.

Jika Anda memantau server on-premise, bidang ini tidak digunakan, dan agen membaca Wilayah dari profil `AmazonCloudWatchAgent` file konfigurasi AWS .

- `credentials`— Menentukan peran IAM untuk digunakan saat mengirim metrik, log, dan jejak ke akun yang berbeda. AWS Jika ditentukan, bidang ini berisi satu parameter, `role_arn`.
  - `role_arn` – Menentukan Amazon Resource Name (ARN) dari peran IAM yang akan digunakan untuk autentikasi saat mengirim metrik, log, dan jejak ke akun yang berbeda. AWS Untuk informasi selengkapnya, lihat Mengirim metrik, log, dan jejak ke akun lain.
- `debug` – Opsional. Menentukan menjalankan CloudWatch agen dengan pesan log debug. Nilai bawaannya adalah `false`.
- `aws_sdk_log_level` – Opsional. Hanya didukung di versi 1.247350.0 dan agen yang lebih baru. CloudWatch

  Anda dapat menentukan bidang ini agar agen melakukan pencatatan untuk titik akhir AWS SDK. Nilai untuk bidang ini dapat mencakup satu atau beberapa opsi berikut. Pisahkan beberapa opsi dengan karakter `|`.
  - `LogDebug`
  - `LogDebugWithSigning`
  - `LogDebugWithHTTPBody`
  - `LogDebugRequestRetries`
  - `LogDebugWithEventStreamBody`

  Untuk informasi selengkapnya tentang opsi ini, lihat LogLevelType.
- `logfile`— Menentukan lokasi di mana CloudWatch agen menulis pesan log. Jika Anda menentukan string kosong, log akan beralih ke stderr. Jika Anda tidak menentukan opsi ini, lokasi defaultnya adalah sebagai berikut:
  - Linux: `/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log`
  - Server Windows: `c:\\ProgramData\\Amazon\\CloudWatchAgent\\Logs\\amazon-cloudwatch-agent.log`

  CloudWatch Agen secara otomatis memutar file log yang dibuatnya. File log dirotasi ketika ukurannya mencapai 100 MB. Agen menyimpan file log yang dirotasi selama hingga tujuh hari, dan menyimpan sebanyak lima file log cadangan yang telah dirotasi. File log cadangan memiliki

stempel waktu yang ditambahkan ke nama file mereka. Jadwal menunjukkan tanggal dan waktu file dirotasi: misalnya, `amazon-cloudwatch-agent-2018-06-08T21-01-50.247.log.gz`.

- `omit_hostname` – Opsional. Secara bawaan, nama host dipublikasikan sebagai dimensi metrik yang dikumpulkan oleh agen, kecuali jika Anda menggunakan bidang `append_dimensions` di bagian `metrics`. Atur `omit_hostname` ke `true` untuk mencegah nama host diterbitkan sebagai dimensi bahkan jika Anda tidak menggunakan `append_dimensions`. Nilai bawaannya adalah `false`.
- `run_as_user` – Opsional. Menentukan pengguna yang akan digunakan untuk menjalankan CloudWatch agen. Jika Anda tidak menetapkan parameter ini, pengguna root akan digunakan. Opsi ini hanya valid di server Linux.

  Jika Anda menentukan opsi ini, pengguna harus ada sebelum Anda memulai CloudWatch agen. Untuk informasi selengkapnya, lihat Menjalankan CloudWatch agen sebagai pengguna yang berbeda.
- `user_agent` – Opsional. Menentukan `user-agent` string yang digunakan oleh CloudWatch agen ketika membuat panggilan API ke CloudWatch backend. Nilai bawaannya adalah string yang terdiri dari versi agen, versi bahasa pemrograman Go yang digunakan untuk mengompilasi agen, sistem operasi runtime dan arsitektur, waktu pembuatan, dan plugin yang diaktifkan.
- `usage_data` – Opsional. Secara default, CloudWatch agen mengirimkan data kesehatan dan kinerja tentang dirinya sendiri CloudWatch kapan pun ia menerbitkan metrik atau log ke. CloudWatch Data ini tidak membebankan biaya untuk Anda. Anda dapat mencegah agen mengirim data ini dengan menentukan `false` untuk`usage_data`. Jika Anda menghilangkan parameter ini, default `true` digunakan dan agen mengirimkan data kesehatan dan performa.

  Jika menetapkan nilai ini`false`, Anda harus berhenti dan memulai ulang agen agar berlaku.

Berikut ini adalah contoh dari sebuah bagian `agent`.

```
"agent": {
   "metrics_collection_interval": 60,
   "region": "us-west-1",
   "logfile": "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log",
   "debug": false,
   "run_as_user": "cwagent"
  }
```

CloudWatch file konfigurasi agen: Bagian metrik

Bidang umum untuk Linux dan Windows

Pada server yang menjalankan Linux atau Windows Server, `metrics` bagian ini mencakup kolom berikut:

- `namespace` – Opsional. Ruangnama yang digunakan untuk metrik yang dikumpulkan oleh agen. Nilai bawaannya adalah `CWAgent`. Panjang maksimalnya adalah 255 karakter. Berikut ini salah satu contohnya:

  ```
  {
    "metrics": {
      "namespace": "Development/Product1Metrics",
     ......
     },
  }
  ```

- `append_dimensions` – Opsional. Menambahkan dimensi metrik Amazon EC2 ke semua metrik yang dikumpulkan oleh agen. Hal ini juga menyebabkan agen untuk tidak menerbitkan nama host sebagai dimensi.

  Satu-satunya pasangan nilai kunci yang didukung untuk `append_dimensions` ditampilkan dalam daftar berikut. Pasangan kunci lainnya diabaikan. Agen mendukung pasangan nilai kunci ini persis seperti yang ditampilkan dalam daftar berikut. Anda tidak dapat mengubah nilai kunci untuk menerbitkan nama dimensi yang berbeda untuknya.

  - `"ImageId":"${aws:ImageId}"` menetapkan AMI ID instans sebagai nilai dari `ImageId` dimensi.
  - `"InstanceId":"${aws:InstanceId}"` menetapkan ID instans dari instans sebagai nilai `InstanceId` dimensi.
  - `"InstanceType":"${aws:InstanceType}"` menetapkan tipe instans dari instans sebagai nilai dimensi `InstanceType`.
  - `"AutoScalingGroupName":"${aws:AutoScalingGroupName}"` menetapkan nama contoh grup Auto Scaling `AutoScalingGroupName` dimensi tertentu.

  Jika Anda ingin menambahkan dimensi ke metrik dengan pasangan nilai kunci arbitrer, gunakan `append_dimensions` di bidang untuk tipe metrik tertentu.

Jika menentukan nilai yang bergantung pada metadata EC2 Amazon dan menggunakan proksi, Anda harus memastikan bahwa server dapat mengakses titik akhir untuk Amazon EC2. Untuk informasi selengkapnya tentang titik akhir ini, silakan lihat Amazon Elastic Compute Cloud (Amazon EC2) di. Referensi Umum Amazon Web Services

- `aggregation_dimensions` – Opsional. Menentukan dimensi yang akan digabungkan dengan metrik yang dikumpulkan. Sebagai contoh, jika Anda menggulir metrik pada `AutoScalingGroupName` dimensi, metrik dari semua instans dalam setiap kelompok grup Auto Scaling dikumpulkan dan dapat dilihat secara keseluruhan.

  Anda dapat menggulirkan metrik di sepanjang dimensi tunggal atau ganda. Misalnya, menentukan `[["InstanceId"], ["InstanceType"], ["InstanceId","InstanceType"]]` mengagregasi metrik untuk ID instans secara tunggal, tipe instans secara tunggal, dan untuk kombinasi kedua dimensi tersebut.

  Anda juga dapat menentukan `[]` untuk menggulirkan semua metrik menjadi satu koleksi, mengabaikan semua dimensi.

- `endpoint_override` – Menentukan titik akhir FIPS atau tautan privat untuk digunakan sebagai titik akhir tempat agen mengirimkan metrik. Menentukan hal ini dan mengatur tautan privat memungkinkan Anda mengirimkan metrik ke titik akhir VPC Amazon VPC. Untuk informasi selengkapnya, silakan lihat Apa Itu Amazon VPC?.

  Nilai dari `endpoint_override` harus string yang berupa URL.

  Misalnya, bagian berikut dari bagian metrik pada file konfigurasi menetapkan agen untuk menggunakan Titik Akhir VPC saat mengirim metrik.

```
{
  "metrics": {
    "endpoint_override": "vpce-XXXXXXXXXXXXXXXXXXXXXXXX.monitoring.us-
east-1.vpce.amazonaws.com",
   ......
   },
}
```

- `metrics_collected` – Wajib. Menentukan metrik mana yang akan dikumpulkan, termasuk metrik kustom yang dikumpulkan melalui atau `StatsD` atau `collectd`. Bagian ini mencakup beberapa subbagian.

Isi dari `metrics_collected` bergantung pada apakah file konfigurasi ini untuk server yang menjalankan Linux atau Windows Server.

- `force_flush_interval` – Menentukan dalam hitungan detik lamanya waktu maksimal metrik itu tetap berada dalam buffer memori sebelum dikirim ke server. Apa pun pengaturannya, jika ukuran metrik di buffer mencapai 1 MB atau 1000 metrik berbeda, metrik akan segera dikirim ke server.

  Nilai bawaannya adalah 60.
- `credentials` – Menentukan peran IAM yang akan digunakan saat mengirim metrik ke akun yang berbeda. Jika ditentukan, bidang ini berisi satu parameter, `role_arn`.
  - `role_arn` – Menentukan ARN peran IAM yang akan digunakan untuk autentikasi saat mengirim metrik ke akun yang berbeda. Untuk informasi selengkapnya, lihat Mengirim metrik, log, dan jejak ke akun lain. Jika ditentukan di sini, nilai ini akan menggantikan `role_arn` yang ditentukan dalam `agent` dari file konfigurasi, jika ada.

Bagian Linux

Di server yang menjalankan Linux, `metrics_collected` bagian file konfigurasi juga dapat berisi bidang-bidang berikut.

Banyak dari bidang ini dapat mencakup `measurement` bagian yang mencantumkan metrik yang ingin Anda kumpulkan untuk sumber daya tersebut. Ini `measurement` Anda dapat menentukan nama metrik lengkap seperti `swap_used`, atau hanya bagian dari nama metrik yang akan ditambahkan ke jenis sumber daya. Misalnya, menentukan `reads` dalam `measurement` bagian dari `diskio` menyebabkan `diskio_reads` metrik yang akan dikumpulkan.

- `collectd` – Opsional. Menentukan bahwa Anda ingin mengambil metrik kustom menggunakan `collectd` protokol. Anda menggunakan `collectd` perangkat lunak untuk mengirim metrik ke CloudWatch agen. Untuk informasi selengkapnya tentang opsi konfigurasi yang tersedia untuk yang dikumpulkan, silakan lihat Ambil Metrik Kustom dengan koleksi.
- `cpu` – Opsional. Menentukan bahwa metrik CPU harus dikumpulkan. Bagian ini hanya valid untuk instans Linux. Anda harus memasukkan setidaknya satu `resources` dan `totalcpu` untuk setiap metrik CPU yang akan dikumpulkan. Bagian ini dapat mencakup bidang-bidang berikut:
  - `drop_original_metrics` – Opsional. Jika Anda menggunakan bidang `aggregation_dimensions` di bagian `metrics` untuk menggulung metrik ke dalam hasil agregat, maka secara default agen mengirimkan metrik-metrik agregat dan metrik asli yang dipisahkan untuk setiap nilai dimensi. Jika Anda tidak ingin metrik asli dikirim CloudWatch,

Anda dapat menentukan parameter ini dengan daftar metrik. Metrik yang ditentukan bersama dengan parameter ini tidak memiliki metrik berdasarkan dimensi yang dilaporkan. CloudWatch Sebaliknya, hanya metrik-metrik agregat saja yang dilaporkan. Hal ini akan mengurangi jumlah metrik yang dikumpulkan oleh agen, dan akan mengurangi biaya Anda.

- `resources` – Opsional. Tentukan bidang ini dengan nilai `*` untuk mengumpulkan metrik per-cpu. Satu-satunya nilai yang diperbolehkan adalah `*`.
- `totalcpu` – Opsional. Menentukan apakah pelaporan metrik cpu dikumpulkan di seluruh inti permasalahan. Bawaannya adalah benar.
- `measurement` – Tentukan rangkaian metrik cpu yang akan dikumpulkan. Kemungkinan nilai adalah `time_active`, `time_guest`, `time_guest_nice`, `time_idle`, `time_iowait`, `time_irq`, `time_nice`, `time_softirq`, `time_steal`, `time_system`, `time_user`, `usage_active`, `usage_guest`, `usage_guest_nice`, `usage_idle`, `usage_iowait`, `usage_irq`, `usage_nice`, `usage_softirq`, `usage_steal`, `usage_system`, dan `usage_user`. Kolom ini wajib diisi jika Anda menyertakan `cpu`.

  Secara bawaan, unit untuk `cpu_usage_*` adalah `Percent`, dan `cpu_time_*` metrik tidak memiliki unit.

  Di dalam entri untuk setiap metrik individu, secara opsional Anda dapat menentukan salah satu atau kedua hal berikut:

  - `rename` – Menentukan nama yang berbeda untuk metrik ini.
  - `unit` – Menentukan unit yang akan digunakan untuk metrik ini, membatalkan unit default `None` untuk metrik. Unit yang Anda tentukan harus merupakan satuan CloudWatch metrik yang valid, seperti yang tercantum dalam `Unit` deskripsi di MetricDatum.
- `metrics_collection_interval` – Opsional. Menentukan seberapa sering mengumpulkan metrik cpu, menggulingkan global `metrics_collection_interval` yang ditentukan dalam bagian `agent` dari file konfigurasi.

  Nilai ini ditentukan dalam detik. Misalnya, menentukan 10 metrik yang akan dikumpulkan setiap 10 detik, dan menetapkannya menjadi 300 metrik kustom untuk dikumpulkan setiap 5 menit.

  Jika Anda mengatur nilai ini di bawah 60 detik, setiap metrik dikumpulkan sebagai metrik resolusi tinggi. Untuk informasi selengkapnya tentang metrik resolusi tinggi, silakan lihat Metrik resolusi tinggi.
- `append_dimensions` – Opsional. Dimensi tambahan untuk digunakan hanya pada metrik cpu. Jika Anda menentukan bidang ini, maka bidang tersebut digunakan sebagai tambahan dimensi

yang ditentukan dalam `append_dimensions` global yang digunakan untuk semua jenis metrik yang dikumpulkan oleh agen.

- `disk` – Opsional. Menentukan bahwa metrik disk harus dikumpulkan. Bagian ini hanya valid untuk instans Linux. Bagian ini dapat mencakup bidang-bidang berikut:
  - `drop_original_metrics` – Opsional. Jika Anda menggunakan bidang `aggregation_dimensions` di bagian `metrics` untuk menggulung metrik ke dalam hasil agregat, maka secara default agen mengirimkan metrik-metrik agregat dan metrik asli yang dipisahkan untuk setiap nilai dimensi. Jika Anda tidak ingin metrik asli dikirim CloudWatch, Anda dapat menentukan parameter ini dengan daftar metrik. Metrik yang ditentukan bersama dengan parameter ini tidak memiliki metrik berdasarkan dimensi yang dilaporkan. CloudWatch Sebaliknya, hanya metrik-metrik agregat saja yang dilaporkan. Hal ini akan mengurangi jumlah metrik yang dikumpulkan oleh agen, dan akan mengurangi biaya Anda.
  - `resources` – Opsional. Menentukan rangkaian titik pemasangan disk. Bidang ini membatasi CloudWatch untuk mengumpulkan metrik hanya dari titik pemasangan yang terdaftar. Anda dapat menentukan * sebagai nilai untuk mengumpulkan metrik dari semua titik pemasangan. Nilai bawaannya adalah mengumpulkan metrik dari semua titik dudukan.
  - `measurement` – Menentukan larik metrik disk yang akan dikumpulkan. Kemungkinan nilai adalah `free`, `total`, `used`, `used_percent`, `inodes_free`, `inodes_used`, dan `inodes_total`. Kolom ini wajib diisi jika Anda menyertakan `disk`.

    > **Note**
    >
    > Metrik-metrik `disk` tersebut memiliki sebuah dimensi untuk `Partition`, yang berarti bahwa jumlah metrik kustom yang dihasilkan tergantung pada jumlah partisi yang dikaitkan dengan instans Anda. Jumlah partisi disk yang Anda miliki tergantung pada AMI mana yang Anda gunakan dan jumlah volume EBS Amazon yang Anda lampirkan di server.

    Untuk melihat satuan bawaan untuk setiap `disk` metrik, silakan lihat Metrik yang dikumpulkan oleh CloudWatch agen di instans Linux dan macOS.

    Di dalam entri untuk setiap metrik individu, secara opsional Anda dapat menentukan salah satu atau kedua hal berikut:

    - `rename` – Menentukan nama yang berbeda untuk metrik ini.

    - `unit` – Menentukan unit yang akan digunakan untuk metrik ini, membatalkan unit default `None` dari `None` untuk metrik tersebut. Unit yang Anda tentukan harus merupakan satuan CloudWatch metrik yang valid, seperti yang tercantum dalam `Unit` deskripsi di MetricDatum.
- `ignore_file_system_types` – Menentukan jenis sistem file untuk dikecualikan saat mengumpulkan metrik disk. Nilai yang valid termasuk `sysfs`, `devtmpfs`, dan sebagainya.
- `drop_device` – Mengatur ini untuk `true` penyebab `Device` tidak disertakan sebagai dimensi untuk metrik disk.

  Dengan mencegah `Device` agar tidak digunakan sebagai sebuah dimensi dapat berguna pada instans yang menggunakan sistem Nitro karena pada instans tersebut, nama perangkat berubah untuk setiap pemasangan disk ketika instans dinyalakan ulang. Hal ini dapat menyebabkan data tidak konsisten dalam metrik Anda dan menyebabkan alarm berdasarkan metrik tersebut masuk `INSUFFICIENT DATA` kondisi.

  Bawaannya adalah `false`.
- `metrics_collection_interval` – Opsional. Menentukan seberapa sering mengumpulkan metrik disk, menggulingkan `metrics_collection_interval` yang ditentukan dalam `agent` di file konfigurasi.

  Nilai ini ditentukan dalam detik.

  Jika Anda mengatur nilai ini di bawah 60 detik, setiap metrik dikumpulkan sebagai metrik resolusi tinggi. Untuk informasi selengkapnya, lihat Metrik resolusi tinggi.
- `append_dimensions` – Opsional. Dimensi tambahan untuk digunakan hanya pada metrik disk. Jika Anda menentukan bidang ini, maka bidang ini akan digunakan sebagai tambahan dimensi yang ditentukan dalam bidang `append_dimensions` yang digunakan untuk semua jenis metrik yang dikumpulkan oleh agen.

- `diskio` – Opsional. Menentukan bahwa metrik disk i/o harus dikumpulkan. Bagian ini hanya valid untuk instans Linux. Bagian ini dapat mencakup bidang-bidang berikut:
  - `drop_original_metrics` – Opsional. Jika Anda menggunakan bidang `aggregation_dimensions` di bagian `metrics` untuk menggulung metrik ke dalam hasil agregat, maka secara default agen mengirimkan metrik-metrik agregat dan metrik asli yang dipisahkan untuk setiap nilai dimensi. Jika Anda tidak ingin metrik asli dikirim CloudWatch, Anda dapat menentukan parameter ini dengan daftar metrik. Metrik yang ditentukan bersama dengan parameter ini tidak memiliki metrik berdasarkan dimensi yang dilaporkan. CloudWatch

Sebaliknya, hanya metrik-metrik agregat saja yang dilaporkan. Hal ini akan mengurangi jumlah metrik yang dikumpulkan oleh agen, dan akan mengurangi biaya Anda.

- `resources` – Opsional. Jika Anda menentukan larik perangkat, CloudWatch kumpulkan metrik hanya dari perangkat tersebut. Jika tidak, metrik untuk semua perangkat akan dikumpulkan. Anda juga dapat menentukan * sebagai nilai untuk mengumpulkan metrik dari semua perangkat.
- `measurement` – Menentukan larik metrik diskio yang akan dikumpulkan. Kemungkinan nilai adalah `reads`, `writes`, `read_bytes`, `write_bytes`, `read_time`, `write_time`, `io_time`, dan `iops_in_progress`. Kolom ini wajib diisi jika Anda menyertakan `diskio`.

  Di dalam entri untuk setiap metrik individu, secara opsional Anda dapat menentukan salah satu atau kedua hal berikut:

  - `rename` – Menentukan nama yang berbeda untuk metrik ini.
  - `unit` – Menentukan unit yang akan digunakan untuk metrik ini, membatalkan unit default `None` dari `None` untuk metrik tersebut. Unit yang Anda tentukan harus merupakan satuan CloudWatch metrik yang valid, seperti yang tercantum dalam `Unit` deskripsi di MetricDatum.
- `metrics_collection_interval` – Opsional. Menentukan seberapa sering mengumpulkan metrik diskio, menggulingkan `metrics_collection_interval` yang ditentukan dalam `agent` di file konfigurasi.

  Nilai ini ditentukan dalam detik.

  Jika Anda mengatur nilai ini di bawah 60 detik, setiap metrik dikumpulkan sebagai metrik resolusi tinggi. Untuk informasi selengkapnya tentang metrik resolusi tinggi, silakan lihat Metrik resolusi tinggi.
- `append_dimensions` – Opsional. Dimensi tambahan untuk digunakan hanya untuk diskio metrics. Jika Anda menentukan bidang ini, maka bidang ini akan digunakan sebagai tambahan dimensi yang ditentukan dalam bidang `append_dimensions` yang digunakan untuk semua jenis metrik yang dikumpulkan oleh agen.

- `swap` – Opsional. Menentukan bahwa metrik memori swap harus dikumpulkan. Bagian ini hanya valid untuk instans Linux. Bagian ini dapat mencakup bidang-bidang berikut:
  - `drop_original_metrics` – Opsional. Jika Anda menggunakan bidang `aggregation_dimensions` di bagian `metrics` untuk menggulung metrik ke dalam hasil agregat, maka secara default agen mengirimkan metrik-metrik agregat dan metrik asli yang dipisahkan untuk setiap nilai dimensi. Jika Anda tidak ingin metrik asli dikirim CloudWatch, Anda dapat menentukan parameter ini dengan daftar metrik. Metrik yang ditentukan bersama dengan parameter ini tidak memiliki metrik berdasarkan dimensi yang dilaporkan. CloudWatch

Sebaliknya, hanya metrik-metrik agregat saja yang dilaporkan. Hal ini akan mengurangi jumlah metrik yang dikumpulkan oleh agen, dan akan mengurangi biaya Anda.

- `measurement` – Menentukan rangkaian metrik swap yang akan dikumpulkan. Nilai yang mungkin adalah `free`, `used`, dan `used_percent`. Kolom ini wajib diisi jika Anda menyertakan `swap`.

  Untuk melihat satuan bawaan untuk setiap `swap` metrik, silakan lihat Metrik yang dikumpulkan oleh CloudWatch agen di instans Linux dan macOS.

  Di dalam entri untuk setiap metrik individu, secara opsional Anda dapat menentukan salah satu atau kedua hal berikut:

  - `rename` – Menentukan nama yang berbeda untuk metrik ini.
  - `unit` – Menentukan unit yang akan digunakan untuk metrik ini, membatalkan unit default `None` dari `None` untuk metrik tersebut. Unit yang Anda tentukan harus merupakan satuan CloudWatch metrik yang valid, seperti yang tercantum dalam `Unit` deskripsi di MetricDatum.
- `metrics_collection_interval` – Opsional. Menentukan seberapa sering mengumpulkan metrik swap, menggulingkan `metrics_collection_interval` yang ditentukan dalam bagian `agent` di file konfigurasi.

  Nilai ini ditentukan dalam detik.

  Jika Anda mengatur nilai ini di bawah 60 detik, setiap metrik dikumpulkan sebagai metrik resolusi tinggi. Untuk informasi selengkapnya tentang metrik resolusi tinggi, silakan lihat Metrik resolusi tinggi.
- `append_dimensions` – Opsional. Dimensi tambahan untuk digunakan hanya pada metrik swap. Jika Anda menentukan bidang ini, maka bidang tersebut akan digunakan sebagai tambahan dimensi yang ditentukan dalam bidang `append_dimensions` global yang digunakan untuk semua jenis metrik yang dikumpulkan oleh agen. Ini dikumpulkan sebagai metrik resolusi tinggi.

- `mem` – Opsional. Menentukan bahwa metrik memori harus dikumpulkan. Bagian ini hanya valid untuk instans Linux. Bagian ini dapat mencakup bidang-bidang berikut:
  - `drop_original_metrics` – Opsional. Jika Anda menggunakan bidang `aggregation_dimensions` di bagian `metrics` untuk menggulung metrik ke dalam hasil agregat, maka secara default agen mengirimkan metrik-metrik agregat dan metrik asli yang dipisahkan untuk setiap nilai dimensi. Jika Anda tidak ingin metrik asli dikirim CloudWatch, Anda dapat menentukan parameter ini dengan daftar metrik. Metrik yang ditentukan bersama

dengan parameter ini tidak memiliki metrik berdasarkan dimensi yang dilaporkan. CloudWatch Sebaliknya, hanya metrik-metrik agregat saja yang dilaporkan. Hal ini akan mengurangi jumlah metrik yang dikumpulkan oleh agen, dan akan mengurangi biaya Anda.

- `measurement` – Menentukan susunan metrik memori yang akan dikumpulkan. Kemungkinan nilai adalah `active`, `available`, `available_percent`, `buffered`, `cached`, `free`, `inactive`, `total`, `used`, dan `used_percent`. Kolom ini wajib diisi jika Anda menyertakan `mem`.

  Untuk melihat satuan bawaan untuk setiap `mem` metrik, silakan lihat Metrik yang dikumpulkan oleh CloudWatch agen di instans Linux dan macOS.

  Di dalam entri untuk setiap metrik individu, secara opsional Anda dapat menentukan salah satu atau kedua hal berikut:

  - `rename` – Menentukan nama yang berbeda untuk metrik ini.
  - `unit` – Menentukan unit yang akan digunakan untuk metrik ini, membatalkan unit default `None` untuk metrik. Unit yang Anda tentukan harus merupakan satuan CloudWatch metrik yang valid, seperti yang tercantum dalam `Unit` deskripsi di MetricDatum.

- `metrics_collection_interval` – Opsional. Menentukan seberapa sering mengumpulkan metrik mem, menggulingkan `metrics_collection_interval` yang ditentukan dalam `agent` di file konfigurasi.

  Nilai ini ditentukan dalam detik.

  Jika Anda mengatur nilai ini di bawah 60 detik, setiap metrik dikumpulkan sebagai metrik resolusi tinggi. Untuk informasi selengkapnya tentang metrik resolusi tinggi, silakan lihat Metrik resolusi tinggi.

- `append_dimensions` – Opsional. Dimensi tambahan untuk digunakan hanya untuk metrik mem. Jika Anda menetapkan bidang ini, maka bidang tersebut akan digunakan sebagai tambahan dimensi yang ditentukan dalam `append_dimensions` yang digunakan untuk semua jenis metrik yang dikumpulkan oleh agen.

- `net` – Opsional. Menentukan bahwa metrik jaringan akan dikumpulkan. Bagian ini hanya valid untuk instans Linux. Bagian ini dapat mencakup bidang-bidang berikut:

  - `drop_original_metrics` – Opsional. Jika Anda menggunakan bidang `aggregation_dimensions` di bagian `metrics` untuk menggulung metrik ke dalam hasil agregat, maka secara default agen mengirimkan metrik-metrik agregat dan metrik asli yang dipisahkan untuk setiap nilai dimensi. Jika Anda tidak ingin metrik asli dikirim CloudWatch,

Anda dapat menentukan parameter ini dengan daftar metrik. Metrik yang ditentukan bersama dengan parameter ini tidak memiliki metrik berdasarkan dimensi yang dilaporkan. CloudWatch Sebaliknya, hanya metrik-metrik agregat saja yang dilaporkan. Hal ini akan mengurangi jumlah metrik yang dikumpulkan oleh agen, dan akan mengurangi biaya Anda.

- `resources` – Opsional. Jika Anda menentukan array antarmuka jaringan, CloudWatch kumpulkan metrik hanya dari antarmuka tersebut. Jika tidak, metrik untuk semua perangkat akan dikumpulkan. Anda juga dapat menentukan `*` sebagai nilai untuk mengumpulkan metrik dari semua antarmuka.
- `measurement` – Menentukan susunan metrik jaringan yang akan dikumpulkan. Kemungkinan nilai adalah `bytes_sent`, `bytes_recv`, `drop_in`, `drop_out`, `err_in`, `err_out`, `packets_sent`, dan `packets_recv`. Kolom ini wajib diisi jika Anda menyertakan `net`.

  Untuk melihat satuan bawaan untuk setiap `net` metrik, silakan lihat Metrik yang dikumpulkan oleh CloudWatch agen di instans Linux dan macOS.

  Di dalam entri untuk setiap metrik individu, secara opsional Anda dapat menentukan salah satu atau kedua hal berikut:

  - `rename` – Menentukan nama yang berbeda untuk metrik ini.
  - `unit` – Menentukan unit yang akan digunakan untuk metrik ini, membatalkan unit default `None` untuk metrik. Unit yang Anda tentukan harus merupakan satuan CloudWatch metrik yang valid, seperti yang tercantum dalam `Unit` deskripsi di MetricDatum.
- `metrics_collection_interval` – Opsional. Menentukan seberapa sering mengumpulkan metrik bersih, menggulingkan `metrics_collection_interval` yang ditentukan dalam `agent` di file konfigurasi.

  Nilai ini ditentukan dalam detik. Misalnya, menentukan 10 metrik yang akan dikumpulkan setiap 10 detik, dan menetapkannya menjadi 300 metrik kustom untuk dikumpulkan setiap 5 menit.

  Jika Anda mengatur nilai ini di bawah 60 detik, setiap metrik dikumpulkan sebagai metrik resolusi tinggi. Untuk informasi selengkapnya tentang metrik resolusi tinggi, silakan lihat Metrik resolusi tinggi.
- `append_dimensions` – Opsional. Dimensi tambahan untuk digunakan hanya pada metrik bersih. Jika Anda menentukan bidang ini, maka bidang tersebut akan digunakan sebagai tambahan dimensi yang ditentukan dalam bidang `append_dimensions` yang digunakan untuk semua jenis metrik yang dikumpulkan oleh agen.

- `netstat` – Opsional. Menentukan bahwa metrik status koneksi TCP dan koneksi UDP akan dikumpulkan. Bagian ini hanya valid untuk instans Linux. Bagian ini dapat mencakup bidang-bidang berikut:
  - `drop_original_metrics` – Opsional. Jika Anda menggunakan bidang `aggregation_dimensions` di bagian `metrics` untuk menggulung metrik ke dalam hasil agregat, maka secara default agen mengirimkan metrik-metrik agregat dan metrik asli yang dipisahkan untuk setiap nilai dimensi. Jika Anda tidak ingin metrik asli dikirim CloudWatch, Anda dapat menentukan parameter ini dengan daftar metrik. Metrik yang ditentukan bersama dengan parameter ini tidak memiliki metrik berdasarkan dimensi yang dilaporkan. CloudWatch Sebaliknya, hanya metrik-metrik agregat saja yang dilaporkan. Hal ini akan mengurangi jumlah metrik yang dikumpulkan oleh agen, dan akan mengurangi biaya Anda.
  - `measurement` – Menentukan susunan metrik netstat yang akan dikumpulkan. Kemungkinan nilai adalah `tcp_close`, `tcp_close_wait`, `tcp_closing`, `tcp_established`, `tcp_fin_wait1`, `tcp_fin_wait2`, `tcp_last_ack`, `tcp_listen`, `tcp_none`, `tcp_syn_sent`, `tcp_syn_recv`, `tcp_time_wait`, dan `udp_socket`. Kolom ini wajib diisi jika Anda menyertakan `netstat`.

    Untuk melihat satuan bawaan untuk setiap `netstat` metrik, silakan lihat Metrik yang dikumpulkan oleh CloudWatch agen di instans Linux dan macOS.

    Di dalam entri untuk setiap metrik individu, secara opsional Anda dapat menentukan salah satu atau kedua hal berikut:
    - `rename` – Menentukan nama yang berbeda untuk metrik ini.
    - `unit` – Menentukan unit yang akan digunakan untuk metrik ini, membatalkan unit default `None` untuk metrik. Unit yang Anda tentukan harus merupakan satuan CloudWatch metrik yang valid, seperti yang tercantum dalam `Unit` deskripsi di MetricDatum.
  - `metrics_collection_interval` – Opsional. Menentukan seberapa sering mengumpulkan metrik netstat, menggulingkan `metrics_collection_interval` yang ditentukan dalam `agent` di file konfigurasi.

    Nilai ini ditentukan dalam detik.

    Jika Anda mengatur nilai ini di bawah 60 detik, setiap metrik dikumpulkan sebagai metrik resolusi tinggi. Untuk informasi selengkapnya tentang metrik resolusi tinggi, silakan lihat Metrik resolusi tinggi.

  - `append_dimensions` – Opsional. Dimensi tambahan untuk digunakan hanya pada metrik netstat. Jika Anda menentukan bidang ini, maka bidang tersebut akan digunakan sebagai tambahan dimensi yang ditentukan dalam bidang `append_dimensions` yang digunakan untuk semua jenis metrik yang dikumpulkan oleh agen.
- `processes` – Opsional. Menentukan bahwa metrik proses akan dikumpulkan. Bagian ini hanya valid untuk instans Linux. Bagian ini dapat mencakup bidang-bidang berikut:
  - `drop_original_metrics` – Opsional. Jika Anda menggunakan bidang `aggregation_dimensions` di bagian `metrics` untuk menggulung metrik ke dalam hasil agregat, maka secara default agen mengirimkan metrik-metrik agregat dan metrik asli yang dipisahkan untuk setiap nilai dimensi. Jika Anda tidak ingin metrik asli dikirim CloudWatch, Anda dapat menentukan parameter ini dengan daftar metrik. Metrik yang ditentukan bersama dengan parameter ini tidak memiliki metrik berdasarkan dimensi yang dilaporkan. CloudWatch Sebaliknya, hanya metrik-metrik agregat saja yang dilaporkan. Hal ini akan mengurangi jumlah metrik yang dikumpulkan oleh agen, dan akan mengurangi biaya Anda.
  - `measurement` – Menentukan rangkaian metrik proses yang akan dikumpulkan. Kemungkinan nilai adalah `blocked`, `dead`, `idle`, `paging`, `running`, `sleeping`, `stopped`, `total`, `total_threads`, `wait`, dan `zombies`. Kolom ini wajib diisi jika Anda menyertakan `processes`.

    Untuk semua `processes` metrik, unit bawaan adalah `None`.

    Di dalam entri untuk setiap metrik individu, secara opsional Anda dapat menentukan salah satu atau kedua hal berikut:

    - `rename` – Menentukan nama yang berbeda untuk metrik ini.
    - `unit` – Menentukan unit yang akan digunakan untuk metrik ini, membatalkan unit default `None` untuk metrik. Unit yang Anda tentukan harus merupakan satuan CloudWatch metrik yang valid, seperti yang tercantum dalam `Unit` deskripsi di MetricDatum.
  - `metrics_collection_interval` – Opsional. Menentukan seberapa sering mengumpulkan metrik proses, menggulingkan `metrics_collection_interval` yang ditentukan dalam `agent` di file konfigurasi.

    Nilai ini ditentukan dalam detik. Misalnya, menentukan 10 metrik yang akan dikumpulkan setiap 10 detik, dan menetapkannya menjadi 300 metrik kustom untuk dikumpulkan setiap 5 menit.

    Jika Anda mengatur nilai ini di bawah 60 detik, setiap metrik dikumpulkan sebagai metrik resolusi tinggi. Untuk informasi selengkapnya, lihat Metrik resolusi tinggi.

  - `append_dimensions` – Opsional. Dimensi tambahan untuk digunakan hanya pada metrik proses. Jika Anda menentukan bidang ini, maka bidang tersebut akan digunakan sebagai tambahan dimensi yang ditentukan dalam bidang `append_dimensions` yang digunakan untuk semua jenis metrik yang dikumpulkan oleh agen.

- `nvidia_gpu` – Opsional. Menentukan bahwa metrik NVIDIA GPU harus dikumpulkan. Bagian ini hanya berlaku untuk instans Linux pada host yang dikonfigurasi dengan akselerator GPU NVIDIA dan melakukan instalasi Antarmuka Manajemen Sistem NVIDIA (nvidia-smi).

  Metrik GPU NVIDIA yang dikumpulkan diawali dengan string `nvidia_smi_` untuk membedakannya dari metrik yang dikumpulkan untuk jenis akselerator lainnya. Bagian ini dapat mencakup bidang-bidang berikut:

  - `drop_original_metrics` – Opsional. Jika Anda menggunakan bidang `aggregation_dimensions` di bagian `metrics` untuk menggulung metrik ke dalam hasil agregat, maka secara default agen mengirimkan metrik-metrik agregat dan metrik asli yang dipisahkan untuk setiap nilai dimensi. Jika Anda tidak ingin metrik asli dikirim CloudWatch, Anda dapat menentukan parameter ini dengan daftar metrik. Metrik yang ditentukan bersama dengan parameter ini tidak memiliki metrik berdasarkan dimensi yang dilaporkan. CloudWatch Sebaliknya, hanya metrik-metrik agregat saja yang dilaporkan. Hal ini akan mengurangi jumlah metrik yang dikumpulkan oleh agen, dan akan mengurangi biaya Anda.
  - `measurement` – Menentukan larik metrik NVIDIA GPU yang akan dikumpulkan. Untuk daftar nilai yang mungkin digunakan di sini, silakan lihat kolom Metrik dalam tabel di[Kumpulkan metrik GPU NVIDIA](#).

    Di dalam entri untuk setiap metrik individu, Anda dapat menentukan salah satu atau kedua hal berikut secara opsional:

    - `rename` – Menentukan nama yang berbeda untuk metrik ini.
    - `unit` – Menentukan unit yang akan digunakan untuk metrik ini, membatalkan unit default `None` untuk metrik. Unit yang Anda tentukan harus merupakan satuan CloudWatch metrik yang valid, seperti yang tercantum dalam `Unit` deskripsi di [MetricDatum](#).
  - `metrics_collection_interval` – Opsional. Menentukan seberapa sering mengumpulkan metrik GPU NVIDIA, membatalkan global yang `metrics_collection_interval` ditentukan dalam `agent` bagian file konfigurasi.

- `procstat` – Opsional. Menjelaskan bahwa Anda ingin mengambil metrik dari proses individu. Untuk informasi selengkapnya tentang opsi konfigurasi yang tersedia untuk procstat, silakan lihat [Kumpulkan Metrik Proses dengan Plugin procstat](#).

- `statsd` – Opsional. Menentukan bahwa Anda ingin mengambil metrik kustom menggunakan `StatsD` protokol. CloudWatch Agen bertindak sebagai daemon untuk protokol. Anda menggunakan `StatsD` klien standar apa pun untuk mengirim metrik ke CloudWatch agen. Untuk informasi selengkapnya tentang opsi konfigurasi yang tersedia untuk StatsD, silakan lihat Ambil metrik kustom dengan StatSD .
- `ethtool` – Opsional. Menentukan bahwa Anda ingin mengambil metrik jaringan menggunakan plugin `ethtool`. Plugin ini dapat mengimpor metrik yang dikumpulkan oleh utilitas ettool standar, serta metrik performa jaringan dari instans Amazon EC2. Untuk informasi selengkapnya tentang opsi konfigurasi yang tersedia untuk ethtool, silakan lihat Mengumpulkan metrik performa jaringan.

Berikut ini adalah contoh `metrics` untuk server Linux. Dalam contoh ini, tiga metrik CPU, tiga metrik netstat, tiga metrik proses, dan satu metrik disk dikumpulkan, dan agen diatur untuk menerima metrik tambahan dari `collectd` klien mereka.

```
"metrics": {
    "aggregation_dimensions" : [["AutoScalingGroupName"], ["InstanceId",
 "InstanceType"],[]],
    "metrics_collected": {
      "collectd": {},
      "cpu": {
        "resources": [
          "*"
        ],
        "measurement": [
          {"name": "cpu_usage_idle", "rename": "CPU_USAGE_IDLE", "unit": "Percent"},
          {"name": "cpu_usage_nice", "unit": "Percent"},
          "cpu_usage_guest"
        ],
        "totalcpu": false,
        "drop_original_metrics": [ "cpu_usage_guest" ],
        "metrics_collection_interval": 10,
        "append_dimensions": {
          "test": "test1",
          "date": "2017-10-01"
        }
      },
      "netstat": {
        "measurement": [
          "tcp_established",
          "tcp_syn_sent",
```

```
          "tcp_close"
        ],
        "metrics_collection_interval": 60
      },
       "disk": {
        "measurement": [
          "used_percent"
        ],
        "resources": [
          "*"
        ],
        "drop_device": true
      },
      "processes": {
        "measurement": [
          "running",
          "sleeping",
          "dead"
        ]
      }
    },
    "append_dimensions": {
      "ImageId": "${aws:ImageId}",
      "InstanceId": "${aws:InstanceId}",
      "InstanceType": "${aws:InstanceType}",
      "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
    }
  }
```

Server Windows

Di bagian `metrics_collected` untuk Windows Server, Anda dapat memiliki subbagian untuk setiap objek performa Windows, seperti `Memory`, `Processor`, dan `LogicalDisk`. Untuk informasi tentang apa objek dan counter tersedia, silakan lihat Performance Counters di dokumentasi Microsoft Windows.

Dalam subbagian untuk setiap objek, Anda menentukan `measurement` baris konter untuk mengumpulkan. `measurement` larik diperlukan untuk setiap objek yang Anda tentukan dalam file konfigurasi. Anda juga dapat menentukan `resources` untuk menamai kejadian untuk mengumpulkan metrik. Anda juga dapat menentukan `*` untuk `resources` untuk mengumpulkan metrik terpisah untuk setiap instans. Jika Anda menghilangkan `resources` penghitung yang memiliki instans, data untuk semua instans dikumpulkan menjadi satu set. Jika Anda menghilangkan

`resources` penghitung yang tidak memiliki instance, penghitung tidak dikumpulkan oleh agen. CloudWatch Untuk menentukan apakah counter memiliki instans, Anda dapat menggunakan salah satu dari perintah berikut.

PowerShell:

```
Get-Counter -ListSet *
```

Baris perintah (bukan Powershell):

```
TypePerf.exe -q
```

Dalam setiap bagian objek, Anda juga dapat menentukan kolom opsional berikut:

- `metrics_collection_interval` – Opsional. Menentukan seberapa sering mengumpulkan metrik untuk objek ini, menggulingkan `metrics_collection_interval` yang ditentukan dalam `agent` di file konfigurasi.

  Nilai ini ditentukan dalam detik. Misalnya, menentukan 10 metrik yang akan dikumpulkan setiap 10 detik, dan menetapkannya menjadi 300 metrik kustom untuk dikumpulkan setiap 5 menit.

  Jika Anda mengatur nilai ini di bawah 60 detik, setiap metrik dikumpulkan sebagai metrik resolusi tinggi. Untuk informasi selengkapnya, lihat Metrik resolusi tinggi.
- `append_dimensions` – Opsional. Menentukan dimensi tambahan untuk digunakan hanya pada metrik untuk objek ini. Jika Anda menentukan bidang ini, maka bidang tersebut akan digunakan sebagai tambahan dimensi yang ditentukan dalam bidang `append_dimensions` global yang digunakan untuk semua jenis metrik yang dikumpulkan oleh agen.
- `drop_original_metrics` – Opsional. Jika Anda menggunakan bidang `aggregation_dimensions` di bagian `metrics` untuk menggulung metrik ke dalam hasil agregat, maka secara default agen mengirimkan metrik-metrik agregat dan metrik asli yang dipisahkan untuk setiap nilai dimensi. Jika Anda tidak ingin metrik asli dikirim CloudWatch, Anda dapat menentukan parameter ini dengan daftar metrik. Metrik yang ditentukan bersama dengan parameter ini tidak memiliki metrik berdasarkan dimensi yang dilaporkan. CloudWatch Sebaliknya, hanya metrik-metrik agregat saja yang dilaporkan. Hal ini akan mengurangi jumlah metrik yang dikumpulkan oleh agen, dan akan mengurangi biaya Anda.

Dalam setiap bagian penghitung, Anda juga dapat menentukan kolom opsional berikut:

- `rename`- Menentukan nama yang berbeda untuk digunakan dalam CloudWatch untuk metrik ini.
- `unit` – Menentukan unit yang akan digunakan untuk metrik ini. Unit yang Anda tentukan harus merupakan satuan CloudWatch metrik yang valid, seperti yang tercantum dalam `Unit` deskripsi di MetricDatum.

Ada dua bagian opsional lain yang dapat Anda sertakan di `metrics_collected`:

- `statsd` – Memungkinkan Anda mengambil kembali metrik kustom menggunakan `StatsD` protokol. CloudWatch Agen bertindak sebagai daemon untuk protokol. Anda menggunakan `StatsD` klien standar apa pun untuk mengirim metrik ke CloudWatch agen. Untuk informasi selengkapnya, lihat Ambil metrik kustom dengan StatSD .
- `procstat` – Memungkinkan Anda mengambil metrik dari proses individu. Untuk informasi selengkapnya, lihat Kumpulkan Metrik Proses dengan Plugin procstat.

Berikut ini adalah contoh `metrics` untuk digunakan di Server Windows. Dalam contoh ini, banyak metrik Windows yang dikumpulkan, dan komputer juga diatur untuk menerima metrik tambahan dari `StatsD` klien mereka.

```
"metrics": {
    "metrics_collected": {
      "statsd": {},
      "Processor": {
        "measurement": [
          {"name": "% Idle Time", "rename": "CPU_IDLE", "unit": "Percent"},
          "% Interrupt Time",
          "% User Time",
          "% Processor Time"
        ],
        "resources": [
          "*"
        ],
        "append_dimensions": {
          "d1": "win_foo",
          "d2": "win_bar"
        }
      },
      "LogicalDisk": {
        "measurement": [
          {"name": "% Idle Time", "unit": "Percent"},
```

```
          {"name": "% Disk Read Time", "rename": "DISK_READ"},
          "% Disk Write Time"
        ],
        "resources": [
          "*"
        ]
      },
      "Memory": {
        "metrics_collection_interval": 5,
        "measurement": [
          "Available Bytes",
          "Cache Faults/sec",
          "Page Faults/sec",
          "Pages/sec"
        ],
        "append_dimensions": {
          "d3": "win_bo"
        }
      },
      "Network Interface": {
        "metrics_collection_interval": 5,
        "measurement": [
          "Bytes Received/sec",
          "Bytes Sent/sec",
          "Packets Received/sec",
          "Packets Sent/sec"
        ],
        "resources": [
          "*"
        ],
        "append_dimensions": {
          "d3": "win_bo"
        }
      },
      "System": {
        "measurement": [
          "Context Switches/sec",
          "System Calls/sec",
          "Processor Queue Length"
        ],
        "append_dimensions": {
          "d1": "win_foo",
          "d2": "win_bar"
        }
```

```
      }
    },
    "append_dimensions": {
      "ImageId": "${aws:ImageId}",
      "InstanceId": "${aws:InstanceId}",
      "InstanceType": "${aws:InstanceType}",
      "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
    },
    "aggregation_dimensions" : [["ImageId"], ["InstanceId", "InstanceType"], ["d1"],[]]
    }
  }
```

CloudWatch file konfigurasi agen: Bagian Log

`logs` bagian ini mencakup kolom berikut:

- `logs_collected` – Wajib jika bagian `logs` ini disertakan. Menentukan file log dan log kejadian Windows mana yang harus dikumpulkan dari server. Ini dapat mencakup dua bidang, `files` dan `windows_events`.
  - `files`— Menentukan file log reguler yang CloudWatch agen adalah untuk mengumpulkan. Ini berisi satu bidang, `collect_list`, yang selanjutnya mendefinisikan file-file ini.
    - `collect_list` – Wajib jika `files` disertakan. Berisi serangkaian entri, masing-masing menentukan satu file log yang akan dikumpulkan. Masing-masing entri ini dapat menyertakan kolom berikut:
      - `file_path`- Menentukan jalur file log untuk meng-upload ke CloudWatch Log. Aturan pencocokan lob Unix standar diterima, dengan tambahan `**` sebagai bintang super besar. Misalnya, menentukan `/var/log/**.log` menyebabkan semua `.log` file dalam `/var/log` pohon direktori yang akan dikumpulkan. Untuk contoh lainnya, silakan lihat Glob Library.

        Anda juga dapat menggunakan tanda bintang standar sebagai wildcard standar. Misalnya, `/var/log/system.log*` sesuaikan file seperti `system.log_1111`, `system.log_2222`, dan lainnya di `/var/log`.

        Hanya file terbaru yang didorong ke CloudWatch Log berdasarkan waktu modifikasi file. Kami merekomendasikan bahwa Anda menggunakan wildcard untuk menentukan serangkaian file dari jenis yang sama, seperti `access_log.2018-06-01-01` dan `access_log.2018-06-01-02`, tetapi tidak banyak jenis file, seperti `access_log_80` dan

`access_log_443`. Untuk menentukan beberapa jenis file, tambahkan log stream lain ke file konfigurasi agen sehingga setiap jenis file log pergi ke aliran log yang berbeda.

- `auto_removal` – Opsional. Jika ini`true`, CloudWatch agen secara otomatis menghapus file log ini setelah membacanya dan telah diputar. Biasanya file log dihapus setelah seluruh isinya diunggah ke CloudWatch Log, tetapi jika agen mencapai EOF (akhir file) dan juga mendeteksi file log baru lainnya yang cocok dengan yang sama`file_path`, agen menghapus file LAMA, jadi Anda harus memastikan bahwa Anda selesai menulis ke file LAMA sebelum membuat file BARU. Pustaka penelusuran RUST memiliki ketidakcocokan yang diketahui karena berpotensi membuat file log BARU dan kemudian masih mencoba menulis ke file log LAMA.

  Agen hanya menghapus file lengkap dari log yang membuat beberapa file, seperti log yang membuat file terpisah untuk setiap tanggal. Jika lg terus menerus menulis ke file tunggal, log tersebut tidak dihapus.

  Jika Anda sudah memiliki metode rotasi atau penghapusan file log, sebaiknya hapus kolom ini atau atur menjadi `false`.

  Jika Anda menghilangkan kolom ini, nilai bawaan dari `false` digunakan.

- `log_group_name` – Opsional. Menentukan apa yang harus digunakan sebagai nama grup log di CloudWatch Log.

  Kami sarankan Anda untuk menggunakan bidang ini untuk menentukan nama grup log untuk mencegah kebingungan. Jika Anda menghilangkan `log_group_name`, nilai dari `file_path` hingga titik terakhir digunakan sebagai nama grup log. Sebagai contoh, jika alur filenya `/tmp/TestLogFile.log.2017-07-11-14`, nama grup log adalah `/tmp/TestLogFile.log`.

  Jika Anda menentukan sebuah nama grup log, Anda dapat menggunakan `{instance_id}`, `{hostname}`, `{local_hostname}`, dan `{ip_address}` sebagai variabel dalam nama tersebut. `{hostname}` mengambil nama host dari metadata EC2, dan `{local_hostname}` menggunakan nama host dari file konfigurasi jaringan.

  Jika Anda menggunakan variabel ini untuk membuat banyak grup log yang berbeda, ingatlah batas 1.000.000 grup log per Wilayah per akun.

  Karakter yang diperbolehkan termasuk a–z, A–Z, 0–9, '_' (garis bawah), '-' (tanda hubung), '/' (garis miring berikutnya), dan '.' (period).

- `log_group_class` – Opsional. Menentukan kelas grup log mana yang akan digunakan untuk grup log baru. Untuk informasi selengkapnya tentang kelas grup log, silakan lihat Kelas log.

  Nilai yang valid adalah `STANDARD` dan `INFREQUENT_ACCESS`. Jika Anda menghilangkan bidang ini, `STANDARD` default akan digunakan.

  > **Important**
  >
  > Setelah sebuah grup log dibuat, kelas log itu tidak dapat diubah.

- `log_stream_name` – Opsional. Menentukan apa yang harus digunakan sebagai nama log stream di CloudWatch Log. Sebagai bagian dari nama, Anda dapat menggunakan `{instance_id}`, `{hostname}`, `{local_hostname}`, dan `{ip_address}` sebagai variabel dalam nama. `{hostname}` mengambil nama host dari metadata EC2, dan `{local_hostname}` menggunakan nama host dari file konfigurasi jaringan.

  Jika Anda menghilangkan kolom ini, nilai `log_stream_name` di seluruh dunia `logs` yang digunakan. Jika itu juga dihilangkan, nilai bawaan `{instance_id}` digunakan.

  Jika belum ada, maka aliran log stream akan dibuat secara otomatis.

- `retention_in_days` – Opsional. Menentukan jumlah hari untuk mempertahankan log acara dalam grup log yang ditentukan.
  - Jika agen tersebut membuat grup log ini sekarang, dan Anda menghilangkan bidang ini, maka retensi grup log baru ini akan diatur untuk tidak pernah mengalami kedaluwarsa.
  - Jika grup log ini sudah ada dan Anda menentukan bidang ini, maka retensi baru yang Anda tentukan akan digunakan. Jika Anda menghilangkan bidang ini untuk grup log yang sudah ada, maka retensi grup log tidak akan diubah.

    Wizard CloudWatch agen digunakan `-1` sebagai nilai default untuk bidang ini saat digunakan untuk membuat file konfigurasi agen dan Anda tidak menentukan nilai untuk retensi log. `-1`Nilai yang ditetapkan oleh wizard ini menentukan bahwa peristiwa dalam grup log tidak akan pernah kedaluwarsa. Namun demikian, mengedit nilai ini secara manual tidak `-1` berpengaruh.

  Nilai yang valid adalah 1, 3, 5, 7, 14, 30, 60, 90, 120, 150, 180, 365, 400, 545, 731, 1.827, 2.192, 2.557, 2.922, 3.288, dan 3.653

Jika Anda mengonfigurasi agen untuk menulis beberapa aliran log stream ke grup log yang sama, menentukan `retention_in_days` di satu tempat akan mengatur retensi log untuk seluruh grup log. Jika Anda menentukan `retention_in_days` untuk grup log yang sama di beberapa tempat, maka retensi akan disetel jika semua nilai tersebut sama. Namun demikian, jika `retention_in_days` nilai yang berbeda ditentukan untuk grup log yang sama di beberapa tempat, retensi log tidak akan disetel dan agen akan berhenti, mengembalikan kesalahan.

> **Note**
>
> Peran IAM atau pengguna IAM yang dimiliki agen harus memiliki `logs:PutRetentionPolicy` agar dapat menetapkan kebijakan retensi. Untuk informasi selengkapnya, lihat Mengizinkan CloudWatch agen menyetel kebijakan retensi log.

> **Warning**
>
> Jika Anda menetapkan `retention_in_days` untuk sebuah grup log yang sudah ada, maka semua log dalam grup log yang diterbitkan sebelum jumlah hari yang Anda tentukan akan dihapus. Sebagai contoh, menyetelnya menjadi 3 akan menyebabkan semua log dari 3 hari yang lalu dan sebelumnya dihapus.

- `filters` – Opsional. Dapat berisi array entri, yang masing-masing menentukan ekspresi reguler dan jenis filter untuk menentukan apakah akan menerbitkan atau menjatuhkan entri log yang cocok dengan filter. Jika Anda menghilangkan bidang ini, semua log dalam file log dipublikasikan ke CloudWatch Log. Jika Anda menyertakan bidang ini, agen akan memproses setiap pesan log dengan semua filter yang Anda tentukan, dan hanya peristiwa log yang meneruskan semua filter yang dipublikasikan ke CloudWatch Log. Entri log yang tidak melewati semua filter akan tetap berada di file log host, tetapi tidak akan dikirim ke CloudWatch Log.

  Setiap entri dalam larik filter dapat mencakup bidang berikut:

  - `type`– Menunjukkan jenis filter. Nilai yang valid adalah `include` dan `exclude`. Dengan`include`, entri log harus cocok dengan ekspresi yang akan dipublikasikan ke

CloudWatch Log. Dengan`exclude`, setiap entri log yang cocok dengan filter tidak dikirim ke CloudWatch Log.

- `expression`– Sebuah string ekspresi reguler yang mengikuti Sintaks RE2.

> **Note**
>
> CloudWatch Agen tidak memeriksa kinerja ekspresi reguler apa pun yang Anda berikan, atau membatasi waktu proses evaluasi ekspresi reguler. Kami menyarankan Anda berhati-hati untuk tidak menulis ekspresi yang sulit untuk dievaluasi. Untuk informasi selengkapnya tentang kemungkinan masalah, lihat Ekspresi reguler Denial of Service - ReDo S

Misalnya, kutipan berikut dari file konfigurasi CloudWatch agen menerbitkan log yang PUT dan POST permintaan ke Log, tetapi tidak termasuk CloudWatch log yang berasal dari Firefox.

```
"collect_list": [
  {
    "file_path": "/opt/aws/amazon-cloudwatch-agent/logs/test.log",
    "log_group_name": "test.log",
    "log_stream_name": "test.log",
    "filters": [
      {
        "type": "exclude",
        "expression": "Firefox"
      },
      {
        "type": "include",
        "expression": "P(UT|OST)"
      }
    ]
  },
  .....
]
```

> **Note**
>
> Urutan filter dalam file konfigurasi penting untuk performa. Dalam contoh sebelumnya, agen menjatuhkan semua log yang cocok `Firefox` sebelum mulai

mengevaluasi filter kedua. Untuk menyebabkan lebih sedikit entri log dievaluasi oleh lebih dari satu filter, letakkan filter yang Anda harapkan untuk mengesampingkan lebih banyak log terlebih dahulu di file konfigurasi.

- `timezone` – Opsional. Menentukan zona waktu yang akan digunakan saat menempatkan stempel waktu pada kejadian log. Nilai yang valid adalah `UTC` dan `Local`. Nilai bawaannya adalah `Local`.

  Parameter ini diabaikan jika Anda tidak menentukan nilai untuk `timestamp_format`.

- `timestamp_format` – Opsional. Menentukan format stempel waktu, menggunakan teks biasa dan simbol khusus yang dimulai dengan %. Jika Anda menghilangkan bidang ini, waktu saat ini akan digunakan. Jika menggunakan kolom ini, Anda dapat menggunakan simbol dalam daftar berikut sebagai bagian dari format.

  Jika satu entri log berisi dua stempel waktu yang cocok dengan format, maka stempel pertama akan digunakan.

  Daftar simbol ini berbeda dari daftar yang digunakan oleh agen CloudWatch Log lama. Untuk rangkuman perbedaan ini, silakan lihat Perbedaan stempel waktu antara CloudWatch agen terpadu dan agen Log sebelumnya CloudWatch

  `%y`

  Tahun tanpa abad sebagai angka desimal penambah nol. Misalnya, 19 untuk mewakili 2019.

  `%Y`

  Tahun dengan abad sebagai angka desimal. Misalnya, 2019.

  `%b`

  Bulan sebagai nama singkat lokal

  `%B`

  Bulan sebagai nama lengkap daerah

  `%m`

  Bulan sebagai angka desimal bernilai nol

`%-m`

Bulan sebagai angka desimal (bukan nol-tambahan)

`%d`

Tanggal dalam bulan sebagai angka desimal bernilai nol

`%-d`

Tanggal dalam bulan sebagai angka desimal (bukan nol-tambahan)

`%A`

Nama lengkap hari kerja, seperti `Monday`

`%a`

Singkatan hari kerja, seperti `Mon`

`%H`

Jam (dalam waktu 24 jam) sebagai angka desimal bernilai nol

`%I`

Jam (dalam waktu 12 jam) sebagai angka desimal bernilai nol

`%-I`

Jam (dalam waktu 12 jam) sebagai angka desimal (bukan nol tambahan)

`%p`

Pagi atau Sore

`%M`

Menit sebagai angka desimal penambah nol

`%-M`

Menit sebagai angka desimal (bukan penambahan nol)

`%S`

Detik sebagai angka desimal nol-padded

`%-S`

Detik sebagai angka desimal (bukan nol padded)

`%f`

detik pecah sebagai angka desimal (1-9 digit), penambahan nol di sebelah kiri.

`%Z`

Zona waktu, misalnya PST

`%z`

Zona waktu, dinyatakan sebagai offset antara zona waktu lokal dan UTC. Sebagai contoh, `-0700`. Hanya format ini yang didukung. Misalnya, `-07:00` bukan format yang valid.

- `multi_line_start_pattern` – Menentukan pola untuk mengidentifikasi awal pesan log. Pesan log dibuat dari garis yang sesuai dengan pola dan setiap garis berikutnya yang tidak cocok dengan pola.

  Jika Anda menghilangkan kolom ini, mode multi-line dinonaktifkan, dan semua garis yang dimulai dengan karakter non-ruangan putih menutup pesan log sebelumnya dan memulai pesan log baru.

  Jika Anda menyertakan bidang ini, Anda dapat menentukan `{timestamp_format}` menggunakan ekspresi reguler yang sama dengan format stempel waktu Anda. Jika tidak, Anda dapat menentukan ekspresi reguler yang berbeda untuk CloudWatch Log yang akan digunakan untuk menentukan baris awal entri multi-baris.
- `encoding` – Menentukan pengkodean file log sehingga dapat dibaca dengan benar. Jika Anda menentukan kode yang salah, mungkin ada kehilangan data karena karakter yang tidak dapat didekodekan diganti dengan karakter lain.

  Nilai bawaannya adalah `utf-8`. Berikut ini adalah semua nilai yang mungkin:

  `ascii, big5, euc-jp, euc-kr, gbk, gb18030, ibm866, iso2022-jp, iso8859-2, iso8859-3, iso8859-4, iso8859-5, iso8859-6, iso8859-7, iso8859-8, iso8859-8-i, iso8859-10, iso8859-13, iso8859-14, iso8859-15, iso8859-16, koi8-r, koi8-u, macintosh, shift_jis, utf-8, utf-16, utf-16le, UTF-16, UTF-16LE, windows-874, windows-1250,`

`windows-1251, windows-1252, windows-1253, windows-1254, windows-1255, windows-1256, windows-1257, windows-1258, x-mac-cyrillic`

- `windows_events` menentukan jenis peristiwa Windows yang dikumpulkan dari server yang menjalankan Server Windows. Ini mencakup kolom-kolom berikut:
  - `collect_list` – Wajib jika `windows_events` disertakan. Menentukan jenis dan tingkat acara Windows yang akan dikumpulkan. Setiap log yang akan dikumpulkan memiliki entri di bagian ini, yang dapat mencakup kolom berikut:
    - `event_name` – Menentukan jenis peristiwa Windows untuk log. Ini setara dengan nama saluran log peristiwa Windows: misalnya, `System`, `Security`, `Application`, dan sebagainya. Bidang ini diperlukan untuk setiap jenis peristiwa Windows yang akan dicatat log-nya.

      > **Note**
      >
      > Saat CloudWatch mengambil pesan dari saluran log Windows, ia mencari saluran log berdasarkan `Full Name` propertinya. Sementara itu, panel navigasi Peraga Peristiwa Windows menampilkan properti `Log Name` saluran log. `Full Name` dan `Log Name` tidak selalu cocok. Untuk mengonfirmasi saluran `Full Name`, klik kanan di penampil Windows Event dan buka Properti.

    - `event_levels` – Menentukan tingkat-tingkat kejadian yang akan dicatat lognya. Anda harus menentukan setiap level yang akan dicatat log-nya. Nilai yang mungkin termasuk adalah `INFORMATION`, `WARNING`, `ERROR`, `CRITICAL`, dan `VERBOSE`. Bidang ini diperlukan untuk setiap jenis peristiwa Windows yang akan dicatat log-nya.
    - `log_group_name` – Wajib. Menentukan apa yang harus digunakan sebagai nama grup log di CloudWatch Log.
    - `log_stream_name` – Opsional. Menentukan apa yang harus digunakan sebagai nama log stream di CloudWatch Log. Sebagai bagian dari nama, Anda dapat menggunakan `{instance_id}`, `{hostname}`, `{local_hostname}`, dan `{ip_address}` sebagai variabel dalam nama. `{hostname}` mengambil nama host dari metadata EC2, dan `{local_hostname}` menggunakan nama host dari file konfigurasi jaringan.

      Jika Anda menghilangkan kolom ini, nilai `log_stream_name` di seluruh dunia `logs` yang digunakan. Jika itu juga dihilangkan, nilai bawaan `{instance_id}` digunakan.

Jika belum ada, maka aliran log stream akan dibuat secara otomatis.

- `event_format` – Opsional. Menentukan format yang akan digunakan saat menyimpan peristiwa Windows di CloudWatch Log. `xml`menggunakan format XMLseperti pada Windows Event Viewer. `text`menggunakan format agen CloudWatch Log lama.
- `retention_in_days` – Opsional. Menentukan jumlah hari untuk mempertahankan acara Windows di grup log yang ditentukan.
  - Jika agen tersebut membuat grup log ini sekarang, dan Anda menghilangkan bidang ini, maka retensi grup log baru ini akan diatur untuk tidak pernah mengalami kedaluwarsa.
  - Jika grup log ini sudah ada dan Anda menentukan bidang ini, maka retensi baru yang Anda tentukan akan digunakan. Jika Anda menghilangkan bidang ini untuk grup log yang sudah ada, maka retensi grup log tidak akan diubah.

    Wizard CloudWatch agen digunakan `-1` sebagai nilai default untuk bidang ini saat digunakan untuk membuat file konfigurasi agen dan Anda tidak menentukan nilai untuk retensi log. `-1` Nilai ini menentukan ditetapkan oleh pemandu menentukan bahwa peristiwa dalam grup log tidak kedaluwarsa. Namun demikian, mengedit nilai ini secara manual tidak `-1` berpengaruh.

  Nilai yang valid adalah 1, 3, 5, 7, 14, 30, 60, 90, 120, 150, 180, 365, 400, 545, 731, 1.827, 2.192, 2.557, 2.922, 3.288, dan 3.653

  Jika Anda mengonfigurasi agen untuk menulis beberapa aliran log stream ke grup log yang sama, menentukan `retention_in_days` di satu tempat akan mengatur retensi log untuk seluruh grup log. Jika Anda menentukan `retention_in_days` untuk grup log yang sama di beberapa tempat, maka retensi akan disetel jika semua nilai tersebut sama. Namun demikian, jika `retention_in_days` nilai yang berbeda ditentukan untuk grup log yang sama di beberapa tempat, retensi log tidak akan disetel dan agen akan berhenti, mengembalikan kesalahan.

> **Note**
>
> Peran IAM atau pengguna IAM yang dimiliki agen harus memiliki `logs:PutRetentionPolicy` agar dapat menetapkan kebijakan retensi. Untuk informasi selengkapnya, lihat Mengizinkan CloudWatch agen menyetel kebijakan retensi log.

> **Warning**
>
> Jika Anda menetapkan `retention_in_days` untuk sebuah grup log yang sudah ada, maka semua log dalam grup log yang diterbitkan sebelum jumlah hari yang Anda tentukan akan dihapus. Sebagai contoh, menyetelnya menjadi 3 akan menyebabkan semua log dari 3 hari yang lalu dan sebelumnya dihapus.

- `log_stream_name` – Wajib. Menentukan nama aliran log stream default yang akan digunakan untuk setiap log atau kejadian Windows yang tidak memiliki nama aliran log stream individu yang ditentukan di parameter `log_stream_name` dalam entri mereka di `collect_list`.
- `endpoint_override` – Menentukan titik akhir FIPS atau tautan privat untuk digunakan sebagai titik akhir tempat agen mengirimkan log. Menentukan bidang ini dan mengatur tautan privat memungkinkan Anda mengirim log ke titik akhir VPC Amazon. Untuk informasi selengkapnya, silakan lihat Apa Itu Amazon VPC?.

  Nilai dari `endpoint_override` harus string yang berupa URL.

  Misalnya, bagian log berikut pada file konfigurasi mengatur agen untuk menggunakan Titik Akhir VPC saat mengirim log.

```
{
  "logs": {
    "endpoint_override": "vpce-XXXXXXXXXXXXXXXXXXXXXXX.logs.us-
east-1.vpce.amazonaws.com",
   ......
   },
}
```

- `force_flush_interval` – Menentukan dalam hitungan detik lamanya waktu maksimum log tetap berada dalam buffer memori sebelum dikirim ke server. Apa pun pengaturan yang Anda tetapkan untuk bidang ini, jika ukuran log dalam penyangga mencapai 1 MB, log akan segera dikirim ke server saat itu juga. Nilai bawaannya adalah 5.

  Jika Anda menggunakan agen untuk melaporkan metrik resolusi tinggi dalam format metrik tersemat, dan Anda menyetel alarm pada metrik tersebut, tetapkan parameter ini ke nilai default 5. Jika tidak, metrik dilaporkan dengan penundaan yang dapat menyebabkan kekhawatiran pada data sebagian atau tidak lengkap.

- `credentials`- Menentukan peran IAM untuk digunakan saat mengirim log ke akun yang berbeda AWS . Jika ditentukan, bidang ini berisi satu parameter, `role_arn`.
  - `role_arn`— Menentukan ARN dari peran IAM untuk digunakan untuk otentikasi saat mengirim log ke akun yang berbeda. AWS Untuk informasi selengkapnya, lihat Mengirim metrik, log, dan jejak ke akun lain. Jika disebutkan di sini, ini akan menggantikan `role_arn` yang ditentukan dalam `agent` dari file konfigurasi, jika ada.
- `metrics_collected`— Bidang ini dapat berisi bagian untuk menentukan bahwa agen akan mengumpulkan log untuk mengaktifkan kasus penggunaan seperti Sinyal CloudWatch Aplikasi dan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS.
  - `app_signals`(Opsional) Menentukan bahwa Anda ingin mengaktifkan Sinyal CloudWatch Aplikasi Untuk informasi lebih lanjut tentang konfigurasi ini, lihatAktifkan Sinyal CloudWatch Aplikasi.
  - `kubernetes` – Bidang ini dapat berisi `enhanced_container_insights` parameter, yang dapat Anda gunakan untuk mengaktifkan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS.
    - `enhanced_container_insights` – Setel ini ke `true` untuk mengaktifkan Insights kontainer dengan peningkatan observabilitas untuk Amazon EKS. Untuk informasi selengkapnya, lihat Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS.
    - `accelerated_compute_metrics`— Setel ini `false` untuk memilih keluar dari pengumpulan metrik GPU Nvidia di kluster Amazon EKS. Untuk informasi selengkapnya, lihat Metrik GPU NVIDIA.
  - `emf` – Untuk mengumpulkan metrik yang disematkan dalam log, tidak perlu lagi menambahkan `emf` bidang ini. Ini adalah bidang warisan yang menentukan bahwa agen akan mengumpulkan log yang berada dalam format metrik tersemat. Anda dapat membuat data metrik dari log ini. Untuk informasi selengkapnya, lihat Menyematkan metrik dalam log.

Berikut ini adalah contoh bagian `logs`.

```
"logs":
   {
       "logs_collected": {
           "files": {
               "collect_list": [
                   {
                       "file_path": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\
\Logs\\amazon-cloudwatch-agent.log",
```

```
                        "log_group_name": "amazon-cloudwatch-agent.log",
                        "log_stream_name": "my_log_stream_name_1",
                        "timestamp_format": "%H: %M: %S%y%b%-d"
                    },
                    {
                        "file_path": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\
\Logs\\test.log",
                        "log_group_name": "test.log",
                        "log_stream_name": "my_log_stream_name_2"
                    }
                ]
            },
            "windows_events": {
                "collect_list": [
                    {
                        "event_name": "System",
                        "event_levels": [
                            "INFORMATION",
                            "ERROR"
                        ],
                        "log_group_name": "System",
                        "log_stream_name": "System"
                    },
                    {
                        "event_name": "CustomizedName",
                        "event_levels": [
                            "INFORMATION",
                            "ERROR"
                        ],
                        "log_group_name": "CustomizedLogGroup",
                        "log_stream_name": "CustomizedLogStream"
                    }
                ]
            }
        },
        "log_stream_name": "my_log_stream_name",
        "metrics_collected": {
            "kubernetes": {
                 "enhanced_container_insights": true
       }
     }
   }
```

CloudWatch file konfigurasi agen: Bagian Jejak

Dengan menambahkan `traces` bagian ke file konfigurasi CloudWatch agen, Anda dapat mengaktifkan Sinyal CloudWatch Aplikasi atau mengumpulkan jejak dari X-Ray dan dari SDK OpenTelemetry instrumentasi dan mengirimkannya ke X-Ray.

> **Important**
>
> Peran IAM agen atau pengguna IAM harus memiliki AWSXrayWriteOnlyAccesskebijakan untuk mengirim data jejak ke X-Ray. Untuk informasi selengkapnya, lihat Buat peran IAM dan pengguna untuk digunakan dengan agen CloudWatch .

Untuk memulai dengan cepat untuk mengumpulkan jejak, Anda dapat menambahkan hanya yang berikut ke file konfigurasi CloudWatch agen.

```
"traces_collected": {
        "xray": {
        },
        "otlp": {
        }
      }
```

Jika Anda menambahkan bagian sebelumnya ke file konfigurasi CloudWatch agen dan memulai ulang agen, ini menyebabkan agen mulai mengumpulkan jejak menggunakan opsi dan nilai default berikut. Untuk informasi selengkapnya tentang parameter ini, silakan lihat definisi parameter nanti di bagian ini.

```
"traces_collected": {
        "xray": {
            "bind_address": "127.0.0.1:2000",
            "tcp_proxy": {
              "bind_address": "127.0.0.1:2000"
            }
        },
        "otlp": {
            "grpc_endpoint": "127.0.0.1:4317",
            "http_endpoint": "127.0.0.1:4318"
        }
      }
```

Bagian `traces` dapat mencakup bidang-bidang berikut:

- `traces_collected` – Wajib jika bagian `traces` ini disertakan. Menentukan SDK mana untuk mengumpulkan jejak dari. Hal ini dapat mencakup kolom berikut:
  - `app_signals` – Opsional. Menentukan bahwa Anda ingin mengaktifkan Sinyal CloudWatch Aplikasi Untuk informasi lebih lanjut tentang konfigurasi ini, lihatAktifkan Sinyal CloudWatch Aplikasi.
  - `xray` – Opsional. Menentukan bahwa Anda ingin mengumpulkan jejak dari X-Ray SDK. Bagian ini dapat mencakup bidang-bidang berikut:
    - `bind_address` – Opsional. Menentukan alamat UDP untuk CloudWatch agen yang akan digunakan untuk mendengarkan jejak X-Ray. Formatnya adalah `ip:port`. Alamat ini harus sesuai dengan alamat yang ditetapkan dalam X-Ray SDK.

      Jika Anda menghilangkan bidang ini, `127.0.0.1:2000` default akan digunakan.
    - `tcp_proxy` – Opsional. Mengkonfigurasi alamat untuk proxy yang digunakan untuk mendukung X-Ray remote sampling. Untuk informasi selengkapnya, silakan lihat Configuring sampling rules dalam dokumentasi X-Ray.

      Bagian ini dapat berisi kolom berikut.
      - `bind_address` – Opsional. Menentukan alamat TCP yang CloudWatch agen harus mengatur proxy. Formatnya adalah `ip:port`. Alamat ini harus sesuai dengan alamat yang ditetapkan dalam X-Ray SDK.

        Jika Anda menghilangkan bidang ini, `127.0.0.1:2000` default akan digunakan.
- `otlp` – Opsional. Menentukan bahwa Anda ingin mengumpulkan jejak dari OpenTelemetry SDK. Untuk informasi selengkapnya tentang AWS Distro OpenTelemetry, lihat AWS Distro untuk. OpenTelemetry Untuk informasi selengkapnya tentang AWS Distro untuk OpenTelemetry SDK, lihat Pendahuluan.

  Bagian ini dapat mencakup bidang-bidang berikut:
  - `grpc_endpoint` – Opsional. Menentukan alamat CloudWatch agen yang akan digunakan untuk mendengarkan OpenTelemetry jejak yang dikirim menggunakan Panggilan Prosedur Jarak Jauh gRPC. Formatnya adalah `ip:port`. Alamat ini harus cocok dengan alamat yang ditetapkan untuk eksportir gRPC di SDK. OpenTelemetry

    Jika Anda menghilangkan bidang ini, `127.0.0.1:4317` default akan digunakan.

  - `http_endpoint` – Opsional. Menentukan alamat untuk CloudWatch agen yang akan digunakan untuk mendengarkan jejak OTLP yang dikirim melalui HTTP. Formatnya adalah `ip:port`. Alamat ini harus cocok dengan alamat yang ditetapkan untuk eksportir HTTP di OpenTelemetry SDK.

    Jika Anda menghilangkan bidang ini, `127.0.0.1:4318` default akan digunakan.
- `concurrency` – Opsional. Menentukan jumlah maksimal panggilan bersamaan ke X-Ray yang dapat digunakan untuk meng-upload jejak. Nilai bawaannya adalah 8.
- `local_mode` – Opsional. Jika `true`, agen tidak mengumpulkan metadata instans Amazon EC2. Bawaannya adalah `false`
- `endpoint_override` – Opsional. Menentukan endpoint FIPS atau link pribadi untuk digunakan sebagai endpoint di mana agen mengirimkan jejak. CloudWatch Menentukan bidang ini dan mengatur tautan privat memungkinkan Anda mengirim jejak ke titik akhir VPC Amazon. Untuk informasi selengkapnya, silakan lihat Apa Itu Amazon VPC

  Nilai dari `endpoint_override` harus berupa string yang berupa URL.
- `region_override` – Opsional. Menentukan Wilayah yang akan digunakan untuk titik akhir X-Ray. CloudWatch Agen mengirimkan jejak ke X-Ray di Wilayah yang ditentukan. Jika Anda menghilangkan bidang ini, agen akan mengirimkan jejak ke Wilayah tempat instans Amazon EC2 berada.

  Jika Anda menentukan Wilayah di sini, itu lebih diutamakan daripada pengaturan `region` parameter di `agent` bagian file konfigurasi.
- `proxy_override` – Opsional. Menentukan alamat server proxy untuk CloudWatch agen yang akan digunakan saat mengirim permintaan ke X-Ray. Protokol server proksi harus ditentukan sebagai bagian dari alamat ini.
- `credentials`— Menentukan peran IAM untuk digunakan saat mengirim jejak ke akun yang berbeda AWS . Jika ditentukan, bidang ini berisi satu parameter, `role_arn`.
  - `role_arn`— Menentukan ARN peran IAM yang akan digunakan untuk otentikasi saat mengirim jejak ke akun yang berbeda. AWS Untuk informasi selengkapnya, lihat Mengirim metrik, log, dan jejak ke akun lain. Jika disebutkan di sini, ini akan menggantikan `role_arn` yang ditentukan dalam `agent` dari file konfigurasi, jika ada.

CloudWatch file konfigurasi agen: Contoh lengkap

Berikut ini adalah contoh file konfigurasi CloudWatch agen lengkap untuk server Linux.

Item yang tercantum dalam `measurement` bagian untuk metrik yang ingin Anda kumpulkan dapat menentukan nama metrik lengkap atau hanya sebagian nama metrik yang akan ditambahkan ke jenis sumber daya. Misalnya, menentukan `reads` atau `diskio_reads` dalam `measurement` bagian dari `diskio` bagian ini akan menyebabkan `diskio_reads` metrik yang akan dikumpulkan.

Contoh ini mencakup kedua cara menentukan metrik dalam `measurement` bagian.

```
   {
     "agent": {
       "metrics_collection_interval": 10,
       "logfile": "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log"
     },
     "metrics": {
       "namespace": "MyCustomNamespace",
       "metrics_collected": {
         "cpu": {
           "resources": [
             "*"
           ],
           "measurement": [
             {"name": "cpu_usage_idle", "rename": "CPU_USAGE_IDLE", "unit":
"Percent"},
             {"name": "cpu_usage_nice", "unit": "Percent"},
             "cpu_usage_guest"
           ],
           "totalcpu": false,
           "metrics_collection_interval": 10,
           "append_dimensions": {
             "customized_dimension_key_1": "customized_dimension_value_1",
             "customized_dimension_key_2": "customized_dimension_value_2"
           }
         },
         "disk": {
           "resources": [
             "/",
             "/tmp"
           ],
           "measurement": [
             {"name": "free", "rename": "DISK_FREE", "unit": "Gigabytes"},
             "total",
             "used"
           ],
            "ignore_file_system_types": [
```

```
            "sysfs", "devtmpfs"
          ],
          "metrics_collection_interval": 60,
          "append_dimensions": {
            "customized_dimension_key_3": "customized_dimension_value_3",
            "customized_dimension_key_4": "customized_dimension_value_4"
          }
        },
        "diskio": {
          "resources": [
            "*"
          ],
          "measurement": [
            "reads",
            "writes",
            "read_time",
            "write_time",
            "io_time"
          ],
          "metrics_collection_interval": 60
        },
        "swap": {
          "measurement": [
            "swap_used",
            "swap_free",
            "swap_used_percent"
          ]
        },
        "mem": {
          "measurement": [
            "mem_used",
            "mem_cached",
            "mem_total"
          ],
          "metrics_collection_interval": 1
        },
        "net": {
          "resources": [
            "eth0"
          ],
          "measurement": [
            "bytes_sent",
            "bytes_recv",
            "drop_in",
```

```
            "drop_out"
          ]
        },
        "netstat": {
          "measurement": [
            "tcp_established",
            "tcp_syn_sent",
            "tcp_close"
          ],
          "metrics_collection_interval": 60
        },
        "processes": {
          "measurement": [
            "running",
            "sleeping",
            "dead"
          ]
        }
      },
      "append_dimensions": {
        "ImageId": "${aws:ImageId}",
        "InstanceId": "${aws:InstanceId}",
        "InstanceType": "${aws:InstanceType}",
        "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
      },
      "aggregation_dimensions" : [["ImageId"], ["InstanceId", "InstanceType"],
 ["d1"],[]],
      "force_flush_interval" : 30
    },
    "logs": {
      "logs_collected": {
        "files": {
          "collect_list": [
            {
              "file_path": "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-
agent.log",
              "log_group_name": "amazon-cloudwatch-agent.log",
              "log_stream_name": "amazon-cloudwatch-agent.log",
              "timezone": "UTC"
            },
            {
              "file_path": "/opt/aws/amazon-cloudwatch-agent/logs/test.log",
              "log_group_name": "test.log",
              "log_stream_name": "test.log",
```

```
                "timezone": "Local"
              }
            ]
          }
        },
        "log_stream_name": "my_log_stream_name",
        "force_flush_interval" : 15,
        "metrics_collected": {
           "kubernetes": {
                "enhanced_container_insights": true
      }
    }
  }
}
```

Berikut ini adalah contoh file konfigurasi CloudWatch agen lengkap untuk server yang menjalankan Windows Server.

```
{
      "agent": {
        "metrics_collection_interval": 60,
        "logfile": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\Logs\\amazon-
cloudwatch-agent.log"
      },
      "metrics": {
        "namespace": "MyCustomNamespace",
        "metrics_collected": {
          "Processor": {
            "measurement": [
              {"name": "% Idle Time", "rename": "CPU_IDLE", "unit": "Percent"},
              "% Interrupt Time",
              "% User Time",
              "% Processor Time"
            ],
            "resources": [
              "*"
            ],
            "append_dimensions": {
              "customized_dimension_key_1": "customized_dimension_value_1",
              "customized_dimension_key_2": "customized_dimension_value_2"
            }
          },
          "LogicalDisk": {
```

```
            "measurement": [
              {"name": "% Idle Time", "unit": "Percent"},
              {"name": "% Disk Read Time", "rename": "DISK_READ"},
              "% Disk Write Time"
            ],
            "resources": [
              "*"
            ]
          },
          "customizedObjectName": {
            "metrics_collection_interval": 60,
            "customizedCounterName": [
              "metric1",
              "metric2"
            ],
            "resources": [
              "customizedInstances"
            ]
          },
          "Memory": {
            "metrics_collection_interval": 5,
            "measurement": [
              "Available Bytes",
              "Cache Faults/sec",
              "Page Faults/sec",
              "Pages/sec"
            ]
          },
          "Network Interface": {
            "metrics_collection_interval": 5,
            "measurement": [
              "Bytes Received/sec",
              "Bytes Sent/sec",
              "Packets Received/sec",
              "Packets Sent/sec"
            ],
            "resources": [
              "*"
            ],
            "append_dimensions": {
              "customized_dimension_key_3": "customized_dimension_value_3"
            }
          },
          "System": {
```

```
            "measurement": [
              "Context Switches/sec",
              "System Calls/sec",
              "Processor Queue Length"
            ]
          }
        },
        "append_dimensions": {
          "ImageId": "${aws:ImageId}",
          "InstanceId": "${aws:InstanceId}",
          "InstanceType": "${aws:InstanceType}",
          "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
        },
        "aggregation_dimensions" : [["ImageId"], ["InstanceId", "InstanceType"],
 ["d1"],[]]
      },
      "logs": {
        "logs_collected": {
          "files": {
            "collect_list": [
              {
                "file_path": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\Logs\
\amazon-cloudwatch-agent.log",
                "log_group_name": "amazon-cloudwatch-agent.log",
                "timezone": "UTC"
              },
              {
                "file_path": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\Logs\
\test.log",
                "log_group_name": "test.log",
                "timezone": "Local"
              }
            ]
          },
          "windows_events": {
            "collect_list": [
              {
                "event_name": "System",
                "event_levels": [
                  "INFORMATION",
                  "ERROR"
                ],
                "log_group_name": "System",
                "log_stream_name": "System",
```

```
            "event_format": "xml"
          },
          {
            "event_name": "CustomizedName",
            "event_levels": [
              "WARNING",
              "ERROR"
            ],
            "log_group_name": "CustomizedLogGroup",
            "log_stream_name": "CustomizedLogStream",
            "event_format": "xml"
          }
        ]
      }
    },
    "log_stream_name": "example_log_stream_name"
  }
}
```

Simpan file konfigurasi CloudWatch agen secara manual

Jika Anda membuat atau mengedit file konfigurasi CloudWatch agen secara manual, Anda dapat memberikan nama apa pun. Untuk kemudahan dalam pemecahan masalah, kami sarankan Anda menamainya `/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json` di server Linux dan `$Env:ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.json` pada server yang menjalankan Server Windows. Setelah Anda membuat file, Anda dapat menyalinnya ke server lain tempat Anda ingin menjalankan agen.

Mengunggah file konfigurasi CloudWatch agen ke Systems Manager Parameter Store

Jika Anda berencana menggunakan Agen SSM untuk menginstal CloudWatch agen di server, setelah Anda mengedit file konfigurasi CloudWatch agen secara manual, Anda dapat mengunggahnya ke Systems Manager Parameter Store. Untuk melakukan hal itu, gunakan perintah Manajer Sistem `put-parameter`.

Untuk dapat menyimpan file di Parameter Store, Anda harus menggunakan peran IAM dengan izin yang memadai. Untuk informasi selengkapnya, lihat Buat peran IAM dan pengguna untuk digunakan dengan agen CloudWatch .

Gunakan perintah berikut, di mana *Nama parameter* adalah nama yang akan digunakan untuk file ini di Parameter Store dan *configuration_file_pathname* adalah alur dan nama file dari file konfigurasi yang Anda edit.

```
aws ssm put-parameter --name "parameter name" --type "String" --value
 file://configuration_file_pathname
```

Aktifkan Sinyal CloudWatch Aplikasi

Gunakan Sinyal CloudWatch Aplikasi untuk secara otomatis mengarahkan aplikasi Anda AWS sehingga Anda dapat melacak kinerja aplikasi terhadap tujuan bisnis Anda. Sinyal Aplikasi memberi Anda tampilan terpadu, aplikasi-sentris dari aplikasi Java Anda, dependensinya, dan tepinya. Untuk informasi selengkapnya, lihat Sinyal Aplikasi.

CloudWatch Sinyal Aplikasi memanfaatkan CloudWatch agen untuk menerima metrik dan jejak dari aplikasi instrumen otomatis Anda, secara opsional menerapkan aturan untuk mengurangi kardinalitas tinggi, dan kemudian mempublikasikan telemetri yang diproses. CloudWatch Anda dapat memberikan konfigurasi khusus kepada CloudWatch agen khusus untuk Sinyal Aplikasi menggunakan file konfigurasi agen. Untuk memulainya, keberadaan bagian di bawah `app_signals` bagian dalam `metrics_collected` bagian file konfigurasi agen menentukan bahwa CloudWatch agen akan menerima metrik dari aplikasi instrumentasi otomatis Anda. `logs` Demikian pula, keberadaan `app_signals` bagian di bawah `traces_collected` bagian dalam `traces` bagian file konfigurasi agen menentukan bahwa CloudWatch agen diaktifkan untuk menerima jejak dari aplikasi instrumentasi otomatis Anda. Selain itu, Anda dapat secara opsional meneruskan aturan konfigurasi khusus untuk mengurangi penerbitan telemetri kardinalitas tinggi seperti yang diuraikan dalam bagian ini.

- Untuk kluster Amazon EKS, saat Anda menginstal add-on Amazon CloudWatch Observability EKS, CloudWatch agen secara default diaktifkan untuk menerima metrik dan jejak dari aplikasi instrumentasi otomatis Anda. Jika ingin meneruskan aturan konfigurasi khusus secara opsional, Anda dapat melakukannya dengan meneruskan konfigurasi agen khusus ke add-on Amazon EKS saat Anda membuat atau memperbaruinya dengan menggunakan konfigurasi tambahan, seperti yang diuraikan dalam. (Opsional) Konfigurasi tambahan
- Untuk platform lain yang didukung termasuk Amazon EC2, Anda harus memulai CloudWatch agen dengan konfigurasi agen yang mengaktifkan Sinyal Aplikasi dengan menentukan `app_signals` bagian dan secara opsional aturan konfigurasi kustom seperti yang diuraikan nanti di bagian ini.

Berikut ini adalah ikhtisar bidang dalam file konfigurasi CloudWatch agen yang terkait dengan Sinyal CloudWatch Aplikasi.

- `logs`
  - `metrics_collected`— Bidang ini dapat berisi bagian untuk menentukan bahwa agen akan mengumpulkan log untuk mengaktifkan kasus penggunaan seperti Sinyal CloudWatch Aplikasi dan Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS.

    > **Note**
    >
    > Sebelumnya bagian ini juga sudah digunakan untuk menentukan bahwa agen akan mengumpulkan log yang berada dalam format metrik tersemat. Pengaturan tersebut tidak lagi dibutuhkan.

    - `app_signals`(Opsional) Menentukan bahwa Anda ingin mengaktifkan Sinyal CloudWatch Aplikasi untuk menerima metrik dari aplikasi yang diinstrumentasi otomatis untuk memfasilitasi Sinyal Aplikasi. CloudWatch
      - `rules`(Opsional) Sebuah array dari aturan-aturan untuk memilih metrik-metrik dan jejak secara kondisional dan menerapkan tindakan untuk menangani skenario kardinalitas tinggi. Setiap aturan dapat berisi kolom berikut:
        - `rule_name` (Opsional) Nama aturan.
        - `selectors` (Opsional) Sebuah array metrik dan melacak pencocokan dimensi. Setiap pemilih harus memberikan kolom berikut:
          - `dimension` Diperlukan jika `selectors` tidak kosong. Ini menentukan dimensi metrik dan jejak untuk digunakan sebagai filter.
          - `match` Diperlukan `selectors` jika tidak kosong. Pola wildcard yang digunakan untuk mencocokkan nilai dimensi yang ditentukan.
        - `action`(Opsional) Tindakan yang akan diterapkan pada metrik dan jejak yang cocok dengan pemilih yang ditentukan. Nilai `action` harus berupa salah satu kata kunci berikut:
          - `keep`Menentukan untuk mengirim hanya metrik dan jejak untuk CloudWatch jika cocok dengan. `selectors`
          - `drop` Menentukan untuk menjatuhkan metrik dan jejak yang cocok dengan `selectors`.

      - `replace` Menentukan untuk mengganti dimensi metrik dan jejak yang cocok. `selectors` Mereka diganti sesuai dengan `replacements` bagian.
    - `replacements` Diperlukan jika `action` adalah `replace`. Array pasangan dimensi dan nilai yang akan diterapkan pada metrik dan jejak yang cocok dengan yang ditentukan `selectors` saat `action` ada`replace`. Setiap pengganti harus memberikan kolom berikut:
      - `target_dimension` Diperlukan `replacements` jika tidak kosong. Menentukan dimensi yang perlu diganti.
      - `value` Diperlukan `replacements` jika tidak kosong. Nilai untuk mengganti nilai asli `target_dimension` dengan.

- `traces`
  - `traces_collected`
    - `app_signals` Opsional. Tentukan ini untuk memungkinkan CloudWatch agen menerima jejak dari aplikasi instrumentasi otomatis Anda untuk memfasilitasi CloudWatch Sinyal Aplikasi.

> **Note**
>
> Meskipun `app_signals` aturan kustom ditentukan di bagian `metrics_collected` yang terkandung dalam bagian `logs`, mereka juga secara implisit berlaku untuk bagian `traces_collected` tersebut juga. Seperangkat aturan yang sama akan berlaku untuk metrik dan jejak.

Ketika ada beberapa aturan dengan tindakan yang berbeda, mereka berlaku dalam urutan berikut: `keep`, kemudian `drop`, lalu `replace`.

Berikut ini adalah contoh file konfigurasi CloudWatch agen lengkap yang menerapkan aturan khusus.

```
{
  "logs": {
    "metrics_collected": {
      "app_signals": {
        "rules": [
          {
            "rule_name": "keep01",
            "selectors": [
```

```
            {
              "dimension": "Service",
              "match": "pet-clinic-frontend"
            },
            {
              "dimension": "RemoteService",
              "match": "customers-service"
            }
          ],
          "action": "keep"
        },
        {
          "rule_name": "drop01",
          "selectors": [
            {
              "dimension": "Operation",
              "match": "GET /api/customer/owners/*"
            }
          ],
          "action": "drop"
        },
        {
          "rule_name": "replace01",
          "selectors": [
            {
              "dimension": "Operation",
              "match": "PUT /api/customer/owners/*/pets/*"
            },
            {
              "dimension": "RemoteOperation",
              "match": "PUT /owners"
            }
          ],
          "replacements": [
            {
              "target_dimension": "Operation",
              "value": "PUT /api/customer/owners/{ownerId}/pets{petId}"
            }
          ],
          "action": "replace"
        }
      ]
    }
  }
```

```
  },
  "traces": {
    "traces_collected": {
      "app_signals": {}
    }
  }
}
```

Untuk contoh file konfigurasi sebelumnya, `rules` diproses sebagai berikut:

1. Aturan `keep01` memastikan bahwa setiap metrik dan jejak dengan dimensi `Service` sebagai `pet-clinic-frontend` dan dimensi `RemoteService` sebagai `customers-service` akan disimpan.
2. Untuk metrik dan jejak yang diproses setelah diterapkan `keep01`, aturan `drop01` memastikan bahwa metrik-metrik dan jejak dengan dimensi `Operation` sebagai `GET /api/customer/owners/*` akan dibuang.
3. Untuk metrik-metrik dan jejak yang diproses setelah penerapan `drop01`, aturan `replace01` akan memperbarui metrik dan jejak yang memiliki dimensi `Operation` sebagai `PUT /api/customer/owners/*/pets/*` dan dimensi `RemoteOperation` sebagai `PUT /owners` sedemikian rupa sehingga dimensi `Operation` yan dimilikinya sekarang diganti menjadi `PUT /api/customer/owners/{ownerId}/pets{petId}`.

Mengumpulkan metrik performa jaringan

Instans EC2 yang berjalan di Linux yang menggunakan Adaptor Jaringan Elastis (ENA) menerbitkan metrik-metrik performa jaringan. Versi 1.246396.0 dan CloudWatch agen yang lebih baru memungkinkan Anda mengimpor metrik kinerja jaringan ini ke dalam. CloudWatch Saat Anda mengimpor metrik kinerja jaringan ini ke dalam CloudWatch, metrik tersebut akan dikenakan biaya sebagai metrik CloudWatch khusus.

Untuk informasi selengkapnya tentang driver ENA, silakan lihat Mengaktifkan jaringan yang ditingkatkan dengan Adaptor Jaringan Elastis (ENA) pada instans Linux dan Mengaktifkan jaringan yang ditingkatkan dengan Adaptor Jaringan Elastis (ENA) pada instans Windows.

Cara Anda mengatur koleksi metrik performa jaringan berbeda di server Linux dan server Windows.

Tabel berikut mencantumkan metrik performa jaringan yang diaktifkan oleh adaptor ENA. Ketika CloudWatch agen mengimpor metrik ini ke CloudWatch dari instance Linux, itu ditambahkan `ethtool_` di awal masing-masing nama metrik ini.

Metrik	Deskripsi
Nama pada server Linux: **bw_in_allowance_exceeded** Nama pada server Windows: **Aggregate inbound BW allowance exceeded**	Jumlah paket yang antre dan/atau dibuang karena bandwidth agregat masuk melebihi batasan maksimal untuk instans. Metrik ini dikumpulkan hanya jika Anda telah mencantumkannya di `ethtool metrics_collected` sub-bagian dari file konfigurasi CloudWatch agen. Untuk informasi selengkapnya, silakan lihat Mengumpulkan metrik performa jaringan Satuan: Tidak ada
Nama pada server Linux: **bw_out_allowance_exceeded** Nama pada server Windows: **Aggregate outbound BW allowance exceeded**	Jumlah paket yang diantrekan dan/atau dijatuhkan karena bandwidth agregat keluar melebihi batas maksimum untuk instans. Metrik ini dikumpulkan hanya jika Anda telah mencantumkannya di `ethtool metrics_collected` sub-bagian dari file konfigurasi CloudWatch agen. Untuk informasi selengkapnya, silakan lihat Mengumpulkan metrik performa jaringan Satuan: Tidak ada
Nama pada server Linux: **conntrack_allowance_available** Nama pada server Windows: **Available connection tracking allowance**	Melaporkan jumlah koneksi yang dilacak yang dapat dibuat oleh instans sebelum menekan tunjangan Connections Tracked dari tipe instans tersebut. Metrik ini hanya tersedia pada instans EC2 berbasis Nitro menggunakan driver Linux untuk Elastic Network Adapter (ENA) mulai dari versi 2.8.1, dan pada komputer yang menggunakan driver Windows untuk Elastic Network Adapter (ENA) mulai dari versi 2.6.0. Metrik ini dikumpulkan hanya jika Anda telah mencantumkannya di `ethtool metrics_c`

Metrik	Deskripsi
	`ollected` sub-bagian dari file konfigurasi CloudWatch agen. Untuk informasi selengkapnya, silakan lihat Mengumpulkan metrik performa jaringan Satuan: Tidak ada
Nama pada server Linux: **`ena_srd_mode`** Nama pada server Windows: **`ena srd mode`**	Menjelaskan fitur ENA Express mana yang diaktifkan. Untuk informasi selengkapnya tentang ENA Express, lihat Meningkatkan kinerja jaringan dengan ENA Express pada instans Linux Nilai adalah sebagai berikut: • `0` = ENA Ekspres mati, UDP mati • `1` = ENA Ekspres aktif, UDP mati • `2` = ENA Ekspres mati, UDP aktif **Note** Ini terjadi hanya ketika ENA Express awalnya diaktifkan, dan UDP dikonfigurasi untuk menggunakannya. Nilai sebelumnya dipertahankan untuk lalu lintas UDP. • `3` = ENA Ekspres aktif, UDP aktif

Metrik	Deskripsi
Nama pada server Linux: **ena_srd_eligible_tx_pkts** Nama pada server Windows: **ena srd eligible tx pkts**	Jumlah paket jaringan yang dikirim dalam jangka waktu tertentu yang memenuhi persyaratan AWS Scalable Reliable Datagram (SRD) untuk kelayakan, sebagai berikut: • Baik tipe instans pengiriman maupun penerimaan didukung. • Instans pengiriman dan penerimaan harus memiliki ENA Ekspres yang dikonfigurasi. • Instance pengiriman dan penerimaan harus berada di subnet yang sama. • Jalur jaringan antara instans tidak boleh menyertakan kotak perangkat lunak perantara (middlewa re). ENA Ekspres saat ini tidak mendukung kotak perangkat lunak perantara (middleware).
Nama pada server Linux: **ena_srd_tx_pkts** Nama pada server Windows: **ena srd tx pkts**	Jumlah paket SRD yang ditransmisikan dalam jangka waktu tertentu.
Nama pada server Linux: **ena_srd_rx_pkts** Nama pada server Windows: **ena srd rx pkts**	Jumlah paket SRD yang diterima dalam jangka waktu tertentu.
Nama pada server Linux: **ena_srd_resource_utilization** Nama pada server Windows: **ena srd resource utilization**	Persentase pemanfaatan memori maksimum yang diizinkan untuk koneksi SRD bersamaan yang telah dikonsumsi instans.

Metrik	Deskripsi
Nama pada server Linux: **`linklocal_allowance_exceeded`** Nama pada server Windows: **`Link local packet rate allowance exceeded`**	Jumlah paket yang dibuang karena PPS lalu lintas ke layanan proksi lokal melebihi batasan maksimum untuk antarmuka jaringan. Hal ini berdampak lalu lintas ke layanan DNS, Layanan Metadata Instans, dan Layanan Amazon Time Sync. Metrik ini dikumpulkan hanya jika Anda telah mencantumkannya di `ethtool metrics_collected` sub-bagian dari file konfigurasi CloudWatch agen. Untuk informasi selengkapnya, silakan lihat Mengumpulkan metrik performa jaringan Satuan: Tidak ada
Nama pada server Linux: **`linklocal_allowance_exceeded`** Nama pada server Windows: **`Link local packet rate allowance exceeded`**	Jumlah paket yang dibuang karena PPS lalu lintas ke layanan proksi lokal melebihi batasan maksimum untuk antarmuka jaringan. Hal ini berdampak lalu lintas ke layanan DNS, Layanan Metadata Instans, dan Layanan Amazon Time Sync. Metrik ini dikumpulkan hanya jika Anda telah mencantumkannya di `ethtool metrics_collected` sub-bagian dari file konfigurasi CloudWatch agen. Untuk informasi selengkapnya, silakan lihat Mengumpulkan metrik performa jaringan Satuan: Tidak ada

Metrik	Deskripsi
Nama pada server Linux: **pps_allowance_exceeded** Nama pada server Windows: **PPS allowance exceeded**	Jumlah paket yang diantrekan dan/atau dibuang karena PPS dua arah melebihi batasan maksimum untuk instans. Metrik ini dikumpulkan hanya jika Anda telah mencantumkannya di `ethtool metrics_collected` sub-bagian dari file konfigurasi CloudWatch agen. Untuk informasi selengkapnya, silakan lihat Mengumpulkan metrik performa jaringan Satuan: Tidak ada

Pengaturan Linux

Di server Linux, plugin ethtool memungkinkan Anda untuk mengimpor metrik kinerja jaringan ke dalam. CloudWatch

ethtool adalah utilitas Linux standar yang dapat mengumpulkan statistik tentang perangkat Ethernet di server Linux. Statistik yang dikumpulkan tergantung pada perangkat jaringan dan driver. Contoh statistik ini termasuk `tx_cnt`, `rx_bytes`, `tx_errors`, dan `align_errors`. Saat Anda menggunakan plugin ethtool dengan CloudWatch agen, Anda juga dapat mengimpor statistik ini ke dalam CloudWatch, bersama dengan metrik kinerja jaringan EC2 yang tercantum sebelumnya di bagian ini.

> **Tip**
>
> Untuk menemukan statistik yang tersedia di sistem operasi dan perangkat jaringan kami, gunakan perintah `ethtool -S`.

Saat CloudWatch agen mengimpor metrik ke dalam CloudWatch, ia menambahkan `ethtool_` awalan ke nama semua metrik yang diimpor. Jadi statistik ethtool standar `rx_bytes` dipanggil `ethtool_rx_bytes` CloudWatch, dan metrik kinerja jaringan EC2 `bw_in_allowance_exceeded` dipanggil`ethtool_bw_in_allowance_exceeded`. CloudWatch

Di server Linux, untuk mengimpor metrik ethtool, tambahkan `ethtool` bagian ke `metrics_collected` bagian file konfigurasi CloudWatch agen. Bagian `ethtool` dapat mencakup subbagian berikut:

- interface_include— Termasuk bagian ini menyebabkan agen mengumpulkan metrik dari hanya antarmuka yang memiliki nama yang tercantum dalam bagian ini. Jika Anda mengabaikan bagian ini, metrik dikumpulkan dari semua antarmuka Ethernet yang tidak tercantum di `interface_exclude`.

  Antarmuka ethernet default adalah `eth0`.
- interface_exclude— Jika Anda menyertakan bagian ini, buat daftar antarmuka Ethernet yang metriknya tidak ingin Anda kumpulkan.

  Plugin ethtool selalu mengabaikan antarmuka loopback.
- metrics_include - Bagian ini mencantumkan metrik untuk diimpor. CloudWatch Ini dapat mencakup statistik standar yang dikumpulkan oleh ethtool dan metrik jaringan resolusi tinggi Amazon EC2.

Contoh berikut menampilkan bagian dari file konfigurasi CloudWatch agen. Konfigurasi ini mengumpulkan metrik ethtool standar `rx_packets` dan `tx_packets`, dan metrik performa jaringan Amazon EC2 hanya dari antarmuka `eth1`.

Untuk informasi selengkapnya tentang file konfigurasi CloudWatch agen, lihat Buat atau edit file konfigurasi CloudWatch agen secara manual.

```
"metrics": {
    "append_dimensions": {
      "InstanceId": "${aws:InstanceId}"
    },
    "metrics_collected": {
      "ethtool": {
        "interface_include": [
          "eth1"
        ],
        "metrics_include": [
          "rx_packets",
          "tx_packets",
          "bw_in_allowance_exceeded",
          "bw_out_allowance_exceeded",
          "conntrack_allowance_exceeded",
          "linklocal_allowance_exceeded",
```

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

Pengaturan Windows

Di server Windows, metrik kinerja jaringan tersedia melalui Penghitung Kinerja Windows, tempat CloudWatch agen telah mengumpulkan metrik. Jadi Anda tidak perlu plugin untuk mengumpulkan metrik ini dari server Windows.

Berikut ini adalah file konfigurasi sampel untuk mengumpulkan metrik performa jaringan dari Windows. Untuk informasi selengkapnya tentang mengedit file konfigurasi CloudWatch agen, lihat Buat atau edit file konfigurasi CloudWatch agen secara manual.

```
{
    "metrics": {
        "append_dimensions": {
            "InstanceId": "${aws:InstanceId}"
        },
        "metrics_collected": {
            "ENA Packets Shaping": {
                "measurement": [
                    "Aggregate inbound BW allowance exceeded",
                    "Aggregate outbound BW allowance exceeded",
                    "Connection tracking allowance exceeded",
                    "Link local packet rate allowance exceeded",
                    "PPS allowance exceeded"
                ],
                "metrics_collection_interval": 60,
                "resources": [
                    "*"
                ]
            }
        }
    }
}
```

Melihat metrik performa jaringan

Setelah mengimpor metrik kinerja jaringan ke dalam CloudWatch, Anda dapat melihat metrik ini sebagai grafik deret waktu, dan membuat alarm yang dapat menonton metrik ini dan memberi tahu Anda jika metrik tersebut melanggar ambang batas yang Anda tentukan. Prosedur berikut menunjukkan cara melihat metrik ettool sebagai grafik deret waktu. Untuk informasi selengkapnya tentang menyetel alarm, silakan lihat Menggunakan CloudWatch alarm Amazon.

Karena semua metrik ini adalah penghitung agregat, Anda dapat menggunakan fungsi matematika CloudWatch metrik seperti `RATE(METRICS())` untuk menghitung laju metrik ini dalam grafik atau menggunakannya untuk mengatur alarm. Untuk informasi selengkapnya tentang fungsi matematika metrik, silakan lihat Gunakan matematika metrik.

Untuk melihat metrik kinerja jaringan di konsol CloudWatch

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik.
3. Pilih namespace untuk metrik-metrik yang dikumpulkan oleh agen. Secara default, ini adalah CWAgent, tetapi Anda mungkin telah menentukan namespace yang berbeda dalam file konfigurasi agen. CloudWatch
4. Pilih dimensi metrik (misalnya, Metrik Per-Instans).
5. Tab Semua metrik menampilkan semua metrik dimensi tersebut di namespace. Anda dapat melakukan hal berikut:
   a. Untuk membuat grafik sebuah metrik, pilih kotak centang di sebelah metrik. Untuk memilih semua metrik, pilih kotak centang di baris judul tabel.
   b. Untuk menyortir tabel, gunakan judul kolomnya.
   c. Untuk memfilter berdasarkan sumber daya, pilih ID sumber daya, kemudian pilih Add to search.
   d. Untuk memfilter berdasarkan metrik, pilih nama metrik, lalu pilih Tambahkan ke pencarian.
6. (Opsional) Untuk menambahkan grafik ini ke CloudWatch dasbor, pilih Tindakan, lalu pilih Tambahkan ke dasbor.

Kumpulkan metrik GPU NVIDIA

Anda dapat menggunakan CloudWatch agen untuk mengumpulkan metrik GPU NVIDIA dari server Linux. Untuk mengatur ini, tambahkan `nvidia_gpu` bagian di dalam `metrics_collected` bagian file konfigurasi CloudWatch agen. Untuk informasi selengkapnya, lihat Bagian Linux.

Selain itu, instans harus memiliki driver NVIDIA yang diinstal. Driver NVIDIA sudah diinstal sebelumnya pada beberapa Amazon Machine Image (AMI). Jika tidak, Anda dapat melakukan instalasi driver secara manual. Untuk informasi selengkapnya, silakan lihat Install NVIDIA drivers pada instans Linux.

Metrik berikut dapat dikumpulkan. Semua metrik ini dikumpulkan tanpa CloudWatch `Unit`, tetapi Anda dapat menentukan unit untuk setiap metrik dengan menambahkan parameter ke file konfigurasi CloudWatch agen. Untuk informasi selengkapnya, lihat Bagian Linux.

Metrik	Nama metrik di CloudWatch	Deskripsi
`utilization_gpu`	`nvidia_smi_utilization_gpu`	Persentase waktu selama periode sampel terakhir di mana satu atau beberapa kernal pada GPU sedang berjalan.
`temperature_gpu`	`nvidia_smi_temperature_gpu`	Suhu GPU inti dalam derajat Celcius.
`power_draw`	`nvidia_smi_power_draw`	Daya tarik terukur terakhir untuk seluruh papan, dalam watt.
`utilization_memory`	`nvidia_smi_utilization_memory`	Persentase waktu selama periode sampel terakhir di mana memori global (perangkat) sedang dibaca atau ditulis.
`fan_speed`	`nvidia_smi_fan_speed`	Persentase kecepatan kipas maksimum yang saat ini dimaksudkan untuk dijalankan oleh kipas perangkat.
`memory_total`	`nvidia_smi_memory_total`	Memori total yang dilaporkan, dalam MB.

Metrik	Nama metrik di CloudWatch	Deskripsi
memory_used	nvidia_smi_memory_used	Memori yang digunakan, dalam MB.
memory_free	nvidia_smi_memory_free	Bebas memori, dalam MB.
pcie_link_gen_current	nvidia_smi_pcie_link_gen_current	Generasi tautan saat ini.
pcie_link_width_current	nvidia_smi_pcie_link_width_current	Lebar tautan saat ini.
encoder_stats_session_count	nvidia_smi_encoder_stats_session_count	Jumlah sesi encoder saat ini.
encoder_stats_average_fps	nvidia_smi_encoder_stats_average_fps	Rata-rata bergerak dari frame encode per detik.
encoder_stats_average_latency	nvidia_smi_encoder_stats_average_latency	Rata-rata bergerak latensi encode dalam mikrodetik.
clocks_current_graphics	nvidia_smi_clocks_current_graphics	Frekuensi jam grafis (shader) saat ini.
clocks_current_sm	nvidia_smi_clocks_current_sm	Frekuensi saat ini dari jam Streaming Multiprocessor (SM).

Metrik	Nama metrik di CloudWatch	Deskripsi
`clocks_current_memory`	`nvidia_smi_clocks_current_memory`	Frekuensi jam memori saat ini.
`clocks_current_video`	`nvidia_smi_clocks_current_video`	Frekuensi jam video (encoder plus decoder) saat ini.

Semua metrik ini dikumpulkan dengan dimensi berikut:

Dimensi	Deskripsi
`index`	Pengidentifikasi unik untuk GPU di server ini. Merupakan indeks NVIDIA Management Library (NVML) perangkat.
`name`	Jenis GPU. Sebagai contoh, `NVIDIA Tesla A100`.
`host`	Nama host server.

Kumpulkan Metrik Proses dengan Plugin procstat

Plugin Prokstat memungkinkan Anda mengumpulkan metrik dari proses individu. Ini didukung pada server Linux dan pada server yang menjalankan versi Windows Server yang didukung.

Topik

Mengkonfigurasi CloudWatch agen untuk procstat

Untuk menggunakan plugin procstat, tambahkan `procstat` bagian di `metrics_collected` bagian file konfigurasi CloudWatch agen. Ada tiga cara untuk menentukan proses untuk memantau. Anda hanya dapat menggunakan salah satu metode ini, tetapi Anda dapat menggunakan metode tersebut untuk menentukan satu atau beberapa proses untuk memantau.

- `pid_file`: Memilih proses berdasarkan nama file nomor identifikasi proses (PID) yang mereka buat.
- `exe`: Memilih proses yang memiliki nama proses yang cocok dengan string yang Anda tentukan, menggunakan aturan pencocokan ekspresi reguler. Kecocokan adalah kecocokan "berisi", artinya jika Anda menentukan `agent` sebagai istilah yang cocok, proses dengan nama seperti `cloudwatchagent` cocok dengan istilah. Untuk informasi selengkapnya, silakan lihat Sintaks.
- `pattern`: Memilih proses berdasarkan baris perintah yang digunakan untuk memulai proses. Semua proses dipilih yang memiliki baris perintah yang cocok dengan string yang ditentukan menggunakan aturan pencocokan ekspresi reguler. Seluruh baris perintah dicentang, termasuk parameter dan opsi yang digunakan dengan perintah.

  Kecocokan adalah kecocokan "berisi", artinya jika Anda menentukan `-c` sebagai istilah yang cocok, proses dengan parameter seperti `-config` cocok dengan istilah.
- `drop_original_metrics` – Opsional. Jika Anda menggunakan bidang `aggregation_dimensions` di bagian `metrics` untuk menggulung metrik ke dalam hasil agregat, maka secara default agen mengirimkan metrik-metrik agregat dan metrik asli yang dipisahkan untuk setiap nilai dimensi. Jika Anda tidak ingin metrik asli dikirim CloudWatch, Anda dapat menentukan parameter ini dengan daftar metrik. Metrik yang ditentukan bersama dengan parameter ini tidak memiliki metrik berdasarkan dimensi yang dilaporkan. CloudWatch Sebaliknya,

hanya metrik-metrik agregat saja yang dilaporkan. Hal ini akan mengurangi jumlah metrik yang dikumpulkan oleh agen, dan akan mengurangi biaya Anda.

CloudWatch Agen hanya menggunakan salah satu metode ini, bahkan jika Anda memasukkan lebih dari satu bagian di atas. Jika Anda menentukan lebih dari satu bagian, CloudWatch agen menggunakan `pid_file` bagian jika ada. Jika tidak, aplikasi menggunakan `exe` bagian.

Di server Linux, string yang Anda tentukan di `exe` atau `pattern` yang dievaluasi adalah ekspresi biasa. Pada server yang menjalankan Windows Server, string ini dievaluasi sebagai pencarian WMI. Sebuah contoh akan `pattern: "%apache%"`. Untuk informasi selengkapnya, silakan lihat LIKE Operator.

Metode apa pun yang Anda gunakan, Anda dapat menyertakan `metrics_collection_interval` parameter pilihan, yang menentukan seberapa sering dalam hitungan detik untuk mengumpulkan metrik tersebut. Jika Anda menghilangkan parameter ini, nilai bawaan 60 detik akan digunakan.

Dalam contoh di bagian berikut, `procstat` bagian adalah satu-satunya bagian yang termasuk dalam `metrics_collected` bagian dari file konfigurasi agen. File konfigurasi aktual juga dapat menyertakan bagian lain di `metrics_collected`. Untuk informasi selengkapnya, lihat Buat atau edit file konfigurasi CloudWatch agen secara manual.

Mengonfigurasi dengan Pid_file

Contoh bagian `procstat` berikut memantau proses yang membuat file PID `example1.pid` dan `example2.pid`. Metrik yang berbeda dikumpulkan dari setiap proses. Metrik yang dikumpulkan dari proses yang menciptakan `example2.pid` dikumpulkan setiap 10 detik, dan metrik yang dikumpulkan dari `example1.pid` proses dikumpulkan setiap 60 detik, nilai bawaan.

```
{
    "metrics": {
        "metrics_collected": {
            "procstat": [
                {
                    "pid_file": "/var/run/example1.pid",
                    "measurement": [
                        "cpu_usage",
                        "memory_rss"
                    ]
                },
                {
```

```
                    "pid_file": "/var/run/example2.pid",
                    "measurement": [
                        "read_bytes",
                        "read_count",
                        "write_bytes"
                    ],
                    "metrics_collection_interval": 10
                }
            ]
        }
    }
}
```

Konfigurasi dengan Exe

Contoh bagian `procstat` berikut memantau semua proses dengan nama yang cocok dengan string `agent` atau `plugin`. Metrik yang sama dikumpulkan dari setiap proses.

```
{
    "metrics": {
        "metrics_collected": {
            "procstat": [
                {
                    "exe": "agent",
                    "measurement": [
                        "cpu_time",
                        "cpu_time_system",
                        "cpu_time_user"
                    ]
                },
                {
                    "exe": "plugin",
                    "measurement": [
                        "cpu_time",
                        "cpu_time_system",
                        "cpu_time_user"
                    ]
                }
            ]
        }
    }
}
```

Mengonfigurasi dengan Pola

Contoh bagian `procstat` berikut memantau semua proses dengan baris perintah yang cocok dengan string `config` atau `-c`. Metrik yang sama dikumpulkan dari setiap proses.

```
{
    "metrics": {
        "metrics_collected": {
            "procstat": [
                {
                    "pattern": "config",
                    "measurement": [
                        "rlimit_memory_data_hard",
                        "rlimit_memory_data_soft",
                        "rlimit_memory_stack_hard",
                        "rlimit_memory_stack_soft"
                    ]
                },
                {
                    "pattern": "-c",
                    "measurement": [
                        "rlimit_memory_data_hard",
                        "rlimit_memory_data_soft",
                        "rlimit_memory_stack_hard",
                        "rlimit_memory_stack_soft"
                    ]
                }
            ]
        }
    }
}
```

Metrik yang Dikumpulkan oleh Procstat

Tabel berikut mencantumkan metrik yang dapat Anda kumpulkan dengan `procstat` plugin.

CloudWatch Agen menambahkan `procstat` ke awal nama metrik berikut. Ada sintaks yang berbeda tergantung pada apakah itu dikumpulkan dari server Linux atau server yang menjalankan Server Windows. Misalnya, `cpu_time` metrik muncul sebagai `procstat_cpu_time` saat dikumpulkan dari Linux dan `procstat cpu_time` saat dikumpulkan dari Server Windows.

Nama metrik	Tersedia pada	Deskripsi
`cpu_time`	Linux	Jumlah waktu proses menggunakan CPU. Metrik ini diukur dalam seperseratus detik. Satuan: Hitungan
`cpu_time_guest`	Linux	Jumlah waktu proses dalam mode tamu. Metrik ini diukur dalam seperseratus detik. Tipe: Float Satuan: Tidak ada
`cpu_time_guest_nice`	Linux	Jumlah waktu proses berjalan dalam tamu yang baik. Metrik ini diukur dalam seperseratus detik. Tipe: Float

Nama metrik	Tersedia pada	Deskripsi
		Satuan: Tidak ada
cpu_time_idle	Linux	Jumlah waktu proses dalam mode siaga. Metrik ini diukur dalam seperseratus detik. Tipe: Float Satuan: Tidak ada
cpu_time_iowait	Linux	Jumlah waktu saat proses menunggu selesainya operasi I/O. Metrik ini diukur dalam seperseratus detik. Tipe: Float Satuan: Tidak ada

Nama metrik	Tersedia pada	Deskripsi
`cpu_time_irq`	Linux	Jumlah waktu proses menggangg u layanan. Metrik ini diukur dalam seperseratus detik. Tipe: Float Satuan: Tidak ada
`cpu_time_nice`	Linux	Jumlah waktu proses dalam mode bagus. Metrik ini diukur dalam seperseratus detik. Tipe: Float Satuan: Tidak ada

Nama metrik	Tersedia pada	Deskripsi
cpu_time_soft_irq	Linux	Jumlah waktu proses layanan perangkat lunak menggangg u layanan. Metrik ini diukur dalam seperseratus detik. Tipe: Float Satuan: Tidak ada
cpu_time_steal	Linux	Jumlah waktu yang dihabiskan untuk berjalan di sistem operasi lain saat berjalan di lingkunga n tervirtua lisasi. Metrik ini diukur dalam seperseratus detik. Tipe: Float Satuan: Tidak ada

Nama metrik	Tersedia pada	Deskripsi
`cpu_time_stolen`	Linux, Server Windows	Jumlah waktu proses waktu hilang, yaitu waktu yang dihabiskan dalam sistem operasi lain dalam lingkungan tervirtualisasi. Metrik ini diukur dalam seperseratus detik. Tipe: Float Satuan: Tidak ada
`cpu_time_system`	Linux, Windows Server, macOS	Jumlah waktu proses dalam mode sistem. Metrik ini diukur dalam seperseratus detik. Tipe: Float Satuan: Hitungan

Nama metrik	Tersedia pada	Deskripsi
`cpu_time_user`	Linux, Windows Server, macOS	Jumlah waktu proses dalam mode pengguna. Metrik ini diukur dalam seperseratus detik. Satuan: Hitungan
`cpu_usage`	Linux, Windows Server, macOS	Persentase waktu aktif proses dalam kapasitas berapa pun. Satuan: Persen
`memory_data`	Linux, MacOS	Jumlah memori yang digunakan proses untuk data. Satuan: Byte
`memory_locked`	Linux, MacOS	Jumlah memori yang proses telah dikunci. Satuan: Byte

Nama metrik	Tersedia pada	Deskripsi
memory_rss	Linux, Windows Server, macOS	Jumlah memori riil (penghuni yang diatur) yang digunakan proses. Satuan: Byte
memory_stack	Linux, MacOS	Jumlah memori tumpukan yang digunakan dalam proses. Satuan: Byte
memory_swap	Linux, MacOS	Jumlah memori swap yang digunakan proses. Satuan: Byte
memory_vms	Linux, Windows Server, macOS	Jumlah memori virtual yang digunakan proses. Satuan: Byte

Nama metrik	Tersedia pada	Deskripsi
`num_fds`	Linux	Jumlah deskriptor file yang telah dibuka oleh proses ini. Satuan: Tidak ada
`num_threads`	Linux, Windows, macOS	Jumlah utas dalam proses ini. Satuan: Tidak ada
`pid`	Linux, Windows Server, macOS	Pengident ifikasi proses (ID). Satuan: Tidak ada

Nama metrik	Tersedia pada	Deskripsi
`pid_count`	Linux, Windows Server, macOS	Jumlah ID proses yang terkait dengan proses. Di server Linux dan komputer macOS, nama lengkap metrik ini adalah `procstat_lookup_pid_count` dan di Windows Server adalah `procstat_lookup pid_count` . Satuan: Tidak ada
`read_bytes`	Linux, Server Windows	Jumlah byte yang proses telah membaca dari disk. Satuan: Byte
`write_bytes`	Linux, Server Windows	Jumlah byte yang proses telah ditulis ke disk. Satuan: Byte

Nama metrik	Tersedia pada	Deskripsi
`read_count`	Linux, Server Windows	Jumlah operasi baca disk yang telah dijalankan proses. Satuan: Tidak ada
`rlimit_realtime_priority_hard`	Linux	Batas keras pada prioritas real-time yang dapat ditetapkan untuk proses ini. Satuan: Tidak ada
`rlimit_realtime_priority_soft`	Linux	Batas lunak pada prioritas real-time yang dapat ditetapkan untuk proses ini. Satuan: Tidak ada

Nama metrik	Tersedia pada	Deskripsi
`rlimit_signals_pending_hard`	Linux	Batas keras pada jumlah maksimum sinyal yang dapat antri oleh proses ini. Satuan: Tidak ada
`rlimit_signals_pending_soft`	Linux	Batas lunak pada jumlah maksimum sinyal yang dapat antri dengan proses ini. Satuan: Tidak ada
`rlimit_nice_priority_hard`	Linux	Batas keras pada prioritas bagus maksimum yang dapat diatur oleh proses ini. Satuan: Tidak ada

Nama metrik	Tersedia pada	Deskripsi
`rlimit_nice_priority_soft`	Linux	Batas lunak pada prioritas bagus maksimum yang dapat diatur oleh proses ini. Satuan: Tidak ada
`rlimit_num_fds_hard`	Linux	Batas keras pada jumlah maksimum deskriptor file yang dapat dibuka oleh proses ini. Satuan: Tidak ada
`rlimit_num_fds_soft`	Linux	Batas lunak pada jumlah maksimal deskriptor file yang dapat dibuka oleh proses ini. Satuan: Tidak ada

Nama metrik	Tersedia pada	Deskripsi
write_count	Linux, Server Windows	Jumlah operasi penulisan disk yang telah dijalankan proses. Satuan: Tidak ada
involuntary_context_switches	Linux	Berapa kali proses tersebut berlangsung tanpa disadari oleh konteks. Satuan: Tidak ada
voluntary_context_switches	Linux	Berapa kali proses tersebut dilakukan secara sukarela dengan peralihan konteks. Satuan: Tidak ada

Nama metrik	Tersedia pada	Deskripsi
`realtime_priority`	Linux	Penggunaan saat ini prioritas waktu nyata untuk proses. Satuan: Tidak ada
`nice_priority`	Linux	Penggunaan saat ini prioritas yang baik untuk proses. Satuan: Tidak ada
`signals_pending`	Linux	Jumlah sinyal yang menunggu untuk ditangani oleh proses. Satuan: Tidak ada
`rlimit_cpu_time_hard`	Linux	Batas sumber daya waktu CPU yang keras untuk proses. Satuan: Tidak ada

Nama metrik	Tersedia pada	Deskripsi
`rlimit_cpu_time_soft`	Linux	Batas sumber daya waktu CPU lunak untuk proses. Satuan: Tidak ada
`rlimit_file_locks_hard`	Linux	File keras mengunci batas sumber daya proses. Satuan: Tidak ada
`rlimit_file_locks_soft`	Linux	Berkas lunak mengunci batas sumber daya untuk proses. Satuan: Tidak ada
`rlimit_memory_data_hard`	Linux	Batas sumber daya keras pada proses memori yang digunakan untuk data. Satuan: Byte

Nama metrik	Tersedia pada	Deskripsi
`rlimit_memory_data_soft`	Linux	Batas sumber daya lunak pada proses untuk memori yang digunakan untuk data. Satuan: Byte
`rlimit_memory_locked_hard`	Linux	Batas sumber daya keras pada proses memori terkunci. Satuan: Byte
`rlimit_memory_locked_soft`	Linux	Batas sumber daya lunak pada proses untuk memori terkunci. Satuan: Byte
`rlimit_memory_rss_hard`	Linux	Batas sumber daya yang keras pada proses untuk memori fisik. Satuan: Byte

Nama metrik	Tersedia pada	Deskripsi
rlimit_memory_rss_soft	Linux	Batas sumber daya lunak pada proses untuk memori fisik. Satuan: Byte
rlimit_memory_stack_hard	Linux	Batas sumber daya keras pada susunan proses. Satuan: Byte
rlimit_memory_stack_soft	Linux	Batas sumber daya lunak pada susunan proses. Satuan: Byte
rlimit_memory_vms_hard	Linux	Batas sumber daya yang keras pada proses untuk memori virtual. Satuan: Byte
rlimit_memory_vms_soft	Linux	Batas sumber daya lunak pada proses untuk memori virtual. Unit: Bita

Melihat metrik proses yang diimpor oleh agen CloudWatch

Setelah mengimpor metrik proses ke dalam CloudWatch, Anda dapat melihat metrik ini sebagai grafik deret waktu, dan membuat alarm yang dapat menonton metrik ini dan memberi tahu Anda jika metrik tersebut melanggar ambang batas yang Anda tentukan. Prosedur berikut menunjukkan cara melihat metrik proses sebagai grafik deret waktu. Untuk informasi selengkapnya tentang menyetel alarm, silakan lihat Menggunakan CloudWatch alarm Amazon.

Untuk melihat metrik proses di konsol CloudWatch

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik.
3. Pilih namespace untuk metrik-metrik yang dikumpulkan oleh agen. Secara default, ini adalah CWAgent, tetapi Anda mungkin telah menentukan namespace yang berbeda dalam file konfigurasi agen. CloudWatch
4. Pilih dimensi metrik (misalnya, Metrik Per-Instans).
5. Tab Semua metrik menampilkan semua metrik dimensi tersebut di namespace. Anda dapat melakukan hal berikut:

    a. Untuk membuat grafik sebuah metrik, pilih kotak centang di sebelah metrik. Untuk memilih semua metrik, pilih kotak centang di baris judul tabel.

    b. Untuk menyortir tabel, gunakan judul kolomnya.

    c. Untuk menyaring berdasarkan sumber daya, pilih ID sumber daya, kemudian pilih Tambahkan ke pencarian.

    d. Untuk menyaring berdasarkan metrik, pilih nama metrik, kemudian pilih Tambahkan ke pencarian.
6. (Opsional) Untuk menambahkan grafik ini ke CloudWatch dasbor, pilih Tindakan, Tambahkan ke dasbor.

Ambil metrik kustom dengan StatSD

Anda dapat mengambil metrik kustom tambahan dari aplikasi atau layanan Anda menggunakan CloudWatch agen dengan protokol. `StatsD` StatsD adalah solusi open-source populer yang dapat mengumpulkan metrik dari berbagai macam aplikasi. StatsD sangat berguna untuk melengkapi metrik Anda sendiri. Untuk contoh penggunaan CloudWatch agen dan StatSD bersama-sama, lihat Cara memantau metrik aplikasi kustom Anda dengan lebih baik menggunakan Agen Amazon. CloudWatch

StatsDdidukung pada server Linux dan server yang menjalankan Windows Server. CloudWatch mendukung StatsD format berikut:

```
MetricName:value|type|@sample_rate|#tag1:
  value,tag1...
```

- MetricName – String tanpa titik dua, bar, # karakter, atau @ karakter.
- value – Ini dapat berupa integer atau float.
- type – Tentukan c untuk konter, g untuk pengukur, ms untuk pengatur waktu, h untuk histogram, atau s untuk pengaturan.
- sample_rate – (Opsional) Pengapungan antara 0 dan 1, inklusif. Gunakan hanya untuk counter, histogram, dan metrik timer. Nilai bawaannya adalah 1 (menampilkan 100% dari waktu).
- tags— (Opsional) Daftar tag yang dipisahkan koma. StatsDtag mirip dengan dimensi di CloudWatch. Gunakan titik dua untuk tanda kunci/nilai, seperti env:prod.

Anda dapat menggunakan StatsD klien apa pun yang mengikuti format ini untuk mengirim metrik ke CloudWatch agen. Untuk informasi selengkapnya tentang beberapa StatsD klien yang tersedia, lihat halaman klien StatSD di. GitHub

Untuk mengumpulkan metrik kustom ini, tambahkan "statsd": {} masuk ke metrics_collected bagian dari file konfigurasi agen. Anda dapat menambahkan baris ini secara manual. Jika Anda menggunakan pemandu untuk membuat file konfigurasi, sudah selesai untuk Anda. Untuk informasi selengkapnya, lihat Buat file konfigurasi CloudWatch agen.

StatsD Konfigurasi default berfungsi untuk sebagian besar pengguna. Ada bidang opsional yang dapat Anda tambahkan ke bagian statsd dari file konfigurasi agen sesuai kebutuhan:

- service_address— Alamat layanan yang harus didengarkan CloudWatch agen. Formatnya adalah *ip*:*port*. Jika Anda menghilangkan alamat IP, agen akan mendengarkan semua antarmuka yang tersedia. Hanya format UDP yang didukung, sehingga Anda tidak perlu menentukan prefiks UDP.

  Nilai bawaannya adalah :8125.
- metrics_collection_interval – Seberapa sering dalam hitungan detik StatsD plugin berjalan dan mengumpulkan metrik. Nilai bawaannya adalah 10 detik. Rentangnya adalah 1–172.000.

- `metrics_aggregation_interval`— Seberapa sering dalam hitungan detik CloudWatch menggabungkan metrik menjadi titik data tunggal. Nilai bawaannya adalah 60 detik.

  Misalnya, jika `metrics_collection_interval` 10 dan `metrics_aggregation_interval` 60, CloudWatch mengumpulkan data setiap 10 detik. Setelah setiap menit, enam pembacaan data dari menit itu dikumpulkan menjadi satu titik data, yang dikirim ke CloudWatch.

  Rentangnya adalah 0–172.000. Pengaturan `metrics_aggregation_interval` menjadi 0 menonaktifkan agregasi `StatsD` metrik.
- `allowed_pending_messages` – Jumlah pesan UDP yang diizinkan untuk mengantri. Ketika antrian penuh, server StatSD mulai menjatuhkan paket. Nilai bawaannya adalah 10000.
- `drop_original_metrics` – Opsional. Jika Anda menggunakan bidang `aggregation_dimensions` di bagian `metrics` untuk menggulung metrik ke dalam hasil agregat, maka secara default agen mengirimkan metrik-metrik agregat dan metrik asli yang dipisahkan untuk setiap nilai dimensi. Jika Anda tidak ingin metrik asli dikirim CloudWatch, Anda dapat menentukan parameter ini dengan daftar metrik. Metrik yang ditentukan bersama dengan parameter ini tidak memiliki metrik berdasarkan dimensi yang dilaporkan. CloudWatch Sebaliknya, hanya metrik-metrik agregat saja yang dilaporkan. Hal ini akan mengurangi jumlah metrik yang dikumpulkan oleh agen, dan akan mengurangi biaya Anda.

Berikut ini adalah contoh bagian statsd dari file konfigurasi agen, menggunakan port default dan interval pengumpulan dan agregasi kustom.

```
{
   "metrics":{
      "metrics_collected":{
         "statsd":{
            "service_address":":8125",
            "metrics_collection_interval":60,
            "metrics_aggregation_interval":300
         }
      }
   }
}
```

Melihat metrik StatSD yang diimpor oleh agen CloudWatch

Setelah mengimpor metrik StatSD, Anda dapat melihat metrik CloudWatch ini sebagai grafik deret waktu, dan membuat alarm yang dapat menonton metrik ini dan memberi tahu Anda jika metrik

tersebut melanggar ambang batas yang Anda tentukan. Prosedur berikut menunjukkan cara melihat metrik StatsD sebagai grafik deret waktu. Untuk informasi selengkapnya tentang menyetel alarm, silakan lihat Menggunakan CloudWatch alarm Amazon.

Untuk melihat metrik StatSD di konsol CloudWatch

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik.
3. Pilih namespace untuk metrik-metrik yang dikumpulkan oleh agen. Secara default, ini adalah CWAgent, tetapi Anda mungkin telah menentukan namespace yang berbeda dalam file konfigurasi agen. CloudWatch
4. Pilih dimensi metrik (misalnya, Metrik Per-Instans).
5. Tab Semua metrik menampilkan semua metrik dimensi tersebut di namespace. Anda dapat melakukan hal berikut:
   a. Untuk membuat grafik sebuah metrik, pilih kotak centang di sebelah metrik. Untuk memilih semua metrik, pilih kotak centang di baris judul tabel.
   b. Untuk menyortir tabel, gunakan judul kolomnya.
   c. Untuk menyaring berdasarkan sumber daya, pilih ID sumber daya, kemudian pilih Tambahkan ke pencarian.
   d. Untuk menyaring berdasarkan metrik, pilih nama metrik, kemudian pilih Tambahkan ke pencarian.
6. (Opsional) Untuk menambahkan grafik ini ke CloudWatch dasbor, pilih Tindakan, Tambahkan ke dasbor.

Ambil Metrik Kustom dengan koleksi

Anda dapat mengambil metrik tambahan dari aplikasi atau layanan Anda menggunakan CloudWatch agen dengan protokol collectd, yang hanya didukung pada server Linux. collectd adalah solusi open-source populer dengan plugin yang dapat mengumpulkan statistik sistem untuk berbagai macam aplikasi. Dengan menggabungkan metrik sistem yang sudah dapat dikumpulkan CloudWatch agen dengan metrik tambahan dari collectd, Anda dapat memantau, menganalisis, dan memecahkan masalah sistem dan aplikasi Anda dengan lebih baik. Untuk informasi selengkapnya tentang yang dikumpulkan, silakan lihat collectd - The system statistics collection daemon.

Anda menggunakan perangkat lunak collectd untuk mengirim metrik ke agen. CloudWatch Untuk metrik collectd, CloudWatch agen bertindak sebagai server sementara plugin collectd bertindak sebagai klien.

Perangkat lunak yang dikumpulkan tidak diinstal secara otomatis di setiap server. Pada server yang menjalankan Amazon Linux 2, ikuti langkah berikut untuk melakukan instalasi yang dikumpulkan

```
sudo amazon-linux-extras install collectd
```

Untuk informasi tentang instalasi yang dikumpulkan pada sistem lain, silakan lihat Download page for collectd.

Untuk mengumpulkan metrik kustom ini, tambahkan "dikumpulkan": {} masuk ke metrik_yang dikumpulkan bagian dari file konfigurasi agen. Anda dapat menambahkan baris ini secara manual. Jika Anda menggunakan pemandu untuk membuat file konfigurasi, hal itu dilakukan untuk Anda. Untuk informasi selengkapnya, lihat Buat file konfigurasi CloudWatch agen.

Parameter opsional juga tersedia. Jika Anda menggunakan pengumpulan dan tidak menggunakan `/etc/collectd/auth_file` sebagai mengumpulkan_auth_file, Anda harus mengatur beberapa opsi ini.

- service_address: Alamat layanan yang harus didengarkan CloudWatch agen. Formatnya adalah `"udp://`*`ip`*`:`*`port`*. Bawaannya adalah `udp://127.0.0.1:25826`.
- name_prefix: Awalan untuk dilampirkan ke awal nama setiap metrik yang dikumpulkan. Bawaannya adalah `collectd_`. Panjang maksimalnya adalah 255 karakter.
- collectd_security_level: Mengatur tingkat keamanan untuk komunikasi jaringan. Standarnya adalah lakukan enkripsi.

  lakukan enkripsi menentukan bahwa hanya data terenkripsi yang diterima. menandatangani menyatakan bahwa hanya data yang ditandatangani dan dienkripsi yang diterima.
  tidak ada menentukan bahwa semua data diterima. Jika Anda menentukan nilai untuk mengumpulkan_auth_file, data terenkripsi didekripsi jika memungkinkan.

  Untuk informasi selengkapnya, silakan lihat Pengaturan klien dan Kemungkinan interaksi di Wiki yang dikumpulkan.
- mengumpulkan_auth_file Mengatur file yang digunakan untuk memetakan nama pengguna menjadi kata sandi. Kata sandi ini digunakan untuk memverifikasi tanda tangan dan untuk mendekripsi paket jaringan terenkripsi. Jika diberikan, data yang ditandatangani diverifikasi dan paket yang

dienkripsi akan didekripsi. Jika tidak, data bertanda tangan diterima tanpa memeriksa tanda tangan dan data terenkripsi tidak dapat didekripsi.

Bawaannya adalah `/etc/collectd/auth_file`.

Jika tingkat_keamanan_terkumpul diatur menjadi tidak ada, ini opsional. Jika Anda mengatur tingkat_keamanan_terkumpul untuk `encrypt` atau menandatangani, Anda harus menentukan mengumpulkan_auth_file.

Untuk format file autentikasi, setiap baris adalah nama pengguna yang diikuti dengan titik dua dan sejumlah spasi yang diikuti dengan kata sandi. Sebagai contoh:

```
user1: user1_password

user2: user2_password
```

- collectd_typesdb: Daftar satu atau beberapa file yang berisi deskripsi kumpulan set data. Daftar harus dikelilingi oleh kurung, bahkan jika hanya ada satu entri dalam daftar. Setiap entri dalam daftar harus dikelilingi oleh kutipan ganda. Jika ada beberapa entri, pisahkan dengan koma. Bawaannya pada server Linux adalah `["/usr/share/collectd/types.db"]`. Bawaannya pada komputer macOS bergantung pada versi collectd. Sebagai contoh, `["/usr/local/Cellar/collectd/5.12.0/share/collectd/types.db"]`.

  Untuk informasi selengkapnya, lihat https://www.collectd.org/documentation/manpages/types.db.html.
- metrics_aggregation_interval: Seberapa sering dalam hitungan detik menggabungkan metrik menjadi titik data tunggal. CloudWatch Bawaannya adalah 60 detik. Rentangnya adalah 0 hingga 172.000. Mengaturnya menjadi 0 menonaktifkan agregasi metrik yang dikumpulkan.

Berikut ini adalah contoh bagian yang dikumpulkan dari file konfigurasi agen.

```
{
   "metrics":{
      "metrics_collected":{
         "collectd":{
            "name_prefix":"My_collectd_metrics_",
            "metrics_aggregation_interval":120
         }
      }
   }
```

```
}
```

Melihat metrik collectd yang diimpor oleh agen CloudWatch

Setelah mengimpor metrik collectd ke dalam CloudWatch, Anda dapat melihat metrik ini sebagai grafik deret waktu, dan membuat alarm yang dapat menonton metrik ini dan memberi tahu Anda jika metrik tersebut melanggar ambang batas yang Anda tentukan. Prosedur berikut menunjukkan cara melihat metrik collectd sebagai grafik deret waktu. Untuk informasi selengkapnya tentang menyetel alarm, silakan lihat Menggunakan CloudWatch alarm Amazon.

Untuk melihat metrik collectd di konsol CloudWatch

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik.
3. Pilih namespace untuk metrik-metrik yang dikumpulkan oleh agen. Secara default, ini adalah CWAgent, tetapi Anda mungkin telah menentukan namespace yang berbeda dalam file konfigurasi agen. CloudWatch
4. Pilih dimensi metrik (misalnya, Metrik Per-Instans).
5. Tab Semua metrik menampilkan semua metrik dimensi tersebut di namespace. Anda dapat melakukan hal berikut:
   a. Untuk membuat grafik sebuah metrik, pilih kotak centang di sebelah metrik. Untuk memilih semua metrik, pilih kotak centang di baris judul tabel.
   b. Untuk menyortir tabel, gunakan judul kolomnya.
   c. Untuk menyaring berdasarkan sumber daya, pilih ID sumber daya, kemudian pilih Tambahkan ke pencarian.
   d. Untuk menyaring berdasarkan metrik, pilih nama metrik, kemudian pilih Tambahkan ke pencarian.
6. (Opsional) Untuk menambahkan grafik ini ke CloudWatch dasbor, pilih Tindakan, Tambahkan ke dasbor.

Siapkan dan konfigurasikan koleksi metrik Prometheus pada instans Amazon EC2

Bagian berikut menjelaskan cara menginstal CloudWatch agen dengan pemantauan Prometheus pada instans EC2, dan cara mengonfigurasi agen untuk mengikis target tambahan. Ini juga

menyediakan tutorial untuk menyiapkan contoh beban kerja untuk menggunakan pengujian dengan pemantauan Prometheus.

Untuk informasi tentang sistem operasi yang didukung oleh CloudWatch agen, lihat Kumpulkan metrik, log, dan jejak dengan agen CloudWatch

Persyaratan grup keamanan VPC

Jika Anda menggunakan VPC, persyaratan berikut berlaku.

- Aturan masuknya kelompok keamanan untuk beban kerja Prometheus harus membuka port CloudWatch Prometheus ke agen untuk mengikis metrik Prometheus oleh IP pribadi.
- Aturan keluar dari grup keamanan untuk CloudWatch agen harus memungkinkan agen untuk terhubung ke port CloudWatch beban kerja Prometheus dengan IP pribadi.

Topik

- Langkah 1: Instal CloudWatch agen
- Langkah 2: Melakukan scraping pada sumber Prometheus dan mengimpor metrik
- Contoh: Siapkan contoh beban kerja Java/JMX untuk pengujian metrik Prometheus

Langkah 1: Instal CloudWatch agen

Langkah pertama adalah menginstal CloudWatch agen pada instans EC2. Untuk petunjuk, lihat Instalasi CloudWatch agen.

Langkah 2: Melakukan scraping pada sumber Prometheus dan mengimpor metrik

CloudWatch Agen dengan pemantauan Prometheus membutuhkan dua konfigurasi untuk mengikis metrik Prometheus. Salah satunya adalah untuk konfigurasi Prometheus standar seperti yang didokumentasikan dalam <scrape_config> dalam dokumentasi Prometheus. Yang lainnya adalah untuk konfigurasi CloudWatch agen.

Konfigurasi scraping Prometheus

CloudWatch <scrape_config>Agen mendukung konfigurasi scrape Prometheus standar seperti yang didokumentasikan dalam dokumentasi Prometheus.https://prometheus.io/docs/prometheus/latest/configuration/configuration/#scrape_config Anda dapat mengedit bagian ini untuk memperbarui konfigurasi yang sudah ada dalam file ini, dan menambahkan target scraping Prometheus tambahan. Contoh file konfigurasi berisi baris konfigurasi global berikut:

```
PS C:\ProgramData\Amazon\AmazonCloudWatchAgent> cat prometheus.yaml
global:
  scrape_interval: 1m
  scrape_timeout: 10s
scrape_configs:
- job_name: MY_JOB
  sample_limit: 10000
  file_sd_configs:
    - files: ["C:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\prometheus_sd_1.yaml",
 "C:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\prometheus_sd_2.yaml"]
```

Bagian `global` menentukan parameter yang valid dalam semua konteks konfigurasi. Mereka juga berfungsi sebagai default untuk bagian konfigurasi lainnya. Ini berisi parameter berikut:

- `scrape_interval`— Menentukan seberapa sering mengeruk target.
- `scrape_timeout`— Mendefinisikan berapa lama untuk menunggu sebelum permintaan pengerukan habis.

Bagian `scrape_configs` menentukan satu set target dan parameter yang menentukan bagaimana untuk mengeruknya. Ini berisi parameter berikut:

- `job_name`— Nama pekerjaan ditugaskan untuk metrik terkeruk secara bawaan.
- `sample_limit`— Batas per goresan pada jumlah sampel terkeruk yang akan diterima.
- `file_sd_configs`— Daftar konfigurasi penemuan layanan file. Itu membaca satu set file yang berisi daftar nol atau lebih konfigurasi statis. Bagian `file_sd_configs` berisi parameter `files` yang mendefinisikan pola untuk file dari grup target yang diekstrak.

CloudWatch Agen mendukung jenis konfigurasi penemuan layanan berikut.

`static_config` Memungkinkan menentukan daftar target dan label umum ditetapkan untuk mereka. Ini adalah cara kanonik untuk menentukan target statis dalam konfigurasi mengeruk.

Berikut ini adalah contoh konfigurasi statis untuk mengeruk metrik Prometheus dari host lokal. Metrik juga dapat diambil dari server lain jika port Prometheus terbuka ke server tempat agen berjalan.

```
PS C:\ProgramData\Amazon\AmazonCloudWatchAgent> cat prometheus_sd_1.yaml
- targets:
    - 127.0.0.1:9404
  labels:
```

```
    key1: value1
    key2: value2
```

Contoh ini berisi parameter berikut:

- `targets`— Target terambil oleh konfigurasi statis.
- `labels`— Label ditugaskan untuk semua metrik yang terambil dari target.

`ec2_sd_config` Memungkinkan mengambil target ambilan dari instans Amazon EC2. Berikut ini adalah sampel `ec2_sd_config` untuk mengambil metrik Prometheus dari daftar instans EC2. Port Prometheus dari instance ini harus terbuka ke server tempat agen berjalan. CloudWatch Peran IAM untuk instans EC2 di mana CloudWatch agen berjalan harus menyertakan izin. `ec2:DescribeInstance` Misalnya, Anda dapat melampirkan kebijakan terkelola Amazonec2 ReadOnlyAccess ke instance yang menjalankan agen. CloudWatch

```
PS C:\ProgramData\Amazon\AmazonCloudWatchAgent> cat prometheus.yaml
global:
  scrape_interval: 1m
  scrape_timeout: 10s
scrape_configs:
  - job_name: MY_JOB
    sample_limit: 10000
    ec2_sd_configs:
      - region: us-east-1
        port: 9404
        filters:
          - name: instance-id
            values:
              - i-98765432109876543
              - i-12345678901234567
```

Contoh ini berisi parameter berikut:

- `region`— AWS Wilayah tempat instans EC2 target berada. Jika Anda membiarkan kosong ini, Wilayah dari metadata instans digunakan.
- `port`— Port untuk mengambil metrik.
- `filters`— Filter opsional yang akan digunakan untuk memfilter daftar instans. Contoh ini memfilter berdasarkan ID instans EC2. Untuk kriteria lainnya yang dapat Anda filter, lihat DescribeInstances.

CloudWatch konfigurasi agen untuk Prometheus

File konfigurasi CloudWatch agen mencakup `prometheus` bagian di bawah keduanya `logs` dan`metrics_collected`. Ini termasuk parameter berikut.

- cluster_name— menentukan nama klaster yang akan ditambahkan sebagai label pada peristiwa log. Bidang ini bersifat opsional.
- log_group_name— menentukan nama grup log untuk metrik-metrik Prometheus yang di-scraping.
- prometheus_config_path— menentukan jalur file konfigurasi scraping Prometheus.
- emf_processor— menentukan konfigurasi prosesor format metrik tersemat. Untuk informasi selengkapnya tentang format metrik tersemat, silakan lihat Menyematkan metrik dalam log.

  Bagian `emf_processor` dapat berisi parameter berikut:

  - metric_declaration_dedup— Ini diatur ke betul, fungsi de-duplikasi untuk metrik format metrik tersemat diaktifkan.
  - metric_namespace - Menentukan namespace metrik untuk metrik yang dipancarkan. CloudWatch
  - metric_unit— Menentukan nama metrik: peta unit metrik. Untuk informasi tentang unit metrit yang didukung, lihat. MetricDatum
  - metric_declaration— adalah bagian-bagian yang menentukan larik log dengan format metrik tersemat yang akan dihasilkan. Ada `metric_declaration` bagian untuk setiap sumber Prometheus yang diimpor agen secara default CloudWatch. Masing-masing bagian ini mencakup bidang-bidang berikut:
    - `source_labels` menentukan nilai dari label-label yang diperiksa oleh baris `label_matcher`.
    - `label_matcher` adalah ekspresi reguler yang memeriksa nilai dari label-label yang tercantum dalam `source_labels`. Metrik yang cocok diaktifkan untuk dimasukkan dalam format metrik tertanam yang dikirim ke CloudWatch.
    - `metric_selectors`adalah ekspresi reguler yang menentukan metrik yang akan dikumpulkan dan dikirim ke. CloudWatch
    - `dimensions`adalah daftar label yang akan digunakan sebagai CloudWatch dimensi untuk setiap metrik yang dipilih.

Berikut ini adalah contoh konfigurasi CloudWatch agen untuk Prometheus.

```
{
   "logs":{
      "metrics_collected":{
         "prometheus":{
            "cluster_name":"prometheus-cluster",
            "log_group_name":"Prometheus",
            "prometheus_config_path":"C:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\
\prometheus.yaml",
            "emf_processor":{
               "metric_declaration_dedup":true,
               "metric_namespace":"CWAgent-Prometheus",
               "metric_unit":{
                  "jvm_threads_current": "Count",
                  "jvm_gc_collection_seconds_sum": "Milliseconds"
               },
               "metric_declaration":[
                  {
                     "source_labels":[
                        "job", "key2"
                     ],
                     "label_matcher":"MY_JOB;^value2",
                     "dimensions":[
                        [
                           "key1", "key2"
                        ],
                        [
                           "key2"
                        ]
                     ],
                     "metric_selectors":[
                        "^jvm_threads_current$",
                        "^jvm_gc_collection_seconds_sum$"
                     ]
                  }
               ]
            }
         }
      }
   }
}
```

Contoh sebelumnya mengonfigurasi bagian format metrik yang tersemat untuk dikirim sebagai peristiwa log jika kondisi berikut terpenuhi:

- Nilai label `job` adalah `MY_JOB`
- Nilai label `key2` adalah `value2`
- Metrik Prometheus `jvm_threads_current` dan `jvm_gc_collection_seconds_sum` berisi label `job` dan `key2`.

Peristiwa log yang dikirim mencakup bagian yang disorot berikut ini.

```
{
    "CloudWatchMetrics": [
        {
            "Metrics": [
                {
                    "Unit": "Count",
                    "Name": "jvm_threads_current"
                },
                {
                    "Unit": "Milliseconds",
                    "Name": "jvm_gc_collection_seconds_sum"
                }
            ],
            "Dimensions": [
                [
                    "key1",
                    "key2"
                ],
                [
                    "key2"
                ]
            ],
            "Namespace": "CWAgent-Prometheus"
        }
    ],
    "ClusterName": "prometheus-cluster",
    "InstanceId": "i-0e45bd06f196096c8",
    "Timestamp": "1607966368109",
    "Version": "0",
    "host": "EC2AMAZ-PDDOIUM",
    "instance": "127.0.0.1:9404",
    "jvm_threads_current": 2,
    "jvm_gc_collection_seconds_sum": 0.006000000000000002,
    "prom_metric_type": "gauge",
    ...
```

```
}
```

Contoh: Siapkan contoh beban kerja Java/JMX untuk pengujian metrik Prometheus

JMX Exporter adalah sebuah pengekspor Prometheus resmi yang dapat melakukan scraping dan mengekspos JMX mBeans sebagai metrik-metrik Prometheus. Untuk informasi selengkapnya, silakan lihat prometheus/jmx_exporter.

CloudWatch Agen dapat mengumpulkan metrik Prometheus yang telah ditentukan sebelumnya dari Java Virtual Machine (JVM), Hjava, dan Tomcat (Catalina), dari eksportir JMX pada instans EC2.

Langkah 1: Instal CloudWatch agen

Langkah pertama adalah menginstal CloudWatch agen pada instans EC2. Untuk petunjuk, silakan lihat Instalasi CloudWatch agen.

Langkah 2: Memulai beban kerja Java/JMX

Langkah selanjutnya adalah memulai beban kerja Java/JMX.

Pertama, unduh file jar JMX exporter terbaru dari lokasi berikut: prometheus/jmx_exporter.

Gunakan jar untuk aplikasi sampel Anda

Contoh perintah di bagian berikut menggunakan `SampleJavaApplication-1.0-SNAPSHOT.jar` sebagai file jar. Ganti bagian perintah ini dengan jar untuk aplikasi Anda.

Siapkan konfigurasi JMX exporter

File `config.yaml` adalah file konfigurasi JMX exporter. Untuk informasi selengkapnya, silakan lihat Configuration di dokumentasi JMX exporter.

Berikut adalah contoh konfigurasi untuk Java dan Tomcat.

```
---
lowercaseOutputName: true
lowercaseOutputLabelNames: true

rules:
- pattern: 'java.lang<type=OperatingSystem><>(FreePhysicalMemorySize|
TotalPhysicalMemorySize|FreeSwapSpaceSize|TotalSwapSpaceSize|SystemCpuLoad|
ProcessCpuLoad|OpenFileDescriptorCount|AvailableProcessors)'
  name: java_lang_OperatingSystem_$1
  type: GAUGE
```

```
- pattern: 'java.lang<type=Threading><>(TotalStartedThreadCount|ThreadCount)'
  name: java_lang_threading_$1
  type: GAUGE

- pattern: 'Catalina<type=GlobalRequestProcessor, name=\"(\w+-\w+)-(\d+)\"><>(\w+)'
  name: catalina_globalrequestprocessor_$3_total
  labels:
    port: "$2"
    protocol: "$1"
  help: Catalina global $3
  type: COUNTER

- pattern: 'Catalina<j2eeType=Servlet, WebModule=//([-a-zA-Z0-9+&@#/%?=~_|!:.,;]*[-
a-zA-Z0-9+&@#/%=~_|]), name=([-a-zA-Z0-9+/$%~_-|!.]*), J2EEApplication=none,
 J2EEServer=none><>(requestCount|maxTime|processingTime|errorCount)'
  name: catalina_servlet_$3_total
  labels:
    module: "$1"
    servlet: "$2"
  help: Catalina servlet $3 total
  type: COUNTER

- pattern: 'Catalina<type=ThreadPool, name="(\w+-\w+)-(\d+)"><>(currentThreadCount|
currentThreadsBusy|keepAliveCount|pollerThreadCount|connectionCount)'
  name: catalina_threadpool_$3
  labels:
    port: "$2"
    protocol: "$1"
  help: Catalina threadpool $3
  type: GAUGE

- pattern: 'Catalina<type=Manager, host=([-a-zA-Z0-9+&@#/%?=~_|!:.,;]*[-a-zA-
Z0-9+&@#/%=~_|]), context=([-a-zA-Z0-9+/$%~_-|!.]*)><>(processingTime|sessionCounter|
rejectedSessions|expiredSessions)'
  name: catalina_session_$3_total
  labels:
    context: "$2"
    host: "$1"
  help: Catalina session $3 total
  type: COUNTER

- pattern: ".*"
```

Mulai aplikasi Java dengan pengekspor Prometheus

Mulai aplikasi sampel. Ini akan memancarkan metrik Prometheus ke port 9404. Pastikan untuk mengganti entry point `com.gubupt.sample.app.App` dengan infromasi yang benar untuk aplikasi java sampel Anda.

Di Linux, masukkan perintah berikut.

```
$ nohup java -javaagent:./jmx_prometheus_javaagent-0.14.0.jar=9404:./config.yaml -cp
  ./SampleJavaApplication-1.0-SNAPSHOT.jar com.gubupt.sample.app.App &
```

Di Windows, masukkan perintah berikut.

```
PS C:\> java -javaagent:.\jmx_prometheus_javaagent-0.14.0.jar=9404:.\config.yaml -cp  .
\SampleJavaApplication-1.0-SNAPSHOT.jar com.gubupt.sample.app.App
```

Verifikasi emisi metrik Prometheus

Verifikasi bahwa metrik Prometheus dipancarkan.

Di Linux, masukkan perintah berikut.

```
$ curl localhost:9404
```

Di Windows, masukkan perintah berikut.

```
PS C:\> curl  http://localhost:9404
```

Contoh keluaran di Linux:

```
StatusCode        : 200
StatusDescription : OK
Content           : # HELP jvm_classes_loaded The number of classes that are currently
 loaded in the JVM
                    # TYPE jvm_classes_loaded gauge
                    jvm_classes_loaded 2526.0
                    # HELP jvm_classes_loaded_total The total number of class...
RawContent        : HTTP/1.1 200 OK
                    Content-Length: 71908
                    Content-Type: text/plain; version=0.0.4; charset=utf-8
                    Date: Fri, 18 Dec 2020 16:38:10 GMT
```

```
                    # HELP jvm_classes_loaded The number of classes that are
 currentl...
Forms             : {}
Headers           : {[Content-Length, 71908], [Content-Type, text/plain; version=0.0.4;
 charset=utf-8], [Date, Fri, 18
                    Dec 2020 16:38:10 GMT]}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : System.__ComObject
RawContentLength  : 71908
```

Langkah 3: Konfigurasikan CloudWatch agen untuk mengikis metrik Prometheus

Selanjutnya, atur konfigurasi scrape Prometheus di file konfigurasi agen. CloudWatch

Untuk mengatur konfigurasi scrape Prometheus untuk contoh Java/JMX

1. Mengatur konfigurasi untuk `file_sd_config` dan `static_config`.

   Di Linux, masukkan perintah berikut.

   ```
   $ cat /opt/aws/amazon-cloudwatch-agent/var/prometheus.yaml
   global:
     scrape_interval: 1m
     scrape_timeout: 10s
   scrape_configs:
     - job_name: jmx
       sample_limit: 10000
       file_sd_configs:
         - files: [ "/opt/aws/amazon-cloudwatch-agent/var/prometheus_file_sd.yaml" ]
   ```

   Di Windows, masukkan perintah berikut.

   ```
   PS C:\ProgramData\Amazon\AmazonCloudWatchAgent> cat prometheus.yaml
   global:
     scrape_interval: 1m
     scrape_timeout: 10s
   scrape_configs:
     - job_name: jmx
       sample_limit: 10000
       file_sd_configs:
   ```

```
      - files: [ "C:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\
\prometheus_file_sd.yaml" ]
```

2. Mengatur konfigurasi target ambilan.

   Di Linux, masukkan perintah berikut.

```
$ cat /opt/aws/amazon-cloudwatch-agent/var/prometheus_file_sd.yaml
- targets:
  - 127.0.0.1:9404
  labels:
    application: sample_java_app
    os: linux
```

   Di Windows, masukkan perintah berikut.

```
PS C:\ProgramData\Amazon\AmazonCloudWatchAgent> cat prometheus_file_sd.yaml
- targets:
  - 127.0.0.1:9404
  labels:
    application: sample_java_app
    os: windows
```

3. Atur konfigurasi ambilan Prometheus dengan `ec2_sc_config`. Ganti *`your-ec2-instance-id`* dengan ID instans EC2 yang benar.

   Di Linux, masukkan perintah berikut.

```
$ cat .\prometheus.yaml
global:
  scrape_interval: 1m
  scrape_timeout: 10s
scrape_configs:
  - job_name: jmx
    sample_limit: 10000
    ec2_sd_configs:
      - region: us-east-1
        port: 9404
        filters:
          - name: instance-id
            values:
              - your-ec2-instance-id
```

Di Windows, masukkan perintah berikut.

```
PS C:\ProgramData\Amazon\AmazonCloudWatchAgent> cat prometheus_file_sd.yaml
- targets:
  - 127.0.0.1:9404
  labels:
    application: sample_java_app
    os: windows
```

4. Siapkan konfigurasi CloudWatch agen. Pertama, arahkan ke direktori yang benar. Di Linux, itu adalah `/opt/aws/amazon-cloudwatch-agent/var/cwagent-config.json`. Pada Windows, ini adalah `C:\ProgramData\Amazon\AmazonCloudWatchAgent\cwagent-config.json`.

   Berikut ini adalah konfigurasi sampel dengan metrik Java/JHX Prometheus didefinisikan. Pastikan untuk mengganti *`path-to-Prometheus-Scrape-Configuration-file`* dengan jalur yang benar.

```
{
  "agent": {
    "region": "us-east-1"
  },
  "logs": {
    "metrics_collected": {
      "prometheus": {
        "cluster_name": "my-cluster",
        "log_group_name": "prometheus-test",
        "prometheus_config_path": "path-to-Prometheus-Scrape-Configuration-file",
        "emf_processor": {
          "metric_declaration_dedup": true,
          "metric_namespace": "PrometheusTest",
          "metric_unit":{
            "jvm_threads_current": "Count",
            "jvm_classes_loaded": "Count",
            "java_lang_operatingsystem_freephysicalmemorysize": "Bytes",
            "catalina_manager_activesessions": "Count",
            "jvm_gc_collection_seconds_sum": "Seconds",
            "catalina_globalrequestprocessor_bytesreceived": "Bytes",
            "jvm_memory_bytes_used": "Bytes",
            "jvm_memory_pool_bytes_used": "Bytes"
          },
```

```
          "metric_declaration": [
            {
              "source_labels": ["job"],
              "label_matcher": "^jmx$",
              "dimensions": [["instance"]],
              "metric_selectors": [
                "^jvm_threads_current$",
                "^jvm_classes_loaded$",
                "^java_lang_operatingsystem_freephysicalmemorysize$",
                "^catalina_manager_activesessions$",
                "^jvm_gc_collection_seconds_sum$",
                "^catalina_globalrequestprocessor_bytesreceived$"
              ]
            },
            {
              "source_labels": ["job"],
              "label_matcher": "^jmx$",
              "dimensions": [["area"]],
              "metric_selectors": [
                "^jvm_memory_bytes_used$"
              ]
            },
            {
              "source_labels": ["job"],
              "label_matcher": "^jmx$",
              "dimensions": [["pool"]],
              "metric_selectors": [
                "^jvm_memory_pool_bytes_used$"
              ]
            }
          ]
        }
      }
    },
    "force_flush_interval": 5
  }
}
```

5. Mulai ulang CloudWatch agen dengan memasukkan salah satu perintah berikut.

   Di Linux, masukkan perintah berikut.

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-
config -m ec2 -s -c file:/opt/aws/amazon-cloudwatch-agent/var/cwagent-config.json
```

Di Windows, masukkan perintah berikut.

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1"
 -a fetch-config -m ec2 -s -c file:C:\ProgramData\Amazon\AmazonCloudWatchAgent
\cwagent-config.json
```

Melihat metrik dan log Prometheus

Anda sekarang dapat melihat metrik Java/JMX yang dikumpulkan.

Untuk melihat metrik untuk sampel beban kerja Java/JMX

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di Wilayah tempat klaster Anda berjalan, pilih Metrik pada panel navigasi yang ada sebelah kiri. Temukan PrometheusTestnamespace untuk melihat metrik.
3. Untuk melihat peristiwa CloudWatch Log, pilih Grup log di panel navigasi. Peristiwa-peristiwa tersebut berada di grup log prometheus-test.

Instal CloudWatch agen dengan menggunakan add-on Amazon CloudWatch Observability EKS

Add-on Amazon CloudWatch Observability EKS menginstal CloudWatch Agen dan agen Fluent-bit pada kluster Amazon EKS, dengan observabilitas yang ditingkatkan Container Insights untuk Amazon CloudWatch EKS dan Sinyal Aplikasi diaktifkan secara default. Dengan menggunakan add-on, Anda dapat mengumpulkan metrik infrastruktur, telemetri performa aplikasi, dan log kontainer dari klaster Amazon EKS.

Dengan Wawasan Kontainer yang memiliki kemampuan observabilitas yang ditingkatkan untuk Amazon EKS, metrik-metrik Wawasan Kontainer akan dikenakan biaya per observasi, bukan dibebankan per metrik yang disimpan atau log yang diserap. Untuk Sinyal Aplikasi, penagihan didasarkan pada permintaan masuk ke aplikasi Anda, permintaan keluar dari aplikasi Anda, dan setiap tujuan tingkat layanan yang dikonfigurasi (SLO). Setiap permintaan masuk yang diterima menghasilkan satu sinyal aplikasi, dan setiap permintaan keluar yang dibuat menghasilkan satu

sinyal aplikasi. Setiap SLO menciptakan dua sinyal aplikasi per periode pengukuran. Untuk informasi selengkapnya tentang CloudWatch harga, lihat CloudWatch Harga Amazon.

Add-on Amazon CloudWatch Observability EKS didukung pada cluster Amazon EKS yang berjalan dengan Kubernetes versi 1.23 atau yang lebih baru.

Saat Anda menginstal add-on, Anda juga harus memberikan izin IAM untuk memungkinkan CloudWatch agen mengirim metrik, log, dan jejak ke. CloudWatch Ada dua cara untuk melakukan hal ini:

- Lampirkan sebuah kebijakan ke peran IAM simpul pekerja Anda. Opsi ini memberikan izin ke node pekerja untuk mengirim telemetri ke. CloudWatch
- Gunakan peran IAM untuk akun layanan untuk pod agen, dan lampirkan kebijakan pada peran ini. Ini hanya berfungsi untuk klaster Amazon EKS. Opsi ini hanya memberikan CloudWatch akses ke pod agen yang sesuai.

Opsi 1: Instal dengan izin IAM pada simpul pekerja

Untuk menggunakan metode ini, pertama-tama lampirkan kebijakan CloudWatchAgentServerPolicyIAM ke node pekerja Anda dengan memasukkan perintah berikut. Dalam perintah ini, ganti *my-worker-node-role*dengan peran IAM yang digunakan oleh node pekerja Kubernetes Anda.

```
aws iam attach-role-policy \
--role-name my-worker-node-role \
--policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy
```

Kemudian instal add-on Amazon CloudWatch Observability EKS. Untuk menginstal add-on, Anda dapat menggunakan, konsol AWS CLI, AWS CloudFormation, atau Terraform.

AWS CLI

Untuk menggunakan AWS CLI untuk menginstal add-on Amazon CloudWatch Observability EKS

Masukkan perintah berikut. Ganti *my-cluster-name* dengan nama klaster Anda.

```
aws eks create-addon --addon-name amazon-cloudwatch-observability --cluster-name my-cluster-name
```

Amazon EKS console

Untuk menggunakan konsol Amazon EKS untuk menambahkan add-on Amazon CloudWatch Observability EKS

1. Buka konsol Amazon EKS di https://console.aws.amazon.com/eks/home#/clusters.
2. Pada panel navigasi sebelah kiri, pilih Klaster.
3. Pilih nama cluster yang ingin Anda konfigurasikan add-on Amazon CloudWatch Observability EKS.
4. Pilih tab Add-ons.
5. Pilih Get more add-ons
6. Pada halaman Pilih add-on, lakukan hal berikut:
   a. Di bagian Amazon EKS-Addons, pilih kotak centang Amazon CloudWatch Observability.
   b. Pilih Berikutnya.
7. Pada halaman Configure selected add-ons settings, lakukan hal berikut:
   a. Pilih Versi yang ingin Anda gunakan.
   b. Untuk Memilih peran IAM, pilih Dapatkan dari simpul
   c. (Opsional) Anda dapat memperluas Pengaturan konfigurasi opsional. Jika Anda memilih Override untuk metode resolusi Konflik, satu atau beberapa pengaturan untuk add-on yang sudah ada dapat ditimpa dengan pengaturan add-on Amazon EKS. Jika Anda tidak mengaktifkan opsi ini dan ada konflik dengan pengaturan yang ada, operasi gagal. Anda dapat menggunakan pesan kesalahan yang dihasilkan untuk memecahkan masalah konflik. Sebelum memilih opsi ini, pastikan add-on Amazon EKS tidak mengelola pengaturan yang perlu Anda kelola sendiri.
   d. Pilih Berikutnya.
8. Pada halaman Review dan add, pilih Create. Setelah penginstalan add-on selesai, Anda melihat add-on yang diinstal.

AWS CloudFormation

Untuk digunakan AWS CloudFormation untuk menginstal add-on Amazon CloudWatch Observability EKS

Ganti *my-cluster-name* dengan nama klaster Anda. Untuk informasi lebih lanjut, lihat AWS::EKS::Addon.

```
{
    "Resources": {
        "EKSAddOn": {
            "Type": "AWS::EKS::Addon",
            "Properties": {
                "AddonName": "amazon-cloudwatch-observability",
                "ClusterName": "my-cluster-name"
            }
        }
    }
}
```

Terraform

Untuk menggunakan Terraform untuk menginstal add-on Amazon CloudWatch Observability EKS

Ganti *my-cluster-name* dengan nama klaster Anda. Untuk informasi selengkapnya, silakan lihat Sumber daya: aws_eks_addon.

```
resource "aws_eks_addon" "example" {
  addon_name   = "amazon-cloudwatch-observability"
  cluster_name = "my-cluster-name"
}
```

Opsi 2: Melakukan instalasi menggunakan peran akun layanan IAM

Sebelum menggunakan metode ini, verifikasi prasyarat berikut:

- Anda memiliki sebuah klaster Amazon EKS fungsional dengan simpul yang terlampir di salah satu Wilayah AWS yang mendukung Wawasan Kontainer. Untuk melihat daftar Wilayah yang didukung, silakan lihat Wawasan Kontainer.
- Anda telah `kubectl` melakukan instalasi dan mengkonfigurasi untuk klaster. Untuk informasi selengkapnya, silakan lihat Menginstal `kubectl` dalam Panduan Pengguna Amazon EKS.
- Anda telah `eksctl` melakukan instalasi. Untuk informasi selengkapnya, silakan lihat Menginstal atau memperbarui `eksctl` di Panduan Pengguna Amazon EKS.

Untuk menginstal add-on Amazon CloudWatch Observability EKS menggunakan peran akun layanan IAM

1. Masukkan perintah berikut untuk membuat penyedia OpenID Connect (OIDC), jika klaster belum memilikinya. Untuk informasi selengkapnya, silakan lihat Mengonfigurasi akun layanan Kubernetes untuk mengambil peran IAM dalam Panduan Pengguna Amazon EKS.

```
eksctl utils associate-iam-oidc-provider --cluster my-cluster-name --approve
```

2. Masukkan perintah berikut untuk membuat peran IAM dengan CloudWatchAgentServerPolicykebijakan terlampir, dan konfigurasikan akun layanan agen untuk mengambil peran tersebut menggunakan OIDC. Ganti *my-cluster-name*dengan nama cluster Anda, dan ganti *my-service-account-role*dengan nama peran yang ingin Anda kaitkan dengan akun layanan. Jika peran tersebut belum ada, `eksctl` buatlah untuk Anda.

```
eksctl create iamserviceaccount \
  --name cloudwatch-agent \
  --namespace amazon-cloudwatch --cluster my-cluster-name \
  --role-name my-service-account-role \
  --attach-policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy \
  --role-only \
  --approve
```

3. Instal add-on berikut dengan memasukan perintah berikut. Ganti *my-cluster-name*dengan nama cluster Anda, ganti *111122223333* dengan ID akun Anda, dan ganti *my-service-account-role*dengan peran IAM yang dibuat pada langkah sebelumnya.

```
aws eks create-addon --addon-name amazon-cloudwatch-observability --cluster-
name my-cluster-name --service-account-role-arn arn:aws:iam::111122223333:role/my-
service-account-role
```

(Opsional) Konfigurasi tambahan

Menyisih dari mengumpulkan log kontainer

Secara default, add-on menggunakan Fluent Bit untuk mengumpulkan log kontainer dari semua pod dan kemudian mengirimkan log ke CloudWatch Log. Untuk informasi tentang log mana yang dikumpulkan, silakan lihat Menyiapkan Fluent Bit.

Untuk memilih keluar dari koleksi log kontainer, berikan opsi berikut saat Anda membuat atau memperbarui add-on:

```
--configuration-values '{ "containerLogs": { "enabled": false } }'
```

Menyisih dari koleksi metrik GPU NVIDIA

Dimulai dengan versi 1.300034.0 CloudWatch agen, Container Insights mengumpulkan metrik GPU NVIDIA dari beban kerja EKS secara default. Metrik ini tercantum dalam tabel diMetrik GPU NVIDIA.

Anda dapat memilih untuk tidak mengumpulkan metrik GPU NVIDIA dengan menyetel `accelerated_compute_metrics` opsi di file konfigurasi CloudWatch agen. `false` Opsi ini ada di `kubernetes` bagian `metrics_collected` bagian dalam file CloudWatch konfigurasi. Berikut ini adalah contoh konfigurasi opt-out.

```
{
  "agent": {
    "region": "us-east-1"
  },
  "logs": {
    "metrics_collected": {
      "emf": {
      },
      "kubernetes": {
        "enhanced_container_insights": true,
        "accelerated_compute_metrics": false
      }
    },
    "force_flush_interval": 5,
  }
}
```

Gunakan konfigurasi CloudWatch agen kustom

Untuk mengumpulkan metrik, log, atau jejak lain menggunakan CloudWatch agen, Anda dapat menentukan konfigurasi khusus sambil juga mengaktifkan Wawasan Kontainer dan Sinyal CloudWatch Aplikasi. Untuk melakukannya, sematkan file konfigurasi CloudWatch agen di dalam kunci konfigurasi di bawah kunci agen dari konfigurasi lanjutan yang dapat Anda gunakan saat membuat atau memperbarui add-on EKS. Berikut ini merupakan konfigurasi agen default ketika Anda tidak memberikan konfigurasi tambahan apa pun.

> **Important**
>
> Konfigurasi kustom apa pun yang Anda berikan menggunakan pengaturan konfigurasi tambahan akan menggantikan konfigurasi default yang digunakan oleh agen. Berhati-hatilah untuk tidak menonaktifkan fungsionalitas yang diaktifkan secara default secara tidak sengaja, seperti Wawasan Kontainer dengan peningkatan observabilitas dan Sinyal Aplikasi. CloudWatch Dalam skenario bahwa Anda diminta untuk menyediakan konfigurasi agen kustom, kami sarankan menggunakan konfigurasi default berikut sebagai dasar dan kemudian memodifikasinya sesuai dengan itu.

```
--configuration-values '{
  "agent": {
    "config": {
      "logs": {
        "metrics_collected": {
          "app_signals": {},
          "kubernetes": {
            "enhanced_container_insights": true
          }
        }
      },
      "traces": {
        "traces_collected": {
          "app_signals": {}
        }
      }
    }

}'
```

Kelola sertifikat TLS webhook penerimaan

Add-on Amazon CloudWatch Observability EKS memanfaatkan webhook masuk Kubernetes untuk memvalidasi dan mengubah permintaan sumber daya `Instrumentation` kustom (CR), dan secara opsional permintaan pod `AmazonCloudWatchAgent` Kubernetes di klaster jika Application Signals diaktifkan. CloudWatch Di Kubernetes, webhook memerlukan sertifikat TLS yang dikonfigurasi oleh server API untuk dipercaya untuk memastikan komunikasi yang aman.

Secara default, add-on Amazon CloudWatch Observability EKS secara otomatis menghasilkan CA yang ditandatangani sendiri dan sertifikat TLS yang ditandatangani oleh CA ini untuk mengamankan komunikasi antara server API dan server webhook. Sertifikat yang dibuat secara otomatis ini memiliki kedaluwarsa default 10 tahun dan tidak diperpanjang secara otomatis setelah kedaluwarsa. Selain itu, bundel CA dan sertifikat dibuat ulang setiap kali add-on ditingkatkan atau diinstal ulang, sehingga mengatur ulang kedaluwarsa. Jika ingin mengubah kedaluwarsa default sertifikat yang dibuat secara otomatis, Anda dapat menggunakan konfigurasi tambahan berikut saat membuat atau memperbarui add-on. Ganti *expiry-in-days*dengan durasi kedaluwarsa yang Anda inginkan dalam beberapa hari.

```
--configuration-values '{ "admissionWebhooks": { "autoGenerateCert":
 { "expiryDays": expiry-in-days } } }'
```

Untuk solusi otoritas sertifikat yang lebih aman dan kaya fitur, add-on ini memiliki dukungan opt-in untuk cert-manager, solusi yang diadopsi secara luas untuk manajemen sertifikat TLS di Kubernetes yang menyederhanakan proses memperoleh, memperbarui, mengelola, dan menggunakan sertifikat tersebut. Ini akan memastikan bahwa sertifikat sudah valid dan terbaru, dan upaya untuk memperbarui sertifikat pada waktu yang dikonfigurasi sebelum kedaluwarsa. cert-manager juga memfasilitasi penerbitan sertifikat dari berbagai sumber yang didukung, termasuk AWS Certificate Manager Private Certificate Authority.

Kami menyarankan Anda meninjau praktik terbaik untuk pengelolaan sertifikat TLS di klaster Anda dan menyarankan Anda untuk ikut serta dalam pengelola sertifikat untuk lingkungan produksi. Perhatikan bahwa jika Anda memilih untuk mengaktifkan cert-manager untuk mengelola sertifikat TLS webhook penerimaan, Anda harus melakukan pra-instal cert-manager di klaster Amazon EKS sebelum menginstal add-on Amazon Observability EKS. CloudWatch Lihat cert-manager documentation untuk mempelajari lebih lanjut tentang opsi penginstalan yang tersedia. Setelah melakukan instalasinya, Anda dapat memilih untuk menggunakan cert-manager untuk mengelola sertifikat TLS webhook penerimaan menggunakan konfigurasi tambahan berikut saat membuat atau memperbarui add-on.

```
--configuration-values '{ "admissionWebhooks": { "certManager": { "enabled":
 true } } }'
```

Konfigurasi lanjutan yang dibahas di bagian ini secara default akan menggunakan SelfSignedpenerbit.

Mengumpulkan ID volume Amazon EBS

Jika ingin mengumpulkan ID volume Amazon EBS di log performa, Anda harus menambahkan kebijakan lain ke peran IAM yang terlampir ke simpul pekerja atau akun layanan. Tambahkan hal berikut sebagai kebijakan selaras. Untuk informasi selengkapnya, silakan lihat Menambahkan dan Menghapus Izin Identitas IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ec2:DescribeVolumes"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

Memecahkan masalah add-on Amazon CloudWatch Observability EKS

Gunakan informasi berikut untuk membantu memecahkan masalah dengan add-on Amazon CloudWatch Observability EKS.

Memperbarui dan menghapus add-on Amazon CloudWatch Observability EKS

Untuk petunjuk tentang memperbarui atau menghapus add-on Amazon CloudWatch Observability EKS, lihat Mengelola add-on Amazon EKS. Gunakan `amazon-cloudwatch-observability` sebagai nama add-on.

Verifikasi versi CloudWatch agen yang digunakan oleh add-on Amazon CloudWatch Observability EKS

Add-on Amazon CloudWatch Observability EKS menginstal jenis sumber daya khusus `AmazonCloudWatchAgent` yang mengontrol perilaku daemonset CloudWatch agen di cluster, termasuk versi agen yang digunakan. CloudWatch Anda bisa mendapatkan daftar semua sumber daya `AmazonCloudWatchAgent` kustom yang diinstal pada klaster Anda dengan memasukkan perintah berikut:

```
kubectl get amazoncloudwatchagent -A
```

Dalam output perintah ini, Anda harus dapat memeriksa versi CloudWatch agen. Atau, Anda juga dapat mendeskripsikan sumber daya `amazoncloudwatchagent` atau salah satu pod `cloudwatch-agent-*` yang berjalan di klaster Anda untuk memeriksa citra yang digunakan.

Menangani a ConfigurationConflict saat mengelola add-on

Saat Anda menginstal atau memperbarui add-on Amazon CloudWatch Observability EKS, jika Anda melihat kegagalan yang `Health Issue` disebabkan oleh tipe `ConfigurationConflict` dengan deskripsi yang dimulai`Conflicts found when trying to apply. Will not continue due to resolve conflicts mode`, kemungkinan karena Anda sudah memiliki CloudWatch agen dan komponen terkait seperti ServiceAccount, ClusterRole dan yang ClusterRoleBinding diinstal pada cluster. Ketika add-on mencoba menginstal CloudWatch agen dan komponen terkait, jika mendeteksi perubahan apa pun dalam konten, secara default gagal instalasi atau pembaruan untuk menghindari penimpaan status sumber daya pada cluster.

Jika Anda mencoba untuk onboard ke add-on Amazon CloudWatch Observability EKS dan Anda melihat kegagalan ini, kami sarankan untuk menghapus penyiapan CloudWatch agen yang ada yang sebelumnya Anda instal di cluster dan kemudian menginstal add-on EKS. Pastikan untuk mencadangkan penyesuaian apa pun yang mungkin telah Anda buat ke penyiapan CloudWatch agen asli seperti konfigurasi agen khusus, dan berikan ini ke add-on EKS servabilitas Amazon CloudWatch Ob saat Anda menginstal atau memperbaruinya berikutnya. Jika sebelumnya Anda telah menginstal CloudWatch agen untuk orientasi ke Container Insights, lihat Menghapus CloudWatch agen dan Fluentd untuk Wawasan Kontainer untuk informasi selengkapnya.

Atau, add-on mendukung opsi konfigurasi resolusi konflik yang memiliki kemampuan untuk menentukan `OVERWRITE`. Anda dapat menggunakan opsi ini untuk melanjutkan dengan melakukan instalasi atau memperbarui add-on dengan menimpa konflik di klaster. Jika Anda menggunakan konsol Amazon EKS, Anda akan menemukan Metode penyelesaian konflik saat Anda memilih Pengaturan konfigurasi opsional ketika Anda membuat atau memperbarui add-on. Jika Anda menggunakan AWS CLI, Anda dapat memberikan perintah Anda `--resolve-conflicts OVERWRITE` untuk membuat atau memperbarui add-on.

Metrik yang dikumpulkan oleh agen CloudWatch

Anda dapat mengumpulkan metrik dari server dengan menginstal CloudWatch agen di server. Anda dapat menginstal agen di instans Amazon EC2 dan server on-premise, dan di komputer yang menjalankan Linux, Windows Server, atau macOS. Jika Anda melakukan instalasi agen di instans

Amazon EC2, metrik yang dikumpulkan adalah sebagai tambahan untuk metrik yang diaktifkan secara bawaan di instans Amazon EC2.

Untuk informasi tentang menginstal CloudWatch agen pada sebuah instance, lihatKumpulkan metrik, log, dan jejak dengan agen CloudWatch .

Semua metrik yang dibahas di bagian ini dikumpulkan langsung oleh CloudWatch agen.

Metrik yang dikumpulkan oleh CloudWatch agen pada instans Windows Server

Di server yang menjalankan Windows Server, menginstal CloudWatch agen memungkinkan Anda mengumpulkan metrik yang terkait dengan penghitung di Windows Performance Monitor. Nama CloudWatch metrik untuk penghitung ini dibuat dengan menempatkan spasi antara nama objek dan nama penghitung. Misalnya, `% Interrupt Time` penghitung `Processor` objek diberi nama metrik `Processor % Interrupt Time` di CloudWatch. Untuk informasi selengkapnya tentang penghitung Windows Performance Monitor, silakan lihat dokumentasi Microsoft Windows Server.

Namespace default untuk metrik yang dikumpulkan oleh CloudWatch agen adalah`CWAgent`, meskipun Anda dapat menentukan namespace yang berbeda saat mengonfigurasi agen.

Metrik yang dikumpulkan oleh CloudWatch agen di instans Linux dan macOS

Tabel berikut mencantumkan metrik yang dapat Anda kumpulkan dengan CloudWatch agen di server Linux dan komputer macOS.

Metrik	Deskripsi
`cpu_time_active`	Jumlah waktu saat CPU aktif dalam kapasitas apa pun. Metrik ini diukur dalam seperseratus detik. Satuan: Tidak ada
`cpu_time_guest`	Jumlah waktu CPU menjalankan CPU virtual untuk sistem operasi tamu. Metrik ini diukur dalam seperseratus detik. Satuan: Tidak ada

Metrik	Deskripsi
cpu_time_guest_nice	Lama waktu CPU menjalankan CPU virtual untuk sistem operasi tamu, yang memiliki prioritas rendah dan dapat terganggu oleh proses lain. Metrik ini diukur dalam seperseratus detik. Satuan: Tidak ada
cpu_time_idle	Lama waktu CPU diam. Metrik ini diukur dalam seperseratus detik. Satuan: Tidak ada
cpu_time_iowait	Jumlah waktu yang ditunggu CPU untuk menyelesa ikan operasi I/O. Metrik ini diukur dalam seperseratus detik. Satuan: Tidak ada
cpu_time_irq	Lamanya waktu CPU mengganggu layanan. Metrik ini diukur dalam seperseratus detik. Satuan: Tidak ada
cpu_time_nice	Lama waktu CPU berada dalam mode pengguna dengan proses prioritas rendah, yang dapat dengan mudah diinterupsi oleh proses prioritas tinggi. Metrik ini diukur dalam seperseratus detik. Satuan: Tidak ada
cpu_time_softirq	Lama waktu CPU melakukan layanan software interrupt. Metrik ini diukur dalam seperseratus detik. Satuan: Tidak ada

Metrik	Deskripsi
`cpu_time_steal`	Jumlah waktu CPU waktu tercuri, yaitu waktu yang dihabiskan dalam sistem operasi lain dalam lingkungan tervirtualisasi. Metrik ini diukur dalam seperseratus detik. Satuan: Tidak ada
`cpu_time_system`	Jumlah waktu CPU berada dalam mode sistem. Metrik ini diukur dalam seperseratus detik. Satuan: Tidak ada
`cpu_time_user`	Jumlah waktu CPU berada dalam mode pengguna. Metrik ini diukur dalam seperseratus detik. Satuan: Tidak ada
`cpu_usage_active`	Persentase waktu saat CPU aktif dalam kapasitas apa pun. Satuan: Persen
`cpu_usage_guest`	Persentase waktu saat CPU menjalankan CPU virtual untuk sistem operasi tamu. Satuan: Persen
`cpu_usage_guest_nice`	Persentase waktu saat CPU menjalankan CPU virtual untuk sistem operasi tamu, yang memiliki prioritas rendah dan dapat terganggu oleh proses lain. Satuan: Persen
`cpu_usage_idle`	Persentase waktu saat CPU tidak digunakan. Satuan: Persen

Metrik	Deskripsi
cpu_usage_iowait	Persentase waktu saat CPU menunggu selesainya operasi I/O. Satuan: Persen
cpu_usage_irq	Persentase waktu dari CPU adalah gangguan layanan. Satuan: Persen
cpu_usage_nice	Persentase waktu saat CPU berada dalam mode pengguna dengan proses prioritas rendah, yang dapat dengan mudah mengganggu proses prioritas. Satuan: Persen
cpu_usage_softirq	Persentase waktu di mana CPU melayani gangguan perangkat lunak. Satuan: Persen
cpu_usage_steal	Persentase waktu CPU waktu hilang, atau waktu yang dihabiskan dalam sistem operasi lain dalam lingkungan tervirtualisasi. Satuan: Persen
cpu_usage_system	Persentase waktu CPU berada dalam mode sistem. Satuan: Persen
cpu_usage_user	Persentase waktu CPU berada dalam mode pengguna. Satuan: Persen
disk_free	Ruang kosong di disk. Satuan: Byte

Metrik	Deskripsi
`disk_inodes_free`	Jumlah node indeks yang tersedia pada disk. Satuan: Hitungan
`disk_inodes_total`	Total jumlah simpul indeks yang dicadangkan pada disk. Satuan: Hitungan
`disk_inodes_used`	Jumlah node indeks yang digunakan pada disk. Satuan: Hitungan
`disk_total`	Total ruang di disk, termasuk digunakan dan gratis. Satuan: Byte
`disk_used`	Menggunakan ruang di disk. Satuan: Byte
`disk_used_percent`	Persentase total ruang disk yang digunakan. Satuan: Persen
`diskio_iops_in_progress`	Jumlah permintaan I/O yang telah dikeluarkan kepada pengemudi perangkat tetapi belum diselesai kan. Satuan: Hitungan
`diskio_io_time`	Jumlah waktu saat disk meminta I/O mengantre. Satuan: Milidetik Satu-satunya statistik yang harus digunakan untuk metrik ini adalah `Sum`. Jangan gunakan `Average`.

Metrik	Deskripsi
diskio_reads	Jumlah operasi baca disk. Satuan: Hitungan Satu-satunya statistik yang harus digunakan untuk metrik ini adalah Sum. Jangan gunakan Average.
diskio_read_bytes	Jumlah byte yang dibaca dari disk. Satuan: Byte Satu-satunya statistik yang harus digunakan untuk metrik ini adalah Sum. Jangan gunakan Average.
diskio_read_time	Lamanya waktu untuk membaca permintaan menunggu di disk. Beberapa permintaan baca menunggu pada saat yang sama meningkatkan angka. Sebagai contoh, jika 5 permintaan semuanya menunggu rata-rata 100 milidetik, 500 akan dilaporkan. Satuan: Milidetik Satu-satunya statistik yang harus digunakan untuk metrik ini adalah Sum. Jangan gunakan Average.
diskio_writes	Angka operasi penulisan disk. Satuan: Hitungan Satu-satunya statistik yang harus digunakan untuk metrik ini adalah Sum. Jangan gunakan Average.
diskio_write_bytes	Jumlah byte yang ditulis ke disk. Satuan: Byte Satu-satunya statistik yang harus digunakan untuk metrik ini adalah Sum. Jangan gunakan Average.

Metrik	Deskripsi
`diskio_write_time`	Lamanya waktu permintaan penulisan menunggu di disk. Beberapa permintaan tertulis menunggu secara bersamaan meningkatkan jumlah. Sebagai contoh, jika 8 permintaan semuanya menunggu rata-rata 1000 milidetik, 8000 dilaporkan. Satuan: Milidetik Satu-satunya statistik yang harus digunakan untuk metrik ini adalah `Sum`. Jangan gunakan `Average`.
`ethtool_bw_in_allowance_exceeded`	Jumlah paket yang antre dan/atau dibuang karena bandwidth agregat masuk melebihi batasan maksimal untuk instans. Metrik ini dikumpulkan hanya jika Anda telah mencantumkannya di `ethtool metrics_collected` sub-bagian dari file konfigurasi CloudWatch agen. Untuk informasi selengkapnya, silakan lihat Mengumpulkan metrik performa jaringan Satuan: Tidak ada
`ethtool_bw_out_allowance_exceeded`	Jumlah paket yang antri dan/atau jatuh karena bandwidth agregat keluar melebihi maksimum untuk instans. Metrik ini dikumpulkan hanya jika Anda telah mencantumkannya di `ethtool metrics_collected` sub-bagian dari file konfigurasi CloudWatch agen. Untuk informasi selengkapnya, silakan lihat Mengumpulkan metrik performa jaringan Satuan: Tidak ada

Metrik	Deskripsi
`ethtool_conntrack_allowance_exceeded`	Jumlah paket yang dibuang karena pelacakan koneksi melebihi batasan maksimal untuk instans dan koneksi baru tidak dapat dibuat. Hal ini dapat mengakibatkan hilangnya paket untuk lalu lintas ke atau dari instans. Metrik ini dikumpulkan hanya jika Anda telah mencantumkannya di `ethtool metrics_collected` sub-bagian dari file konfigurasi CloudWatch agen. Untuk informasi selengkapnya, silakan lihat Mengumpulkan metrik performa jaringan Satuan: Tidak ada
`ethtool_linklocal_allowance_exceeded`	Jumlah paket yang dibuang karena PPS lalu lintas ke layanan proksi lokal melebihi batasan maksimum untuk antarmuka jaringan. Hal ini berdampak lalu lintas ke layanan DNS, Layanan Metadata Instans, dan Layanan Amazon Time Sync. Metrik ini dikumpulkan hanya jika Anda telah mencantumkannya di `ethtool metrics_collected` sub-bagian dari file konfigurasi CloudWatch agen. Untuk informasi selengkapnya, silakan lihat Mengumpulkan metrik performa jaringan Satuan: Tidak ada

Metrik	Deskripsi
`ethtool_pps_allowance_exceeded`	Jumlah paket yang diantrekan dan/atau dibuang karena PPS dua arah melebihi batasan maksimum untuk instans. Metrik ini dikumpulkan hanya jika Anda telah mencantumkannya di `ethtool metrics_collected` sub-bagian dari file konfigurasi CloudWatch agen. Untuk informasi selengkapnya, lihat Mengumpulkan metrik performa jaringan. Satuan: Tidak ada
`mem_active`	Jumlah memori yang telah digunakan selama periode sampel terakhir. Satuan: Byte
`mem_available`	Jumlah memori yang tersedia dan dapat diberikan secara instan ke proses. Satuan: Byte
`mem_available_percent`	Persentase memori yang tersedia dan dapat diberikan secara langsung ke proses. Satuan: Persen
`mem_buffered`	Jumlah memori yang digunakan untuk penyangga. Satuan: Byte
`mem_cached`	Jumlah memori yang digunakan untuk cache file. Satuan: Byte
`mem_free`	Jumlah memori yang tidak digunakan. Satuan: Byte

Metrik	Deskripsi
`mem_inactive`	Jumlah memori yang belum digunakan selama periode sampel terakhir Satuan: Byte
`mem_total`	Total jumlah memori. Satuan: Byte
`mem_used`	Jumlah memori yang saat ini digunakan. Satuan: Byte
`mem_used_percent`	Persentase memori yang saat ini digunakan. Satuan: Persen
`net_bytes_recv`	Jumlah byte yang diterima oleh antarmuka jaringan. Satuan: Byte Satu-satunya statistik yang harus digunakan untuk metrik ini adalah `Sum`. Jangan gunakan `Average`.
`net_bytes_sent`	Jumlah byte yang dikirim oleh antarmuka jaringan. Satuan: Byte Satu-satunya statistik yang harus digunakan untuk metrik ini adalah `Sum`. Jangan gunakan `Average`.
`net_drop_in`	Jumlah paket yang diterima oleh antarmuka jaringan ini yang jatuh. Satuan: Hitungan Satu-satunya statistik yang harus digunakan untuk metrik ini adalah `Sum`. Jangan gunakan `Average`.

Metrik	Deskripsi
net_drop_out	Jumlah paket yang ditransmisikan oleh antarmuka jaringan ini yang jatuh. Satuan: Hitungan Satu-satunya statistik yang harus digunakan untuk metrik ini adalah Sum. Jangan gunakan Average.
net_err_in	Jumlah kesalahan penerimaan yang terdeteksi oleh antarmuka jaringan ini. Satuan: Hitungan Satu-satunya statistik yang harus digunakan untuk metrik ini adalah Sum. Jangan gunakan Average.
net_err_out	Jumlah kesalahan transmisi yang terdeteksi oleh antarmuka jaringan ini. Satuan: Hitungan Satu-satunya statistik yang harus digunakan untuk metrik ini adalah Sum. Jangan gunakan Average.
net_packets_sent	Jumlah paket yang dikirim oleh antarmuka jaringan ini. Satuan: Hitungan Satu-satunya statistik yang harus digunakan untuk metrik ini adalah Sum. Jangan gunakan Average.
net_packets_recv	Jumlah paket yang diterima oleh antarmuka jaringan ini. Satuan: Hitungan Satu-satunya statistik yang harus digunakan untuk metrik ini adalah Sum. Jangan gunakan Average.

Metrik	Deskripsi
`netstat_tcp_close`	Jumlah koneksi TCP tanpa status. Satuan: Hitungan
`netstat_tcp_close_wait`	Jumlah koneksi TCP yang menunggu permintaan pengakhiran dari klien. Satuan: Hitungan
`netstat_tcp_closing`	Jumlah koneksi TCP yang menunggu permintaan pengakhiran dengan pengakuan dari klien. Satuan: Hitungan
`netstat_tcp_established`	Jumlah koneksi TCP yang dibangun. Satuan: Hitungan
`netstat_tcp_fin_wait1`	Jumlah koneksi TCP di `FIN_WAIT1` selama proses menutup koneksi. Satuan: Hitungan
`netstat_tcp_fin_wait2`	Jumlah koneksi TCP di `FIN_WAIT2` selama proses menutup koneksi. Satuan: Hitungan
`netstat_tcp_last_ack`	Jumlah koneksi TCP yang menunggu klien untuk mengirimkan pengakuan atas pesan pengakhiran koneksi. Ini adalah status terakhir kanan sebelum koneksi ditutup. Satuan: Hitungan
`netstat_tcp_listen`	Jumlah port TCP yang saat ini mendengarkan untuk permintaan koneksi. Satuan: Hitungan

Metrik	Deskripsi
netstat_tcp_none	Jumlah koneksi TCP dengan klien tidak aktif. Satuan: Hitungan
netstat_tcp_syn_sent	Jumlah koneksi TCP yang menunggu permintaan koneksi yang sesuai setelah mengirimkan permintaa n koneksi. Satuan: Hitungan
netstat_tcp_syn_recv	Jumlah koneksi TCP yang menunggu penerimaan permintaan koneksi setelah mengirim dan menerima permintaan koneksi. Satuan: Hitungan
netstat_tcp_time_wait	Jumlah koneksi TCP yang saat ini menunggu untuk memastikan bahwa klien menerima pengakuan atas permintaan pengakhiran koneksinya. Satuan: Hitungan
netstat_udp_socket	Jumlah koneksi UDP saat ini. Satuan: Hitungan
processes_blocked	Jumlah proses yang diblokir. Satuan: Hitungan
processes_dead	Jumlah proses yang mati, ditandai dengan X kode negara bagian di Linux. Metrik ini tidak dikumpulkan di komputer macOS. Satuan: Hitungan

Metrik	Deskripsi
processes_idle	Jumlah proses yang tidak digunakan (menyapu lebih dari 20 detik). Hanya tersedia pada instans FreeBSD. Satuan: Hitungan
processes_paging	Jumlah proses yang dalam satuan, ditunjukkan oleh W kode negara bagian di Linux. Metrik ini tidak dikumpulkan di komputer macOS. Satuan: Hitungan
processes_running	Jumlah proses yang berjalan, ditandai dengan R kode negara bagian. Satuan: Hitungan
processes_sleeping	Jumlah proses yang tidur, ditunjukkan oleh S kode negara bagian. Satuan: Hitungan
processes_stopped	Jumlah proses yang dihentikan, ditandai dengan T kode negara bagian. Satuan: Hitungan
processes_total	Total jumlah proses dalam instans. Satuan: Hitungan
processes_total_threads	Total jumlah utas yang membentuk proses. Metrik ini hanya tersedia di instans Linux. Metrik ini tidak dikumpulkan di komputer macOS. Satuan: Hitungan

Metrik	Deskripsi
processes_wait	Jumlah proses yang melakukan paging, ditunjukkan oleh kode status W pada instans FreeBSD. Metrik ini hanya tersedia pada instans FreeBSD, dan tidak tersedia pada instans Linux, Windows Server, atau macOS. Satuan: Hitungan
processes_zombies	Jumlah proses zombie, yang ditunjukkan oleh Z kode negara bagian. Satuan: Hitungan
swap_free	Jumlah ruang swap yang tidak digunakan. Satuan: Byte
swap_used	Jumlah ruang swap yang saat ini digunakan. Satuan: Byte
swap_used_percent	Persentase ruang swap yang saat ini digunakan. Unit: Persen

Definisi metrik memori yang dikumpulkan oleh agen CloudWatch

Ketika CloudWatch agen mengumpulkan metrik memori, sumbernya adalah subsistem manajemen memori host. Sebagai contoh, kernel Linux mengekspos data yang dikelola OS di /proc. Untuk memori, datanya ada di /proc/meminfo.

Setiap sistem operasi dan arsitektur yang berbeda memiliki perhitungan yang berbeda dari sumber daya yang digunakan oleh proses. Untuk informasi selengkapnya, silakan lihat bagian-bagian berikut ini.

Selama setiap interval pengumpulan, CloudWatch agen pada setiap instance mengumpulkan sumber daya instance dan menghitung sumber daya yang digunakan oleh semua proses yang berjalan dalam instance itu. Informasi ini dilaporkan kembali ke CloudWatch metrik. Anda dapat

mengonfigurasi panjang interval pengumpulan dalam file konfigurasi CloudWatch agen. Untuk informasi selengkapnya, lihat CloudWatch file konfigurasi agen: Bagian Agen.

Daftar berikut menjelaskan bagaimana metrik memori yang dikumpulkan CloudWatch agen didefinisikan.

- Memori Aktif – Memori yang digunakan oleh suatu proses. Dengan kata lain, memori yang digunakan oleh aplikasi yang sedang berjalan saat ini.
- Memori yang tersedia – Memori yang dapat langsung diberikan ke proses tanpa sistem masuk ke swap (juga dikenal sebagai memori virtual).
- Buffer Memory – Area data yang dibagi oleh perangkat keras atau proses program yang beroperasi pada kecepatan dan prioritas yang berbeda.
- Memori Cached – Menyimpan instruksi program dan data yang digunakan berulang kali dalam pengoperasian program yang mungkin dibutuhkan CPU selanjutnya.
- Memori Bebas – Memori yang tidak digunakan sama sekali dan sudah tersedia. Ini benar-benar gratis untuk sistem yang akan digunakan saat dibutuhkan.
- Memori Tidak Aktif – Halaman yang belum diakses "baru–baru ini".
- Total Memory – Ukuran RAM memori fisik yang sebenarnya.
- Memori Bekas – Memori yang saat ini digunakan oleh program dan proses.

Topik

Linux: Metrik dikumpulkan dan perhitungan yang digunakan

Metrik yang dikumpulkan dan unit:

- Aktif (Byte)
- Tersedia (Byte)
- Persen Tersedia (Persen)
- Buffered (Byte)

- Cache (Byte)
- Gratis (Byte)
- Tidak Aktif (Byte)
- Jumlah (Byte)
- Digunakan (Byte)
- Persen yang Digunakan (Persen)

Memori yang digunakan = Total Memori - Memori Bebas - Memori cache - Memori penyangga

Total memori = Memori Bekas+Memori Bebas+Memori cache +Memori penyangga

macOS: Metrik dikumpulkan dan perhitungan yang digunakan

Metrik yang dikumpulkan dan unit:

- Aktif (Byte)
- Tersedia (Byte)
- Persen Tersedia (Persen)
- Gratis (Byte)
- Tidak Aktif (Byte)
- Jumlah (Byte)
- Digunakan (Byte)
- Persen yang Digunakan (Persen)

Memori yang tersedia = Memori Gratis + Memori tidak aktif

Memori yang digunakan = Total Memori - Memori yang tersedia

Total memori = Memori yang tersedia + Memori yang Digunakan

Windows: Metrik dikumpulkan

Metrik yang dikumpulkan pada host Windows tercantum di bawah ini. Semua metrik ini memiliki `None` untuk `Unit`.

- Byte yang tersedia
- Kesalahan cache/detik

- Kesalahan halaman/detik
- Halaman/detik

Tidak ada perhitungan yang digunakan untuk metrik Windows karena CloudWatch agen mem-parsing peristiwa dari penghitung kinerja.

Contoh: Menghitung metrik memori di Linux

Sebagai contoh, anggaplah memasukkan perintah **`cat /proc/meminfo`** pada host Linux menunjukkan hasil berikut:

```
MemTotal:       3824388 kB
MemFree:         462704 kB
MemAvailable:   2157328 kB
Buffers:         126268 kB
Cached:         1560520 kB
SReclaimable:    289080 kB>
```

Dalam contoh ini, CloudWatch agen akan mengumpulkan nilai-nilai berikut. Semua nilai yang dikumpulkan dan dilaporkan CloudWatch agen dalam byte.

- `mem_total`: 3916173312 bita
- `mem_available`: 2209103872 byte (+ Cache) MemFree
- `mem_free`: 473808896 bita
- `mem_cached`: 1893990400 byte (`cached` + `SReclaimable`
- `mem_used`: 1419075584 + byte (`MemTotal` – (`MemFree` + `Buffers` + `Cached` + `SReclaimable`)))
- `mem_buffered`: 129667072 bita
- `mem_available_percent`: 56,41%
- `mem_used_percent`: 36,24% (`mem_used` / `mem_total`) * 100

Skenario umum dengan CloudWatch agen

Bagian berikut menguraikan cara menyelesaikan tugas konfigurasi dan penyesuaian umum untuk CloudWatch agen.

Topik

Menjalankan CloudWatch agen sebagai pengguna yang berbeda

Pada server Linux, CloudWatch berjalan sebagai pengguna root secara default. Untuk menjalankan agen sebagai pengguna yang berbeda, gunakan `run_as_user` parameter di `agent` bagian dalam file konfigurasi CloudWatch agen. Opsi ini hanya tersedia di server Linux.

Jika Anda sudah menjalankan agen dengan pengguna root dan ingin mengubah dengan menggunakan pengguna lain, Anda bisa menggunakan salah satu prosedur berikut.

Untuk menjalankan CloudWatch agen sebagai pengguna yang berbeda pada instans EC2 yang menjalankan Linux

1. Unduh dan instal paket CloudWatch agen baru. Untuk informasi selengkapnya, lihat Unduh paket CloudWatch agen.
2. Buat pengguna Linux baru atau gunakan pengguna default yang disebut `cwagent` bahwa file RPM atau DEB yang dibuat.
3. Berikan kredensial untuk pengguna ini dengan salah satu cara berikut:
   - Jika file `.aws/credentials` ada di direktori home pengguna root, Anda harus membuat file kredensyal untuk pengguna yang akan Anda gunakan untuk menjalankan agen. CloudWatch File kredensials ini akan `/home/`*`username`*`/.aws/credentials`. Kemudian atur nilai parameter `shared_credential_file` di `common-config.toml` ke nama jalur file kredensial. Untuk informasi selengkapnya, lihat (Opsional) Ubah Konfigurasi Umum untuk Informasi Proxy atau Wilayah.
   - Jika file `.aws/credentials` tidak ada di direktori home dari pengguna root, Anda dapat melakukan salah satu dari yang berikut ini:

- Buat file kredensyal untuk pengguna yang akan Anda gunakan untuk menjalankan agen. CloudWatch File kredensials ini akan `/home/username/.aws/credentials`. Kemudian atur nilai parameter `shared_credential_file` di `common-config.toml` ke nama jalur file kredensial. Untuk informasi selengkapnya, lihat (Opsional) Ubah Konfigurasi Umum untuk Informasi Proxy atau Wilayah.
- Alih-alih membuat file kredensial, lampirkan peran IAM ke instans. Agen menggunakan peran ini sebagai penyedia kredensial.

4. Dalam file konfigurasi CloudWatch agen, tambahkan baris berikut di `agent` bagian:

```
"run_as_user": "username"
```

   Buat modifikasi lain ke file konfigurasi sesuai kebutuhan. Untuk informasi selengkapnya, silakan lihat Buat file konfigurasi CloudWatch agen

5. Berikan kepada pengguna tersebut izin-izin yang diperlukan. Pengguna harus memiliki izin Read (r) untuk mengumpulkan file log, dan harus memiliki izin Execute (x) di setiap direktori di jalur file log.
6. Mulai agen dengan file konfigurasi yang baru saja Anda modifikasi.

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-
config -m ec2 -s -c file:configuration-file-path
```

Untuk menjalankan CloudWatch agen sebagai pengguna lain di server lokal yang menjalankan Linux

1. Unduh dan instal paket CloudWatch agen baru. Untuk informasi selengkapnya, lihat Unduh paket CloudWatch agen.
2. Buat pengguna Linux baru atau gunakan pengguna default yang disebut `cwagent` bahwa file RPM atau DEB yang dibuat.
3. Menyimpan kredensial pengguna ini ke jalur yang dapat diakses pengguna, seperti `/home/username/.aws/credentials`.
4. Tetapkan nilai dari `shared_credential_file` parameter dalam `common-config.toml` ke nama jalur file kredensial. Untuk informasi selengkapnya, lihat (Opsional) Ubah Konfigurasi Umum untuk Informasi Proxy atau Wilayah.
5. Dalam file konfigurasi CloudWatch agen, tambahkan baris berikut di `agent` bagian:

```
"run_as_user": "username"
```

Buat modifikasi lain ke file konfigurasi sesuai kebutuhan. Untuk informasi selengkapnya, silakan lihat Buat file konfigurasi CloudWatch agen

6. Berikan izin-izin yang diperlukan kepada pengguna. Pengguna harus memiliki izin Read (r) untuk mengumpulkan file log, dan harus memiliki izin Execute (x) di setiap direktori di jalur file log.
7. Mulai agen dengan file konfigurasi yang baru saja Anda modifikasi.

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-
config -m ec2 -s -c file:configuration-file-path
```

Bagaimana CloudWatch agen menangani file log yang jarang

File Sparse adalah file dengan blok kosong dan konten asli. File yang lebih tipis menggunakan ruang disk secara lebih efisien dengan menulis informasi singkat yang mewakili blok kosong ke disk alih-alih byte nol aktual yang membentuk blok. Ini membuat ukuran aktual file yang jarang biasanya jauh lebih kecil dari ukurannya.

Namun, CloudWatch agen tidak memperlakukan file sparse secara berbeda dari memperlakukan file normal. Ketika agen membaca file yang lebih tipis, blok kosong diperlakukan sebagai blok "nyata" yang diisi dengan byte nol. Karena itu, CloudWatch agen menerbitkan byte sebanyak ukuran file yang jarang. CloudWatch

Mengkonfigurasi CloudWatch agen untuk menerbitkan file yang jarang dapat menyebabkan CloudWatch biaya yang lebih tinggi dari yang diharapkan, jadi kami sarankan untuk tidak melakukannya. Misalnya, `/var/logs/lastlog` di Linux biasanya file yang sangat jarang, dan kami menyarankan Anda untuk tidak mempublikasikannya CloudWatch.

Menambahkan dimensi kustom ke metrik yang dikumpulkan oleh agen CloudWatch

Untuk menambahkan dimensi khusus seperti tanda ke metrik yang dikumpulkan oleh agen, tambahkan `append_dimensions` bidang ke bagian file konfigurasi agen yang mencantumkan metrik tersebut.

Misalnya, bagian contoh berikut dari file konfigurasi menambahkan dimensi kustom yang diberi nama `stackName` dengan nilai `Prod` ke `cpu` dan `disk` metrik yang dikumpulkan oleh agen.

```
"cpu":{
  "resources":[
    "*"
  ],
  "measurement":[
    "cpu_usage_guest",
    "cpu_usage_nice",
    "cpu_usage_idle"
  ],
  "totalcpu":false,
  "append_dimensions":{
    "stackName":"Prod"
  }
},
"disk":{
  "resources":[
    "/",
    "/tmp"
  ],
  "measurement":[
    "total",
    "used"
  ],
  "append_dimensions":{
    "stackName":"Prod"
  }
}
```

Ingat bahwa setiap kali Anda mengubah file konfigurasi agen, Anda harus memulai ulang agen agar perubahan berlaku.

Beberapa file konfigurasi CloudWatch agen

Di server Linux dan server Windows, Anda dapat mengatur CloudWatch agen untuk menggunakan beberapa file konfigurasi. Sebagai contoh, Anda dapat menggunakan file konfigurasi umum yang mengumpulkan serangkaian metrik, log, dan jejak yang selalu ingin Anda kumpulkan dari semua server di infrastruktur Anda. Kemudian Anda dapat menggunakan file konfigurasi tambahan yang mengumpulkan metrik dari aplikasi tertentu atau dalam situasi tertentu.

Untuk mengatur ini, pertama-tama buat file konfigurasi yang ingin Anda gunakan. Setiap file konfigurasi yang akan digunakan bersama di server yang sama harus memiliki nama file yang berbeda. Anda dapat menyimpan file konfigurasi di server atau di Parameter Store.

Mulai CloudWatch agen menggunakan `fetch-config` opsi dan tentukan file konfigurasi pertama. Untuk menambahkan file konfigurasi kedua ke agen yang berjalan, gunakan perintah yang sama tetapi dengan `append-config` pilihan yang tepat. Semua metrik, log, dan jejak yang tercantum di salah satu file konfigurasi dikumpulkan. Contoh perintah berikut menggambarkan skenario ini menggunakan konfigurasi disimpan sebagai file. Baris pertama memulai agen dengan menggunakan file konfigurasi `infrastructure.json`, dan baris kedua menambahkan file konfigurasi `app.json`.

Contoh perintah berikut adalah untuk Linux.

```
/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2
 -s -c file:/tmp/infrastructure.json
```

```
/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a append-config -m
 ec2 -s -c file:/tmp/app.json
```

Contoh perintah berikut adalah untuk Windows Server.

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1"
 -a fetch-config -m ec2 -s -c file:"C:\Program Files\Amazon\AmazonCloudWatchAgent
\infrastructure.json"
```

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1"
 -a append-config -m ec2 -s -c file:"C:\Program Files\Amazon\AmazonCloudWatchAgent
\app.json"
```

Contoh file konfigurasi berikut menggambarkan penggunaan untuk fitur ini. File konfigurasi pertama digunakan untuk semua server dalam infrastruktur, dan yang kedua hanya mengumpulkan log dari aplikasi tertentu dan ditambahkan untuk server yang menjalankan aplikasi tersebut.

infrastructure.json

```
{
  "metrics": {
    "metrics_collected": {
      "cpu": {
```

```
        "resources": [
          "*"
        ],
        "measurement": [
          "usage_active"
        ],
        "totalcpu": true
      },
      "mem": {
         "measurement": [
           "used_percent"
        ]
      }
    }
  },
  "logs": {
    "logs_collected": {
      "files": {
        "collect_list": [
          {
            "file_path": "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-
agent.log",
            "log_group_name": "amazon-cloudwatch-agent.log"
          },
          {
            "file_path": "/var/log/messages",
            "log_group_name": "/var/log/messages"
          }
        ]
      }
    }
  }
}
```

app.json

```
{
    "logs": {
        "logs_collected": {
            "files": {
                "collect_list": [
                    {
                        "file_path": "/app/app.log*",
```

```
                        "log_group_name": "/app/app.log"
                    }
                ]
            }
        }
    }
}
```

Setiap file konfigurasi yang ditambahkan ke konfigurasi harus memiliki nama file yang berbeda satu sama lain dan dari file konfigurasi awal. Jika Anda menggunakan `append-config` dengan file konfigurasi dengan nama file yang sama sebagai file konfigurasi yang sudah digunakan agen, perintah penambahan menimpa informasi dari file konfigurasi pertama, bukan menambahkannya. Hal ini berlaku bahkan jika dua file konfigurasi dengan nama file yang sama berada di jalur file yang berbeda.

Contoh sebelumnya menunjukkan penggunaan dua file konfigurasi, tetapi tidak ada batas jumlah file konfigurasi yang dapat Anda tambahkan ke konfigurasi agen. Anda juga dapat menggabungkan penggunaan file konfigurasi yang terletak di server dan konfigurasi yang berada di Parameter Store.

Menggabungkan atau menggulung metrik yang dikumpulkan oleh agen CloudWatch

Untuk menggabungkan atau menggulir metrik yang dikumpulkan oleh agen, tambahkan `aggregation_dimensions` bidang ke bagian untuk metrik tersebut dalam file konfigurasi agen.

Misalnya, cuplikan file konfigurasi berikut menggulung metrik di dimensi `AutoScalingGroupName`. Metrik dari semua contoh dalam setiap grup Auto Scaling dikumpulkan dan dapat dilihat secara keseluruhan.

```
"metrics": {
  "cpu":{...}
  "disk":{...}
  "aggregation_dimensions" : [["AutoScalingGroupName"]]
}
```

Untuk menggulung kombinasi dari setiap `InstanceId` dan `InstanceType` selain bergulir pada nama grup Auto Scaling, tambahkan hal berikut.

```
"metrics": {
```

```
  "cpu":{...}
  "disk":{...}
  "aggregation_dimensions" : [["AutoScalingGroupName"], ["InstanceId", "InstanceType"]]
}
```

Untuk menggabungkan metrik menjadi satu koleksi, gunakan `[]`.

```
"metrics": {
  "cpu":{...}
  "disk":{...}
  "aggregation_dimensions" : [[]]
}
```

Ingat bahwa setiap kali mengubah file konfigurasi agen, Anda harus memulai ulang agen agar perubahan berlaku.

Mengumpulkan metrik resolusi tinggi dengan agen CloudWatch

`metrics_collection_interval` menentukan interval waktu untuk metrik yang dikumpulkan, dalam detik. Dengan menentukan nilai kurang dari 60 untuk bidang ini, metrik dikumpulkan sebagai metrik resolusi tinggi.

Sebagai contoh, jika metrik Anda semua harus resolusi tinggi dan dikumpulkan setiap 10 detik, tentukan 10 sebagai nilai untuk `metrics_collection_interval` di bawah bagian `agent` sebagai interval pengumpulan metrik global.

```
"agent": {
  "metrics_collection_interval": 10
}
```

Sebagai alternatif, contoh berikut menetapkan `cpu` metrik yang dikumpulkan setiap detik, dan semua metrik lainnya dikumpulkan setiap menit.

```
"agent":{
  "metrics_collection_interval": 60
},
"metrics":{
  "metrics_collected":{
    "cpu":{
      "resources":[
```

```
        "*"
      ],
      "measurement":[
        "cpu_usage_guest"
      ],
      "totalcpu":false,
      "metrics_collection_interval": 1
    },
    "disk":{
      "resources":[
        "/",
        "/tmp"
      ],
      "measurement":[
        "total",
        "used"
      ]
    }
  }
}
```

Ingat bahwa setiap kali mengubah file konfigurasi agen, Anda harus memulai ulang agen agar perubahan berlaku.

Mengirim metrik, log, dan jejak ke akun lain

Agar CloudWatch agen mengirim metrik, log, atau jejak ke akun lain, tentukan `role_arn` parameter dalam file konfigurasi agen di server pengirim. Nilai `role_arn` menentukan peran IAM dalam akun target yang digunakan agen saat mengirim data ke akun target. Peran ini memungkinkan akun pengiriman untuk mengambil peran yang sesuai dalam akun target saat mengirimkan metrik atau log ke akun target.

Anda juga dapat menentukan string `role_arn` terpisah dalam file konfigurasi agen: satu untuk digunakan saat mengirim metrik, satu lagi untuk mengirim log, dan satu lagi untuk mengirim jejak.

Contoh bagian dari bagian `agent` file konfigurasi mengatur agen yang akan digunakan `CrossAccountAgentRole` saat mengirim data ke akun yang berbeda.

```
{
  "agent": {
    "credentials": {
```

```
      "role_arn": "arn:aws:iam::123456789012:role/CrossAccountAgentRole"
    }
  },
  .....
}
```

Sebagai alternatif, contoh berikut menetapkan peran yang berbeda untuk akun pengiriman yang akan digunakan untuk mengirim metrik, log, dan jejak:

```
"metrics": {
    "credentials": {
     "role_arn": "RoleToSendMetrics"
    },
    "metrics_collected": {....
```

```
"logs": {
    "credentials": {
    "role_arn": "RoleToSendLogs"
    },
    ....
```

Kebijakan diperlukan

Saat menentukan `role_arn` di file konfigurasi agen, Anda juga harus memastikan bahwa peran IAM dari akun pengiriman dan target memiliki kebijakan tertentu. Peran dalam kedua akun pengiriman dan target harus memiliki `CloudWatchAgentServerPolicy`. Untuk informasi selengkapnya tentang menetapkan kebijakan ini ke peran, silakan lihat Buat peran IAM untuk digunakan dengan CloudWatch agen di instans Amazon EC2.

Peran dalam akun pengiriman juga harus mencakup kebijakan berikut. Anda menambahkan kebijakan ini di Izin di konsol IAM saat Anda mengedit peran.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "sts:AssumeRole"
            ],
            "Resource": [
                "arn:aws:iam::target-account-ID:role/agent-role-in-target-account"
            ]
        }
    ]
}
```

Peran dalam akun target harus mencakup kebijakan berikut sehingga kebijakan tersebut mengenali peran IAM yang digunakan oleh akun pengiriman. Anda menambahkan kebijakan ini di Hubungan kepercayaan di konsol IAM saat Anda mengedit peran. Peran dalam akun target tempat Anda menambahkan kebijakan ini adalah peran yang Anda buat di Buat peran IAM dan pengguna untuk digunakan dengan agen CloudWatch . Peran ini adalah peran yang ditentukan dalam *agent-role-in-target-account* dalam kebijakan yang digunakan oleh akun pengiriman.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::sending-account-ID:role/role-in-sender-account"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Perbedaan stempel waktu antara CloudWatch agen terpadu dan agen Log sebelumnya CloudWatch

CloudWatch Agen mendukung serangkaian simbol yang berbeda untuk format stempel waktu, dibandingkan dengan agen CloudWatch Log sebelumnya. Perbedaan ini ditunjukkan pada tabel berikut.

Simbol yang didukung oleh kedua agen	Simbol hanya didukung oleh agen terpadu CloudWatch	Simbol hanya didukung oleh agen CloudWatch Log sebelumnya
%A, %a, %b, %B, %d, %f, %H, %I, %m, %M, %p, %S, %y, %Y, %Z, %z	%-d, %-I, %-m, %-M, %-S	%c,%j, %U, %W, %w

Untuk informasi selengkapnya tentang arti simbol yang didukung oleh CloudWatch agen baru, lihat File Konfigurasi CloudWatch Agen: Bagian Log di Panduan CloudWatch Pengguna Amazon. Untuk informasi tentang simbol yang didukung oleh agen CloudWatch Log, lihat File Konfigurasi Agen di Panduan Pengguna CloudWatch Log Amazon.

Memecahkan masalah agen CloudWatch

Gunakan informasi berikut untuk membantu memecahkan masalah dengan agen. CloudWatch

Topik

CloudWatch parameter baris perintah agen

Untuk melihat daftar lengkap parameter yang didukung oleh CloudWatch agen, masukkan yang berikut ini di baris perintah di komputer tempat Anda menginstalnya:

```
amazon-cloudwatch-agent-ctl -help
```

Menginstal CloudWatch agen menggunakan Run Command gagal

Untuk menginstal CloudWatch agen menggunakan Systems Manager Run Command, Agen SSM pada server target harus versi 2.2.93.0 atau yang lebih baru. Jika Agen SSM Anda bukan versi yang tepat, Anda mungkin melihat kesalahan yang mencakup pesan berikut:

```
no latest version found for package AmazonCloudWatchAgent on platform linux
```

```
failed to download installation package reliably
```

Untuk informasi tentang cara memperbarui versi Agen SSM Anda, silakan lihat Menginstal dan Mengonfigurasi Agen SSM di Panduan Pengguna AWS Systems Manager .

CloudWatch Agen tidak akan memulai

Jika CloudWatch agen gagal memulai, mungkin ada masalah dalam konfigurasi Anda. Informasi konfigurasi masuk `configuration-validation.log` file Anda. File ini berada di `/opt/aws/amazon-cloudwatch-agent/logs/configuration-validation.log` di server Linux dan di `$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\configuration-validation.log` pada server yang menjalankan Server Windows.

Verifikasi bahwa CloudWatch agen sedang berjalan

Anda dapat menanyakan CloudWatch agen untuk mengetahui apakah itu berjalan atau berhenti. Anda dapat menggunakan AWS Systems Manager untuk melakukan hal ini dari jarak jauh. Anda juga dapat menggunakan baris perintah, tetapi hanya untuk memeriksa server on-premise.

Untuk menanyakan status CloudWatch agen menggunakan Run Command

1. Buka konsol Systems Manager di https://console.aws.amazon.com/systems-manager/.

2. Pada panel navigasi, silakan pilih Perintah Eksekusi.

   -atau-

   Jika AWS Systems Manager halaman beranda terbuka, gulir ke bawah dan pilih Jelajahi Jalankan Perintah.
3. Pilih Jalankan perintah.
4. Dalam daftar dokumen Command, pilih tombol di sebelah AmazonCloudWatch- ManageAgent.
5. Di Action pilih, status.
6. Untuk Sumber Konfigurasi Opsional pilih default dan menjaga Lokasi Konfigurasi Opsional kosong.
7. Pada Target area, pilih instans untuk memeriksa.
8. Pilih Jalankan.

Jika agen sedang berjalan, output menyerupai yang berikut.

```
{
       "status": "running",
       "starttime": "2017-12-12T18:41:18",
       "version": "1.73.4"
}
```

Jika agen dihentikan, `"status"` bidang menampilkan `"stopped"`.

Untuk menanyakan status CloudWatch agen secara lokal menggunakan baris perintah

- Pada server Linux, masukkan hal berikut:

  ```
  sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -m ec2 -a
   status
  ```

  Di server yang menjalankan Windows Server, masukkan yang berikut ini PowerShell sebagai administrator:

  ```
  & $Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1 -m
   ec2 -a status
  ```

CloudWatch Agen tidak akan memulai, dan kesalahan menyebutkan Wilayah Amazon EC2

Jika agen tidak memulai dan pesan kesalahan menyebutkan titik akhir Amazon EC2 Region, Anda mungkin telah mengonfigurasi agen untuk memerlukan akses ke titik akhir Amazon EC2 tanpa memberikan akses tersebut.

Sebagai contoh, jika Anda menentukan nilai untuk `append_dimensions` parameter dalam file konfigurasi agen yang bergantung pada metadata EC2 Amazon dan Anda menggunakan proksi, Anda harus memastikan bahwa server dapat mengakses titik akhir untuk Amazon EC2. Untuk informasi selengkapnya tentang titik akhir ini, silakan lihat Amazon Elastic Compute Cloud (Amazon EC2) di. Referensi Umum Amazon Web Services

CloudWatch Agen tidak akan memulai di Windows Server

Di Windows Server, Anda mungkin melihat kesalahan berikut:

```
Start-Service : Service 'Amazon CloudWatch Agent (AmazonCloudWatchAgent)' cannot be
 started due to the following
error: Cannot start service AmazonCloudWatchAgent on computer '.'.
At C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1:113
 char:12
+     $svc | Start-Service
+            ~~~~~~~~~~~~~
    + CategoryInfo          : OpenError:
 (System.ServiceProcess.ServiceController:ServiceController) [Start-Service],
   ServiceCommandException
    + FullyQualifiedErrorId :
 CouldNotStartService,Microsoft.PowerShell.Commands.StartServiceCommand
```

Untuk memperbaiki hal ini, pastikan bahwa layanan server berjalan. Kesalahan ini dapat terlihat jika agen mencoba memulai saat layanan server tidak sedang berjalan.

Jika layanan server sudah berjalan, hal berikut mungkin menjadi masalah. Pada beberapa instalasi Windows Server, CloudWatch agen membutuhkan waktu lebih dari 30 detik untuk memulai. Karena Windows Server secara default hanya memungkinkan 30 detik untuk memulai layanan, hal ini menyebabkan agen gagal dengan kesalahan yang serupa dengan yang berikut:

Untuk mengatasi masalah ini, tingkatkan nilai waktu layanan. Untuk informasi selengkapnya, silakan lihat Layanan tidak dimulai, dan peristiwa 7000 dan 7011 dimasukkan ke dalam log peristiwa Windows.

Di Mana Metriknya?

Jika CloudWatch agen telah berjalan tetapi Anda tidak dapat menemukan metrik yang dikumpulkan olehnya di AWS Management Console atau di AWS CLI, konfirmasikan bahwa Anda menggunakan namespace yang benar. Secara bawaan, bidang nama untuk metrik yang dikumpulkan oleh agen adalah `CWAgent`. Anda dapat menyesuaikan namespace ini menggunakan bidang `namespace` di bagian `metrics` file konfigurasi agen. Jika Anda tidak melihat metrik yang diharapkan, periksa file konfigurasi untuk mengonfirmasi ruangnama yang sedang digunakan.

Saat pertama kali mengunduh paket CloudWatch agen, file konfigurasi agen adalah`amazon-cloudwatch-agent.json`. File ini berada di direktori tempat Anda menjalankan pemandu konfigurasi, atau Anda mungkin telah memindahkannya ke direktori lain. Jika Anda menggunakan pemandu konfigurasi, output file konfigurasi agen dari pemandu bernama `config.json`. Untuk informasi selengkapnya tentang file konfigurasi, termasuk bidang `namespace`, silakan lihat CloudWatch file konfigurasi agen: Bagian metrik.

CloudWatch Agen membutuhkan waktu lama untuk berjalan dalam wadah atau mencatat kesalahan batas hop

Saat menjalankan CloudWatch agen sebagai layanan penampung dan ingin menambahkan dimensi metrik Amazon EC2 ke semua metrik yang dikumpulkan oleh agen, Anda mungkin melihat kesalahan berikut di versi v1.247354.0 agen:

```
2022-06-07T03:36:11Z E! [processors.ec2tagger] ec2tagger: Unable to retrieve Instance
 Metadata Tags. This plugin must only be used on an EC2 instance.
2022-06-07T03:36:11Z E! [processors.ec2tagger] ec2tagger: Please increase hop limit
 to 2 by following this document https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/
configuring-instance-metadata-options.html#configuring-IMDS-existing-instances.
2022-06-07T03:36:11Z E! [telegraf] Error running agent: could not initialize processor
 ec2tagger: EC2MetadataRequestError: failed to get EC2 instance identity document
caused by: EC2MetadataError: failed to make EC2Metadata request
        status code: 401, request id:
caused by:
```

Anda mungkin akan melihat kesalahan ini jika agen mencoba mendapatkan metadata dari IMDSv2 yang ada di dalam sebuah kontainer yang tidak memiliki batas hop yang sesuai. Dalam versi agen lebih awal dari v1.247354.0, Anda dapat mengalami masalah ini tanpa melihat pesan log.

Untuk mengatasi ini, tingkatkan batas hop menjadi 2 dengan mengikuti instruksi di Konfigurasikan opsi metadata instans.

Saya memperbarui konfigurasi agen saya tetapi tidak melihat metrik atau log baru di konsol CloudWatch

Jika Anda memperbarui file konfigurasi CloudWatch agen Anda, saat berikutnya Anda memulai agen, Anda perlu menggunakan **`fetch-config`** opsi. Sebagai contoh, jika Anda menyimpan file yang diperbarui di komputer lokal, masukkan perintah berikut:

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -s -m ec2 -c file:configuration-file-path
```

CloudWatch file dan lokasi agen

Tabel berikut mencantumkan file yang diinstal oleh dan digunakan dengan CloudWatch agen, bersama dengan lokasi mereka di server yang menjalankan Linux atau Windows Server.

File	Lokasi Linux	Lokasi Server Windows
Naskah kendali yang mengendalikan mulai, menghentikan, dan memulai ulang agen.	`/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl` atau `/usr/bin/amazon-cloudwatch-agent-ctl`	`$Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1`
File log tempat agen menulis. Anda mungkin perlu melampirkan ini saat menghubungi AWS Support.	`/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log` atau `/var/log/amazon/amazon-cloudwatch-agent/`	`$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\amazon-cloudwatch-agent.log`

File	Lokasi Linux	Lokasi Server Windows
	`amazon-cloudwatch-agent.log`	
File validasi konfigurasi agen.	`/opt/aws/amazon-cloudwatch-agent/logs/configuration-validation.log` atau `/var/log/amazon/amazon-cloudwatch-agent/configuration-validation.log`	`$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\configuration-validation.log`
File JSON yang digunakan untuk mengonfigurasi agen segera setelah pemandu membuatnya. Untuk informasi selengkapnya, lihat Buat file konfigurasi CloudWatch agen.	`/opt/aws/amazon-cloudwatch-agent/bin/config.json`	`$Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\config.json`
File JSON yang digunakan untuk mengonfigurasi agen jika file konfigurasi ini telah diunduh dari Parameter Store.	`/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json` atau `/etc/amazon/amazon-cloudwatch-agent/amazon-cloudwatch-agent.json`	`$Env:ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.json`
File TOML digunakan untuk menentukan informasi Wilayah dan informasi kredensial yang akan digunakan oleh agen, yang kemudian membatalkan sistem default.	`/opt/aws/amazon-cloudwatch-agent/etc/common-config.toml` atau `/etc/amazon/amazon-cloudwatch-agent/common-config.toml`	`$Env:ProgramData\Amazon\AmazonCloudWatchAgent\common-config.toml`

File	Lokasi Linux	Lokasi Server Windows
File TOMM yang berisi konten yang dikonversi dari file konfigurasi JSON. Script `amazon-cloudwatch-agent-ctl` menghasilkan file ini. Pengguna tidak boleh langsung memodifikasi file ini. Ini dapat berguna untuk memverifikasi bahwa terjemahan JSON ke TOLL berhasil.	`/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.toml` atau `/etc/amazon/amazon-cloudwatch-agent/amazon-cloudwatch-agent.toml`	`$Env:ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.toml`
File YAMM yang berisi konten yang dikonversi dari file konfigurasi JSON. Script `amazon-cloudwatch-agent-ctl` menghasilkan file ini. Anda tidak boleh langsung memodifikasi file ini. File ini dapat berguna untuk memverifikasi bahwa terjemahan JSON ke YAMAL berhasil.	`/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.yaml or /etc/amazon/amazon-cloudwatch-agent/amazon-cloudwatch-agent.yaml`	`$Env:ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.yaml`

Menemukan informasi tentang versi CloudWatch agen

Untuk menemukan nomor versi CloudWatch agen di server Linux, masukkan perintah berikut:

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a status
```

Untuk menemukan nomor versi CloudWatch agen di Windows Server, masukkan perintah berikut:

```
& $Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1 -m ec2
 -a status
```

> **Note**
>
> Menggunakan perintah ini adalah cara yang benar untuk menemukan versi CloudWatch agen. Jika Anda menggunakan Program dan Fitur di Panel Kendali, Anda akan melihat nomor versi yang salah.

Anda juga dapat mengunduh file README tentang perubahan terbaru ke agen, dan file yang menunjukkan nomor versi yang saat ini tersedia untuk diunduh. File-file ini ada di lokasi-lokasi berikut:

- `https://amazoncloudwatch-agent.s3.amazonaws.com/info/latest/RELEASE_NOTES` atau `https://amazoncloudwatch-agent-region.s3.region.amazonaws.com/info/latest/RELEASE_NOTES`
- `https://amazoncloudwatch-agent.s3.amazonaws.com/info/latest/CWAGENT_VERSION` atau `https://amazoncloudwatch-agent-region.s3.region.amazonaws.com/amazoncloudwatch-agent-region/info/latest/CWAGENT_VERSION`

Log yang dihasilkan oleh CloudWatch agen

Agen membuat log ketika sedang bekerja. Log ini mencakup informasi pemecahan masalah. Log ini adalah file `amazon-cloudwatch-agent.log` Anda. File ini berada di `/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log` di server Linux dan di `$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\amazon-cloudwatch-agent.log` pada server yang menjalankan Server Windows.

Anda dapat mengonfigurasi agen untuk mencatat log detail tambahan dalam file `amazon-cloudwatch-agent.log`. Dalam file konfigurasi agen, di `agent` bagian, atur `debug` bidang ke`true`, lalu konfigurasikan ulang dan restart CloudWatch agen. Untuk menonaktifkan logging informasi tambahan ini, tetapkan bidang `debug` ke bidang `false`. Kemudian, konfigurasikan ulang dan restart agen. Untuk informasi selengkapnya, lihat Buat atau edit file konfigurasi CloudWatch agen secara manual.

Di versi 1.247350.0 dan CloudWatch agen yang lebih baru, Anda dapat secara opsional mengatur `aws_sdk_log_level` bidang di `agent` bagian file konfigurasi agen ke satu atau beberapa opsi berikut. Pisahkan beberapa opsi dengan karakter `|`.

- `LogDebug`
- `LogDebugWithSigning`
- `LogDebugWithHTTPBody`
- `LogDebugRequestRetries`
- `LogDebugWithEventStreamBody`

Untuk informasi selengkapnya tentang opsi ini, lihat LogLevelType.

Menghentikan dan memulai kembali agen CloudWatch

Anda dapat menghentikan CloudWatch agen secara manual menggunakan salah satu AWS Systems Manager atau baris perintah.

Untuk menghentikan CloudWatch agen menggunakan Run Command

1. Buka konsol Systems Manager di https://console.aws.amazon.com/systems-manager/.
2. Pada panel navigasi, silakan pilih Perintah Eksekusi.

   -atau-

   Jika AWS Systems Manager halaman beranda terbuka, gulir ke bawah dan pilih Jelajahi Jalankan Perintah.
3. Pilih Jalankan perintah.
4. Dalam daftar dokumen Command, pilih AmazonCloudWatch- ManageAgent.
5. Di area Target, pilih instance tempat Anda menginstal CloudWatch agen.
6. Di daftar Tindakan, pilih berhenti.
7. Tetap kosongkan Sumber Konfigurasi Opsional dan Lokasi Konfigurasi Opsional.
8. Pilih Jalankan.

Untuk menghentikan CloudWatch agen secara lokal menggunakan baris perintah

- Pada server Linux, masukkan hal berikut:

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -m ec2 -a
 stop
```

Di server yang menjalankan Windows Server, masukkan yang berikut ini PowerShell sebagai administrator:

```
& $Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1 -m
 ec2 -a stop
```

Untuk memulai ulang agen, ikuti petunjuk di Mulai CloudWatch agen.

Menyematkan metrik dalam log

Format metrik tersemat CloudWatch akan memungkinkan Anda menghasilkan metrik kustom secara tidak sinkron dalam bentuk log yang ditulis ke Log CloudWatch. Anda dapat menyematkan metrik-metrik kustom bersama dengan data peristiwa log teperinci, dan CloudWatch secara otomatis mengekstrak metrik-metrik kustom sehingga Anda dapat memvisualisasikan dan membunyikan alarm, untuk mendeteksi insiden waktu nyata. Selain itu, peristiwa log terperinci terkait metrik terekstraksi dapat diperiksa dengan menggunakan Wawasan Log CloudWatch untuk memberikan wawasan mendalam tentang akar masalah dari peristiwa-peristiwa operasi.

Format metrik tersemat ini akan membantu Anda dalam menghasilkan metrik-metrik kustom yang dapat ditindaklanjuti dari sumber daya sementara seperti fungsi Lambda dan kontainer Lambda. Dengan menggunakan format metrik tersemat ini untuk mengirim log dari sumber daya sementara tersebut, Anda sekarang dapat membuat metrik kustom dengan mudah tanpa harus memiliki instrumen atau mempertahankan kode terpisah, sekaligus mendapatkan kemampuan analitik yang kuat pada data log Anda.

Pengaturan tidak diperlukan untuk menggunakan format metrik yang disematkan. Strukturkan log Anda dengan mematuhi Spesifikasi format metrik tersemat, atau membuat log dengan menggunakan pustaka klien kami dan mengirimkannya ke Log CloudWatch dengan menggunakan PutLogEvents API atau Agen CloudWatch.

Biaya akan dikenakan untuk penyerapan dan pengarsipan log, serta metrik-metrik kustom yang dihasilkan. Untuk informasi selengkapnya, silakan lihat Penetapan Harga Amazon CloudWatch.

> **Note**
>
> Anda perlu berhati-hati ketika mengonfigurasi ekstraksi metrik Anda karena hal ini akan berdampak pada penggunaan metrik kustom dan penagihannya. Jika Anda secara tidak sengaja membuat metrik-metrik yang didasarkan pada dimensi yang memiliki kardinalitas tinggi (seperti `requestId`), maka format metrik tersemat akan berdasarkan desain membuat sebuah metrik kustom yang sesuai dengan masing-masing kombinasi dimensi yang unik. Untuk informasi selengkapnya, silakan lihat Dimensi.

Topik

- Menampilkan metrik dan log Anda di konsol
- Menyetel alarm pada metrik-metrik yang dibuat dengan format metrik tersemat

Menerbitkan log dengan format metrik tersemat

Anda dapat membuat log format metrik tersemat dengan metode-metode berikut:

- Membuat dan mengirimkan log dengan menggunakan pustaka klien sumber terbuka.
- Membuat log secara manual dengan menggunakan spesifikasi format metrik tersemat, dan kemudian menggunakan agen CloudWatch atau API PutlogEvents untuk mengirim log.

Topik

- Membuat log dalam format metrik tersemat dengan menggunakan pustaka klien
- Spesifikasi: Format metrik tersemat
- Menggunakan API PutLogEvents untuk mengirim log format metrik tersemat secara manual
- Menggunakan agen CloudWatch untuk mengirimkan log format metrik tersemat
- Menggunakan format metrik tersemat dengan AWS Distro for OpenTelemetry

Membuat log dalam format metrik tersemat dengan menggunakan pustaka klien

Amazon menyediakan pustaka klien sumber terbuka yang dapat digunakan untuk membuat log format metrik tersemat. Saat ini pustaka tersebut sudah tersedia untuk bahasa-bahasa dalam daftar berikut. Contoh lengkap untuk pengaturan yang berbeda dapat Anda temukan di pustaka klien kami pada /examples.

Pustaka dan petunjuk tentang cara menggunakannya bisa Anda temukan di Github. Gunakan tautan-tautan berikut ini.

- Node.Js

> **Note**
>
> Untuk Node.js, wajib versi 4.1.1+, 3.0.2+, 2.0.7+ untuk digunakan dengan format log Lambda JSON. Menggunakan versi sebelumnya di lingkungan Lambda seperti itu akan menyebabkan kehilangan metrik.
> Untuk informasi selengkapnya, silakan lihat Mengakses log Amazon CloudWatch untuk AWS Lambda.

- Python
- Java
- C#

Pustaka klien dimaksudkan untuk dapat bekerja di luar kotak dengan agen CloudWatch. Log format metrik tersemat yang dihasilkan akan dikirim ke agen CloudWatch, yang kemudian menggabungkan dan menerbitkannya ke Log CloudWatch untuk Anda.

> **Note**
>
> Saat menggunakan Lambda, Anda tidak memerlukan agen untuk mengirim log ke CloudWatch. Apa pun yang di-log ke STDOUT akan dikirimkan ke Log CloudWatch melalui Agen Pencatatan Log Lambda.

Spesifikasi: Format metrik tersemat

Format metrik tersemat CloudWatch adalah sebuah spesifikasi JSON yang digunakan untuk menginstruksikan Log CloudWatch agar secara otomatis mengekstrak nilai metrik yang tersemat dalam peristiwa log terstruktur. Anda dapat menggunakan CloudWatch untuk membuat grafik dan alarm berdasarkan nilai-nilai metrik terekstraksi.

Kesepakatan spesifikasi format metrik tersemat

Kata kunci "HARUS", "TIDAK HARUS", "WAJIB", "AKAN", "TIDAK AKAN", "SEHARUSNYA", "SEHARUSNYA TIDAK", "DISARANKAN", "MUNGKIN", dan "OPSIONAL" dalam spesifikasi format ini harus ditafsirkan sebagaimana yang dijelaskan dalam Kata Kunci RFC2119.

Istilah "JSON", "teks JSON", "nilai JSON", "anggota", "elemen", "objek", "susunan", "nomor", "string", "boolean", "benar", "salah", dan "tidak sah" dalam spesifikasi format ini akan ditafsirkan sebagaimana ditentukan dalam Notasi Objek JavaScript RFC8259.

> **Note**
>
> Jika Anda berencana untuk membuat alarm-alarm pada metrik yang telah dibuat dengan menggunakan format metrik tersemat, silakan lihat Menyetel alarm pada metrik-metrik yang dibuat dengan format metrik tersemat untuk rekomendasinya.

Struktur dokumen format metrik tersemat

Bagian ini akan menjelaskan struktur dokumen format metrik tersemat. Dokumen format metrik tersemat didefinisikan dalam Notasi Obyek JavaScript RFC8259.

Kecuali jika dinyatakan lain, objek-objek yang ditentukan oleh spesifikasi ini TIDAK BOLEH memuat anggota-anggota tambahan. Anggota yang tidak dikenali oleh spesifikasi ini HARUS diabaikan. Anggota yang ditentukan dalam spesifikasi ini bersifat peka terhadap huruf besar dan kecil.

Format metrik tersemat tunduk pada batasan yang sama seperti peristiwa log Log CloudWatch standar dan dibatasi hingga ukuran maksimum 256 KB.

Dengan format metrik tersemat, Anda dapat melacak pemrosesan log EMF berdasarkan metrik yang diterbitkan di ruang nama `AWS/Logs` akun Anda. Hal ini dapat digunakan untuk melacak pembuatan metrik yang gagal dari EMF, serta apakah kegagalan terjadi diakibatkan oleh penguraian atau validasi. Untuk informasi selengkapnya, silakan lihat Pemantauan dengan metrik CloudWatch.

Simpul akar

Pesan LogEvent HARUS merupakan sebuah objek JSON yang benar tanpa data tambahan di awal atau akhir string pesan LogEvent. Untuk informasi selengkapnya tentang struktur LogEvent, silakan lihat InputLogEvent.

Dokumen format metrik tersemat HARUS memuat anggota tingkat atas berikut pada simpul akar. Ini adalah sebuah objek Objek metadata.

```
{
 "_aws": {
    "CloudWatchMetrics": [ ... ]
```

```
  }
}
```

Simpul akar HARUS memuat semua anggota Anggota target yang ditentukan oleh referensi yang ada di Obyek MetricDirective.

Simpul akar tersebut MUNGKIN memuat anggota-anggota lain yang tidak termasuk dalam persyaratan di atas. Nilai-nilai dari anggota-anggota ini HARUS merupakan jenis JSON yang benar.

Objek metadata

Anggota `_aws` dapat Anda gunakan untuk mewakili metadata tentang muatan yang menginformasikan layanan hilir bagaimana mereka harus memproses LogEvent. Nilai tersebut HARUS menjadi objek dan HARUS berisi anggota-anggota berikut:

- CloudWatchMetrics— Susunan Obyek MetricDirective yang digunakan untuk menginstruksikan CloudWatch untuk mengekstrak metrik dari simpul akar LogEvent.

```
{
  "_aws": {
    "CloudWatchMetrics": [ ... ]
  }
}
```

- Timestamp— Angka yang mewakili stempel waktu yang digunakan untuk metrik yang diekstrak dari peristiwa. Nilai HARUS dinyatakan sebagai angka milidetik setelah 1 Jan 1970 00:00:00 UTC.

```
{
  "_aws": {
    "Timestamp": 1559748430481
  }
}
```

Obyek MetricDirective

Objek MetricDirective menginstruksikan layanan-layanan hilir bahwa LogEvent berisi metrik yang akan diekstraksi dan diterbitkan ke CloudWatch. MetricDirectives HARUS berisi anggota-anggota berikut:

- Namespace— String yang mewakili ruang nama CloudWatch untuk metrik.

- Dimensi— Sebuah Susunan DimensionSet.
- Metrik— Susunan objek MetricDefinition. Susunan ini TIDAK BOLEH memuat lebih dari 100 objek MetricDefinition.

Susunan DimensionSet

Sebuah DimensionSet adalah sebuah susunan string yang memuat kunci-kunci dimensi yang akan diterapkan ke semua metrik yang ada dalam dokumen. Nilai-nilai dalam susunan ini HARUS juga menjadi anggota di simpul-akar—disebut sebagai Anggota target

Sebuah DimensionSet TIDAK BOLEH memuat lebih dari 30 kunci dimensi. Sebuah DimensionSet MUNGKIN kosong.

Anggota target HARUS memiliki sebuah nilai string. Nilai-nilai ini TIDAK BOLEH memuat lebih dari 1024 karakter. Anggota target menentukan sebuah dimensi yang akan diterbitkan sebagai bagian dari identitas metrik. Setiap DimensionSet yang digunakan akan membuat sebuah metrik baru di CloudWatch. Untuk informasi selengkapnya tentang dimensi, silakan lihat Dimensi dan Dimensi-dimensi.

```
{
 "_aws": {
   "CloudWatchMetrics": [
     {
       "Dimensions": [ [ "functionVersion" ] ],
       ...
     }
   ]
 },
 "functionVersion": "$LATEST"
}
```

> **Note**
>
> Anda perlu berhati-hati ketika mengonfigurasi ekstraksi metrik Anda karena hal ini akan berdampak pada penggunaan metrik kustom dan penagihannya. Jika Anda secara tidak sengaja membuat metrik-metrik yang didasarkan pada dimensi yang memiliki kardinalitas tinggi (seperti `requestId`), maka format metrik tersemat akan berdasarkan desain membuat sebuah metrik kustom yang sesuai dengan masing-masing kombinasi dimensi yang unik. Untuk informasi selengkapnya, silakan lihat Dimensi.

Objek MetricDefinition

Sebuah MetricDefinition adalah sebuah objek yang HARUS memuat anggota berikut:

- Nama— Sebuah string Nilai referensi ke sebuah metrik Anggota target. Target metrik HARUS berupa sebuah nilai numerik atau susunan nilai-nilai numerik.

Sebuah MetricDefinition adalah objek MUNGKIN berisi anggota-anggota berikut:

- Unit— Sebuah nilai string OPSIONAL yang mewakili satuan pengukuran untuk metrik yang sesuai. Nilai-nilai HARUS adalah unit metrik CloudWatch yang valid. Untuk informasi selengkapnya tentang unit yang valid, silakan lihat MetricDatum. Jika sebuah nilai tidak disediakan, maka nilai bawaan diasumsikan sebagai TIDAK ADA.
- StorageResolution— Sebuah nilai integer OPSIONAL yang mewakili resolusi penyimpanan untuk metrik yang sesuai. Menyetel nilai ini dengan 1 akan menentukan metrik ini sebagai sebuah metrik resolusi tinggi, sehingga CloudWatch menyimpan metrik tersebut dengan resolusi sub-menit hingga satu detik. Menyetel ini dengan 60 menentukan metrik ini sebagai resolusi standar, yang disimpan CloudWatch pada resolusi 1 menit. Nilai-nilai HARUS berupa resolusi yang didukung CloudWatch yang valid, 1 atau 60. Jika sebuah nilai tidak disediakan, maka nilai bawaan diasumsikan 60.

  Untuk informasi selengkapnya tentang metrik resolusi tinggi, silakan lihat Metrik resolusi tinggi.

> **Note**
>
> Jika Anda berencana untuk membuat alarm-alarm pada metrik yang telah dibuat dengan menggunakan format metrik tersemat, silakan lihat Menyetel alarm pada metrik-metrik yang dibuat dengan format metrik tersemat untuk rekomendasinya.

```
{
  "_aws": {
    "CloudWatchMetrics": [
      {
        "Metrics": [
          {
            "Name": "Time",
            "Unit": "Milliseconds",
```

```
            "StorageResolution": 60
          }
        ],
        ...
      }
    ]
  },
  "Time": 1
}
```

Nilai referensi

Nilai-nilai referensi adalah nilai string yang mereferensikan anggota-anggota Anggota target pada simpul akar. Referensi-referensi ini TIDAK boleh disalahartikan dengan JSON Pointers yang dijelaskan dalam RFC6901. Nilai target tidak dapat disarangkan.

Anggota target

Target yang benar HARUS merupakan anggota pada simpul akar dan tidak dapat menjadi objek yang disarangkan. Sebagai contoh, suatu _reference_value `"A.a"` HARUS yang cocok dengan anggota-anggota berikut ini:

```
{ "A.a" }
```

TIDAK BOLEH cocok dengan anggota yang disarangkan:

```
{ "A": { "a" } }
```

Nilai yang benar dari anggota-anggota target bergantung pada hal yang mereferensikan mereka. Sebuah target metrik HARUS berupa sebuah nilai numerik atau susunan nilai-nilai numerik. Target metrik susunan numerik TIDAK BOLEH memiliki lebih dari 100 anggota. Sebuah target dimensi HARUS memiliki sebuah nilai string.

Contoh format metrik tersemat dan skema JSON

Berikut ini adalah sebuah contoh format metrik tersemat yang benar.

```
{
  "_aws": {
    "Timestamp": 1574109732004,
    "CloudWatchMetrics": [
```

```
      {
        "Namespace": "lambda-function-metrics",
        "Dimensions": [["functionVersion"]],
        "Metrics": [
          {
            "Name": "time",
            "Unit": "Milliseconds",
            "StorageResolution": 60
          }
        ]
      }
    ]
  },
  "functionVersion": "$LATEST",
  "time": 100,
  "requestId": "989ffbf8-9ace-4817-a57c-e4dd734019ee"
}
```

Anda dapat menggunakan skema berikut untuk melakukan validasi terhadap dokumen format metrik tersemat.

```
{
    "type": "object",
    "title": "Root Node",
    "required": [
        "_aws"
    ],
    "properties": {
        "_aws": {
            "$id": "#/properties/_aws",
            "type": "object",
            "title": "Metadata",
            "required": [
                "Timestamp",
                "CloudWatchMetrics"
            ],
            "properties": {
                "Timestamp": {
                    "$id": "#/properties/_aws/properties/Timestamp",
                    "type": "integer",
                    "title": "The Timestamp Schema",
                    "examples": [
                        1565375354953
```

```
                    ]
                },
                "CloudWatchMetrics": {
                    "$id": "#/properties/_aws/properties/CloudWatchMetrics",
                    "type": "array",
                    "title": "MetricDirectives",
                    "items": {
                        "$id": "#/properties/_aws/properties/CloudWatchMetrics/items",
                        "type": "object",
                        "title": "MetricDirective",
                        "required": [
                            "Namespace",
                            "Dimensions",
                            "Metrics"
                        ],
                        "properties": {
                            "Namespace": {
                                "$id": "#/properties/_aws/properties/CloudWatchMetrics/
items/properties/Namespace",
                                "type": "string",
                                "title": "CloudWatch Metrics Namespace",
                                "examples": [
                                    "MyApp"
                                ],
                                "pattern": "^(.*)$",
                                "minLength": 1,
                                "maxLength": 1024
                            },
                            "Dimensions": {
                                "$id": "#/properties/_aws/properties/CloudWatchMetrics/
items/properties/Dimensions",
                                "type": "array",
                                "title": "The Dimensions Schema",
                                "minItems": 1,
                                "items": {
                                    "$id": "#/properties/_aws/properties/
CloudWatchMetrics/items/properties/Dimensions/items",
                                    "type": "array",
                                    "title": "DimensionSet",
                                    "minItems": 0,
                                    "maxItems": 30,
                                    "items": {
                                        "$id": "#/properties/_aws/properties/
CloudWatchMetrics/items/properties/Dimensions/items/items",
```

```
                                        "type": "string",
                                        "title": "DimensionReference",
                                        "examples": [
                                            "Operation"
                                        ],
                                        "pattern": "^(.*)$",
                                        "minLength": 1,
                                        "maxLength": 250
}
                                    }
                                },
                                "Metrics": {
                                    "$id": "#/properties/_aws/properties/CloudWatchMetrics/
items/properties/Metrics",
                                    "type": "array",
                                    "title": "MetricDefinitions",
                                    "items": {
                                        "$id": "#/properties/_aws/properties/
CloudWatchMetrics/items/properties/Metrics/items",
                                        "type": "object",
                                        "title": "MetricDefinition",
                                        "required": [
                                            "Name"
                                        ],
                                        "properties": {
                                            "Name": {
                                                "$id": "#/properties/_aws/properties/
CloudWatchMetrics/items/properties/Metrics/items/properties/Name",
                                                "type": "string",
                                                "title": "MetricName",
                                                "examples": [
                                                    "ProcessingLatency"
                                                ],
                                                "pattern": "^(.*)$",
                                                "minLength": 1,
                                                "maxLength": 1024
                                            },
                                            "Unit": {
                                                "$id": "#/properties/_aws/properties/
CloudWatchMetrics/items/properties/Metrics/items/properties/Unit",
                                                "type": "string",
                                                "title": "MetricUnit",
                                                "examples": [
                                                    "Milliseconds"
```

```
                                        ],
                                        "pattern": "^(Seconds|Microseconds|
Milliseconds|Bytes|Kilobytes|Megabytes|Gigabytes|Terabytes|Bits|Kilobits|Megabits|
Gigabits|Terabits|Percent|Count|Bytes\\/Second|Kilobytes\\/Second|Megabytes\\/Second|
Gigabytes\\/Second|Terabytes\\/Second|Bits\\/Second|Kilobits\\/Second|Megabits\\/
Second|Gigabits\\/Second|Terabits\\/Second|Count\\/Second|None)$"
                                    },
                                    "StorageResolution": {
                                        "$id": "#/properties/_aws/properties/
CloudWatchMetrics/items/properties/Metrics/items/properties/StorageResolution",
                                        "type": "integer",
                                        "title": "StorageResolution",
                                        "examples": [
                                            60
                                        ]
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

Menggunakan API PutLogEvents untuk mengirim log format metrik tersemat secara manual

Anda dapat mengirimkan log format metrik tersemat ke Log CloudWatch dengan menggunakan API PutLogEvents Log CloudWatch. Ketika memanggil PutLogEvents, Anda dapat menyertakan header HTTP berikut untuk menginstruksikan Log CloudWatch sehingga metrik harus diekstraksi, tetapi metrik tersebut tidak lagi diperlukan.

```
x-amzn-logs-format: json/emf
```

Berikut ini adalah contoh lengkap yang menggunakan SDK AWS for Java 2.x:

```
package org.example.basicapp;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsRequest;
import software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsResponse;
import software.amazon.awssdk.services.cloudwatchlogs.model.InputLogEvent;
import software.amazon.awssdk.services.cloudwatchlogs.model.PutLogEventsRequest;

import java.util.Collections;

public class EmbeddedMetricsExample {
        public static void main(String[] args) {

                final String usage = "To run this example, supply a Region code (eg.
 us-east-1), log group, and stream name as command line arguments"
                                + "Ex: PutLogEvents <region-id> <log-group-name>
 <stream-name>";

                if (args.length != 3) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String regionId = args[0];
                String logGroupName = args[1];
                String logStreamName = args[2];

                CloudWatchLogsClient logsClient =
 CloudWatchLogsClient.builder().region(Region.of(regionId)).build();

                // Build a JSON log using the EmbeddedMetricFormat.
                long timestamp = System.currentTimeMillis();
                String message = "{" +
                                "  \"_aws\": {" +
                                "    \"Timestamp\": " + timestamp  + "," +
                                "    \"CloudWatchMetrics\": [" +
                                "      {" +
                                "        \"Namespace\": \"MyApp\"," +
                                "        \"Dimensions\": [[\"Operation\"], [\"Operation
\", \"Cell\"]]," +
                                "        \"Metrics\": [{ \"Name\": \"ProcessingLatency
\", \"Unit\": \"Milliseconds\", \"StorageResolution\": 60 }]" +
                                "      }" +
                                "    ]" +
                                "  }," +
```

```
                                "  \"Operation\": \"Aggregator\"," +
                                "  \"Cell\": \"001\"," +
                                "  \"ProcessingLatency\": 100" +
                                "}";
                InputLogEvent inputLogEvent = InputLogEvent.builder()
                        .message(message)
                        .timestamp(timestamp)
                        .build();

                // Specify the request parameters.
                PutLogEventsRequest putLogEventsRequest = PutLogEventsRequest.builder()
                        .logEvents(Collections.singletonList(inputLogEvent))
                        .logGroupName(logGroupName)
                        .logStreamName(logStreamName)
                        .build();

                logsClient.putLogEvents(putLogEventsRequest);

                System.out.println("Successfully put CloudWatch log event");
        }

}
```

> **Note**
>
> Dengan format metrik tersemat, Anda dapat melacak pemrosesan log EMF berdasarkan metrik yang diterbitkan di ruang nama `AWS/Logs` akun Anda. Hal ini dapat digunakan untuk melacak pembuatan metrik yang gagal dari EMF, serta apakah kegagalan terjadi diakibatkan oleh penguraian atau validasi. Untuk informasi selengkapnya, silakan lihat Pemantauan dengan metrik CloudWatch.

Menggunakan agen CloudWatch untuk mengirimkan log format metrik tersemat

Untuk menggunakan metode ini, pertama Anda perlu melakukan instalasi agen CloudWatch untuk layanan-layanan yang ingin Anda kirimi log format metrik tersemat, dan kemudian Anda dapat mulai mengirim peristiwa.

Agen CloudWatch harus versi 1.230621.0 atau versi yang lebih baru.

> **Note**
>
> Anda tidak perlu melakukan instalasi agen CloudWatch untuk mengirimkan log dari fungsi Lambda.
> Batas waktu fungsi Lambda tidak akan ditangani secara otomatis. Hal ini berarti bahwa jika fungsi Anda mengalami habis waktu sebelum metrik terkuras, maka metrik untuk invokasi tersebut tidak akan ditangkap.

Menginstal agen CloudWatch

Menginstal agen CloudWatch untuk masing-masing layanan untuk mengirimkan log format metrik tersemat.

Melakukan instalasi agen CloudWatch di EC2

Pertama, Anda perlu melakukan instalasi agen CloudWatch pada instans tersebut. Untuk informasi selengkapnya, silakan lihat Instalasi CloudWatch agen.

Setelah Anda selesai melakukan instalasi agen tersebut, konfigurasi agen untuk mendengarkan di port UDP atau TCP untuk log format metrik tersemat. Berikut ini adalah contoh konfigurasi yang mendengarkan pada soket bawaan `tcp:25888`. Untuk informasi selengkapnya tentang konfigurasi agen, silakan lihat Buat atau edit file konfigurasi CloudWatch agen secara manual.

```
{
  "logs": {
    "metrics_collected": {
      "emf": { }
    }
  }
}
```

Melakukan instalasi agen CloudWatch di Amazon ECS

Cara termudah untuk menerapkan agen CloudWatch di Amazon ECS adalah dengan menjalankannya seperti sebuah sespan, yang mendefinisikannya dalam definisi tugas yang sama seperti aplikasi Anda.

Buat file konfigurasi agen

Membuat file konfigurasi agen CloudWatch Anda secara lokal. Dalam contoh ini, jalur file relatif adalah `amazon-cloudwatch-agent.json`.

Untuk informasi selengkapnya tentang konfigurasi agen, silakan lihat Buat atau edit file konfigurasi CloudWatch agen secara manual.

```
{
  "logs": {
    "metrics_collected": {
      "emf": { }
    }
  }
}
```

Mendorong konfigurasi ke SSM Parameter Store

Masukkan perintah berikut untuk mendorong file konfigurasi agen CloudWatch ke AWS Systems Manager (SSM) Parameter Store.

```
aws ssm put-parameter \
    --name "cwagentconfig" \
    --type "String" \
    --value "`cat amazon-cloudwatch-agent.json`" \
    --region "{{region}}"
```

Mengonfigurasi definisi tugas

Mengonfigurasi definisi tugas Anda untuk menggunakan Agen CloudWatch dan membuka port TCP atau UDP. Definisi tugas sampel sederhana yang harus Anda gunakan bergantung pada mode jaringan Anda.

Perhatikan bahwa `webapp` menentukan variabel lingkungan `AWS_EMF_AGENT_ENDPOINT`. Hal ini digunakan oleh pustaka dan harus menunjuk titik akhir yang didengar oleh agen. Selain itu, `cwagent` menentukan `CW_CONFIG_CONTENT` sebagai parameter "valueFrom" yang menunjuk ke konfigurasi SSM yang Anda buat pada langkah sebelumnya.

Bagian ini memuat satu contoh untuk mode jembatan dan satu contoh untuk mode host atau awsvpc. Untuk contoh selengkapnya tentang cara Anda dapat mengonfigurasi agen CloudWatch di Amazon ECS, silakan lihat Tempat penyimpanan sampel Github

Berikut ini adalah contoh untuk mode jembatan. Ketika mengaktifkan jaringan mode jembatan, agen harus ditautkan ke aplikasi Anda dengan menggunakan parameter `links` dan harus ditangani dengan menggunakan nama kontainer.

```
{
  "containerDefinitions": [
          {
              "name": "webapp",
              "links": [ "cwagent" ],
              "image": "my-org/web-app:latest",
              "memory": 256,
              "cpu": 256,
              "environment": [{
                "name": "AWS_EMF_AGENT_ENDPOINT",
                "value": "tcp://cwagent:25888"
              }],
          },
          {
              "name": "cwagent",
              "mountPoints": [],
              "image": "public.ecr.aws/cloudwatch-agent/cloudwatch-agent:latest",
              "memory": 256,
              "cpu": 256,
              "portMappings": [{
                  "protocol": "tcp",
                  "containerPort": 25888
              }],
              "environment": [{
                "name": "CW_CONFIG_CONTENT",
                "valueFrom": "cwagentconfig"
              }],
          }
      ],
}
```

Berikut ini adalah contoh untuk mode host atau mode awsvpc. Ketika menjalankan mode jaringan ini, agen dapat dialamatkan ke `localhost`.

```
{
  "containerDefinitions": [
          {
              "name": "webapp",
```

```
            "image": "my-org/web-app:latest",
            "memory": 256,
            "cpu": 256,
            "environment": [{
              "name": "AWS_EMF_AGENT_ENDPOINT",
              "value": "tcp://127.0.0.1:25888"
            }],
        },
        {
            "name": "cwagent",
            "mountPoints": [],
            "image": "public.ecr.aws/cloudwatch-agent/cloudwatch-agent:latest",
            "memory": 256,
            "cpu": 256,
            "portMappings": [{
                "protocol": "tcp",
                "containerPort": 25888
            }],
            "environment": [{
              "name": "CW_CONFIG_CONTENT",
              "valueFrom": "cwagentconfig"
            }],
        }
    ],
}
```

Note

Dalam mode awsvpc, Anda harus memberikan alamat IP publik ke VPC (khusus Fargate), menyiapkan NAT gateway, atau menyiapkan titik akhir VPC Log CloudWatch. Untuk informasi selengkapnya tentang menyiapkan sebuah NAT, silakan lihat NAT Gateway. Untuk informasi selengkapnya tentang menyiapkan titik akhir VPC Log CloudWatch, silakan lihat Menggunakan Log CloudWatch dengan Titik Akhir VPC Antarmuka.
Berikut ini adalah contoh tentang cara menetapkan alamat IP publik untuk sebuah tugas yang menggunakan jenis peluncuran Fargate.

```
aws ecs run-task \
--cluster {{cluster-name}} \
--task-definition cwagent-fargate \
--region {{region}} \
--launch-type FARGATE \
```

```
--network-configuration
 "awsvpcConfiguration={subnets=[{{subnetId}}],securityGroups=[{{sgId}}],assignPublicIp=ENA
```

Memastikan izin

Memastikan peran IAM yang melaksanakan tugas Anda memiliki izin untuk membaca dari SSM Parameter Store. Anda dapat menambahkan izin ini dengan melampirkan kebijakan AmazonSSMReadOnlyAccess. Untuk melakukan hal itu, masukkan perintah berikut.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/AmazonSSMReadOnlyAccess
 \
--role-name CWAgentECSExecutionRole
```

Menginstal agen CloudWatch pada Amazon EKS

Bagian-bagian dari proses ini dapat dilewati jika Anda sudah melakukan instalasi Wawasan Kontainer CloudWatch pada klaster ini.

Izin

Jika Anda belum melakukan instalasi Wawasan Kontainer, maka pastikan terlebih dahulu bahwa simpul Amazon EKS Anda memiliki izin IAM yang sesuai. Mereka harus memiliki CloudWatchAgentServerPolicy yang dilampirkan. Untuk informasi selengkapnya, silakan lihat Memverifikasi prasyarat.

Membuat ConfigMap

Membuat sebuah ConfigMap untuk agen. ConfigMap tersebut juga memberi tahu agen untuk mendengarkan di port TCP atau UDP. Gunakan ConfigMap berikut.

```
# cwagent-emf-configmap.yaml
apiVersion: v1
data:
  # Any changes here must not break the JSON format
  cwagentconfig.json: |
    {
      "agent": {
        "omit_hostname": true
      },
      "logs": {
```

```
        "metrics_collected": {
          "emf": { }
        }
      }
    }
kind: ConfigMap
metadata:
  name: cwagentemfconfig
  namespace: default
```

Jika Anda sudah melakukan instalasi Wawasan Kontainer, tambahkan baris `"emf": { }` berikut ke ConfigMap yang sudah ada.

Terapkan ConfigMap

Masukkan perintah berikut ini untuk menerapkan ConfigMap.

```
kubectl apply -f cwagent-emf-configmap.yaml
```

Terapkan agen

Untuk menerapkan agen CloudWatch tersebut sebagai sebuah sespan, Anda harus menambahkan agen ke definisi pod Anda, seperti dalam contoh berikut.

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp
  namespace: default
spec:
  containers:
    # Your container definitions go here
    - name: web-app
      image: my-org/web-app:latest
    # CloudWatch Agent configuration
    - name: cloudwatch-agent
      image: public.ecr.aws/cloudwatch-agent/cloudwatch-agent:latest
      imagePullPolicy: Always
      resources:
        limits:
          cpu: 200m
          memory: 100Mi
```

```
        requests:
          cpu: 200m
          memory: 100Mi
      volumeMounts:
        - name: cwagentconfig
          mountPath: /etc/cwagentconfig
      ports:
  # this should match the port configured in the ConfigMap
        - protocol: TCP
          hostPort: 25888
          containerPort: 25888
  volumes:
    - name: cwagentconfig
      configMap:
        name: cwagentemfconfig
```

Menggunakan agen CloudWatch untuk mengirimkan log format metrik tersemat

Setelah Anda selesai melakukan instalasi agen CloudWatch dan sudah beroperasi, Anda dapat mengirimkan log format metrik tersemat melalui TCP atau UDP. Ada dua persyaratan ketika mengirimkan log melalui agen:

- Log harus memuat sebuah kunci `LogGroupName` yang memberi tahu agen tentang grup log yang harus digunakan.
- Setiap peristiwa log harus berada dalam satu baris. Dengan kata lain, sebuah peristiwa log tidak dapat berisi karakter baris baru (\n).

Peristiwa log juga harus mematuhi spesifikasi format metrik tersemat. Untuk informasi selengkapnya, silakan lihat Spesifikasi: Format metrik tersemat .

Jika Anda berencana untuk membuat alarm-alarm pada metrik yang telah dibuat dengan menggunakan format metrik tersemat, silakan lihat Menyetel alarm pada metrik-metrik yang dibuat dengan format metrik tersemat untuk rekomendasinya.

Berikut ini adalah contoh pengiriman peristiwa log secara manual dari bash shell Linux. Sebagai gantinya, Anda dapat menggunakan antarmuka soket UDP yang disediakan oleh bahasa pemrograman yang Anda kehendaki.

```
echo '{"_aws":{"Timestamp":1574109732004,"LogGroupName":"Foo","CloudWatchMetrics":
[{"Namespace":"MyApp","Dimensions":[["Operation"]],"Metrics":
```

```
[{"Name":"ProcessingLatency","Unit":"Milliseconds","StorageResolution":60}]}]},"Operation":"Agg
 \
> /dev/udp/0.0.0.0/25888
```

> **Note**
>
> Dengan format metrik tersemat, Anda dapat melacak pemrosesan log EMF berdasarkan metrik yang diterbitkan di ruang nama `AWS/Logs` akun Anda. Hal ini dapat digunakan untuk melacak pembuatan metrik yang gagal dari EMF, serta apakah kegagalan terjadi diakibatkan oleh penguraian atau validasi. Untuk informasi selengkapnya, silakan lihat Pemantauan dengan metrik CloudWatch.

Menggunakan format metrik tersemat dengan AWS Distro for OpenTelemetry

Anda dapat menggunakan format metrik tersemat sebagai sebuah bagian dari proyek OpenTelemetry. OpenTelemetry merupakan sebuah prakarsa sumber terbuka yang menghilangkan batas dan larangan antara format khusus vendor untuk penelusuran, log, dan metrik dengan menawarkan satu set spesifikasi dan API. Untuk informasi selengkapnya, silakan lihat OpenTelemetry.

Menggunakan format metrik tersemat dengan OpenTelemetry memerlukan dua komponen: sebuah sumber data yang sesuai dengan OpenTelemetry, dan AWS Distro for OpenTelemetry Collector yang diaktifkan untuk digunakan dengan log format metrik tersemat CloudWatch.

Kami memiliki redistribusi komponen OpenTelemetry yang sudah dikonfigurasi sebelumnya, yang dipelihara oleh AWS, untuk membuat orientasi bisa dilakukan semudah mungkin. Untuk informasi selengkapnya tentang cara menggunakan OpenTelemetry dengan format metrik tersemat, selain layanan AWS lainnya, silakan lihat AWS Distro for OpenTelemetry.

Untuk informasi tambahan mengenai dukungan dan penggunaan bahasa, silakan lihat Observabilitas AWS di Github.

Menampilkan metrik dan log Anda di konsol

Setelah Anda membuat log format metrik tersemat yang mengekstrak metrik, Anda dapat menggunakan konsol CloudWatch untuk menampilkan metrik tersebut. Metrik-metrik tersemat

memiliki dimensi yang Anda tentukan ketika membuat log. Selain itu, metrik tersemat yang Anda buat dengan menggunakan pustaka klien memiliki dimensi-dimensi bawaan berikut:

- ServiceType
- ServiceName
- LogGroup

Untuk menampilkan metrik yang dihasilkan dari log format metrik tersemat

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik.
3. Pilih sebuah ruang nama yang Anda tentukan untuk metrik tersemat Anda ketika Anda membuatnya. Jika Anda menggunakan pustaka klien untuk membuat metrik dan tidak menentukan ruang nama, maka Anda harus memilih aws-embedded-metrics. Ini adalah ruang nama bawaan untuk metrik tersemat yang dibuat dengan menggunakan pustaka klien.
4. Pilih sebuah dimensi metrik (misalnya, ServiceName).
5. Tab Semua metrik menampilkan semua metrik dimensi tersebut di ruang nama. Anda dapat melakukan hal berikut:

    a. Untuk menyortir tabel, gunakan judul kolomnya.

    b. Untuk membuat grafik sebuah metrik, pilih kotak centang di sebelah metrik. Untuk memilih semua metrik, pilih kotak centang di baris judul tabel.

    c. Untuk menyaring berdasarkan sumber daya, pilih ID sumber daya, kemudian pilih Tambahkan ke pencarian.

    d. Untuk menyaring berdasarkan metrik, pilih nama metrik, kemudian pilih Tambahkan ke pencarian.

Melakukan kueri log dengan menggunakan Wawasan Log CloudWatch

Anda dapat menjalankan kueri terhadap peristiwa log terperinci yang terkait dengan metrik yang diekstrak dengan menggunakan Wawasan Log CloudWatch untuk memberikan wawasan mendalam tentang penyebab utama dari peristiwa-peristiwa operasional. Salah satu manfaat mengekstrak metrik dari log Anda adalah Anda dapat memfilter log Anda nanti berdasarkan metrik unik (nama metrik ditambah kumpulan dimensi unik) dan nilai-nilai metrik, untuk mendapatkan konteks tentang peristiwa yang berkontribusi terhadap nilai metrik gabungan

Sebagai contoh, untuk mendapatkan id permintaan yang terdampak atau id jejak sinar-x, Anda dapat menjalankan kueri berikut di Wawasan Log CloudWatch.

```
filter Latency > 1000 and Operation = "Aggregator"
| fields RequestId, TraceId
```

Anda juga dapat melakukan penggabungan waktu kueri pada kunci dengan kardinalitas tinggi, seperti mencari temuan tentang pelanggan-pelanggan yang terdampak oleh sebuah peristiwa. Contoh berikut menggambarkan hal ini.

```
filter Latency > 1000 and Operation = "Aggregator"
| stats count() by CustomerId
```

Untuk informasi selengkapnya, silakan lihat Menganalisis Data Log dengan Log CloudWatch Insights

Menyetel alarm pada metrik-metrik yang dibuat dengan format metrik tersemat

Secara umum, membuat alarm pada metrik-metrik yang dihasilkan oleh format metrik tersemat mengikuti pola yang sama seperti membuat alarm pada metrik-metrik lainnya. Untuk informasi selengkapnya, silakan lihat Menggunakan CloudWatch alarm Amazon.

Pembuatan metrik format metrik tersemat bergantung pada alur penerbitan log Anda, karena log tersebut harus diproses oleh Log CloudWatch untuk diubah menjadi metrik. Jadi, penting bagi Anda untuk menerbitkan log secara tepat waktu sehingga titik data metrik Anda dibuat dalam periode waktu di mana alarm dievaluasi.

Jika Anda berencana menggunakan format metrik tersemat untuk mengirim metrik resolusi tinggi dan membuat alarm pada metrik-metrik ini, kami menyarankan Anda untuk membilas log ke Log CloudWatch dengan interval 5 detik atau kurang untuk menghindari penundaan tambahan yang dapat menyebabkan terpicunya alarm sebagian data atau data yang hilang. Jika Anda menggunakan agen CloudWatch, maka Anda dapat menyesuaikan interval pembilasan dengan mengatur parameter `force_flush_interval` di file konfigurasi agen CloudWatch. Nilai ini bawaannya 5 detik.

Jika Anda menggunakan Lambda di platform lain di mana Anda tidak dapat mengontrol interval pembilasan log, maka Anda harus mempertimbangkan untuk menggunakan alarm "M out of N" untuk mengontrol jumlah titik data yang digunakan untuk memicu alarm. Untuk informasi selengkapnya, silakan lihat Melakukan evaluasi alarm.

AWS layanan yang mempublikasikan CloudWatch metrik

AWS Layanan berikut mempublikasikan metrik ke CloudWatch. Untuk informasi tentang metrik dan dimensi, silakan lihat dokumentasi yang ditentukan.

Layanan	Namespace	Dokumentasi
AWS Amplify	`AWS/AmplifyHosting`	Pemantauan
Amazon API Gateway	`AWS/ApiGateway`	Pantau Eksekusi API dengan Amazon CloudWatch
Amazon AppFlow	`AWS/AppFlow`	Memantau Amazon AppFlow dengan Amazon CloudWatch
AWS Layanan Migrasi Aplikasi	`AWS/MGN`	Memantau Layanan Migrasi Aplikasi dengan Amazon CloudWatch
AWS Pelari Aplikasi	`AWS/AppRunner`	Melihat metrik layanan App Runner dilaporkan CloudWatch
AppStream 2.0	`AWS/AppStream`	Memantau Sumber Daya Amazon AppStream 2.0
AWS AppSync	`AWS/AppSync`	CloudWatch Metrik
Amazon Athena	`AWS/Athena`	Memantau Kueri Athena dengan Metrik CloudWatch
Amazon Aurora	`AWS/RDS`	Metrik Amazon Aurora
AWS Backup	`AWS/Backup`	Memantau Metrik AWS Backup dengan CloudWatch
Amazon Bedrock	`AWS/Bedrock`	Memantau Amazon Bedrock dengan Amazon CloudWatch
AWS Billing and Cost Management	`AWS/Billing`	Biaya Pemantauan dengan Peringatan dan Notifikasi

Layanan	Namespace	Dokumentasi
Amazon Braket	`AWS/Braket/By Device`	Memantau Amazon Braket dengan Amazon CloudWatch
AWS Certificate Manager	`AWS/CertificateManager`	CloudWatch Metrik yang Didukung
AWS Private CA	`AWS/ACMPrivateCA`	CloudWatch Metrik yang Didukung
AWS Chatbot	`AWS/Chatbot`	Pemantauan AWS Chatbot dengan Amazon CloudWatch
Amazon Chime	`AWS/ChimeVoiceConnector`	Pemantauan Amazon Chime dengan Amazon CloudWatch
Amazon Chime SDK	`AWS/ChimeSDK`	Metrik Layanan
AWS Client VPN	`AWS/ClientVPN`	Pemantauan dengan Amazon CloudWatch
Amazon CloudFront	`AWS/CloudFront`	Pemantauan CloudFront Aktivitas Menggunakan CloudWatch
AWS CloudHSM	`AWS/CloudHSM`	Mendapatkan CloudWatch Metrik
Amazon CloudSearch	`AWS/CloudSearch`	Memantau CloudSearch Domain Amazon dengan Amazon CloudWatch
AWS CloudTrail	`AWS/CloudTrail`	CloudWatch Metrik yang didukung
CloudWatch agen	`CWAgent` atau namespace kustom	Metrik yang dikumpulkan oleh agen CloudWatch

Layanan	Namespace	Dokumentasi
CloudWatch aliran metrik	`AWS/CloudWatch/MetricStreams`	Memantau aliran metrik Anda dengan CloudWatch metrik
CloudWatch RUM	`AWS/RUM`	CloudWatch metrik yang dapat Anda kumpulkan dengan CloudWatch RUM
CloudWatch Synthetics	`CloudWatchSynthetics`	CloudWatch metrik yang diterbitkan oleh kenari
CloudWatch Log Amazon	`AWS/Logs`	Memantau Penggunaan dengan CloudWatch Metrik
AWS CodeBuild	`AWS/CodeBuild`	Pemantauan AWS CodeBuild
CodeGuru Peninjau Amazon		Monitoring CodeGuru Reviewer dengan Amazon CloudWatch
Amazon Kendra		Memantau Amazon Kendra dengan Amazon CloudWatch
Amazon CodeWhisperer	`AWS/CodeWhisperer`	Pemantauan Amazon CodeWhisperer dengan Amazon CloudWatch
Amazon Cognito	`AWS/Cognito`	Memantau Amazon Cognito
Amazon Comprehend	`AWS/Comprehend`	Memantau titik Amazon Comprehend akhir
AWS Config	`AWS/Config`	AWS Config Metrik Penggunaan dan Sukses
Amazon Connect	`AWS/Connect`	Memantau Amazon Connect di Amazon CloudWatch Metrics

Layanan	Namespace	Dokumentasi
Amazon Data Lifecycle Manager	`AWS/DataLifecycleManager`	Pantau kebijakan Anda menggunakan Amazon CloudWatch
AWS DataSync	`AWS/DataSync`	Memantau Tugas Anda
Amazon DataZone		Memantau Amazon DataZone dengan Amazon CloudWatch
Amazon DevOps Guru	`AWS/DevOps-Guru`	Pemantauan Amazon DevOps Guru dengan Amazon CloudWatch
AWS Database Migration Service	`AWS/DMS`	AWS DMS Tugas Pemantauan
AWS Direct Connect	`AWS/DX`	Pemantauan dengan Amazon CloudWatch
AWS Directory Service	`AWS/DirectoryService`	Gunakan CloudWatch metrik Amazon untuk menentukan kapan harus menambahkan pengontrol domain
Amazon DocumentDB	`AWS/DocDB`	Metrik Amazon DocumentDB
Amazon DynamoDB	`AWS/DynamoDB`	Metrik dan Dimensi DynamoDB
DynamoDB Accelerator (DAX)	`AWS/DAX`	Melihat Metrik dan Dimensi DAX
Amazon EC2	`AWS/EC2`	Memantau Instance Anda Menggunakan CloudWatch

Layanan	Namespace	Dokumentasi
Amazon EC2 Elastic Graphics	`AWS/ElasticGPUs`	Menggunakan CloudWatch metrik untuk memantau Elastic Graphics
Armada Spot Amazon EC2	`AWS/EC2Spot`	CloudWatch Metrik untuk Armada Spot
Amazon EC2 Auto Scaling	`AWS/AutoScaling`	Memantau Grup dan Instans Auto Scaling Anda Menggunakan CloudWatch
AWS Elastic Beanstalk	`AWS/ElasticBeanstalk`	Menerbitkan Metrik CloudWatch Kustom Amazon untuk Lingkungan
Amazon Elastic Block Store	`AWS/EBS`	CloudWatch Metrik Amazon untuk Amazon EBS
Amazon Elastic Container Registry	`AWS/ECR`	Metrik repositori Amazon ECR
Amazon Elastic Container Service	`AWS/ECS`	Metrik Amazon ECS CloudWatch
Amazon ECS melalui CloudWatch Wawasan Kontainer	`ECS/ContainerInsights`	Metrik-metrik Wawasan Kontainer Amazon ECS
Penskalaan otomatis Klaster Amazon ECS	`AWS/ECS/ManagedScaling`	Penskalaan otomatis Klaster Amazon ECS
AWS Elastic Disaster Recovery		CloudWatch Metrik untuk DRS

Layanan	Namespace	Dokumentasi
Amazon Elastic File System	`AWS/EFS`	Monitoring dengan CloudWatch
Amazon Elastic Inference	`AWS/ElasticInference`	Menggunakan CloudWatch Metrik untuk Memantau Amazon Elastic Inference
Amazon EKS melalui CloudWatch Wawasan Kontainer	`ContainerInsights`	Metrik-metrik Wawasan Kontainer Amazon EKS dan Kubernetes
Penyeimbang Beban Elastis	`AWS/ApplicationELB`	CloudWatch Metrik untuk Application Load Balancer
Penyeimbang Beban Elastis	`AWS/NetworkELB`	CloudWatch Metrik untuk Network Load Balancer
Penyeimbang Beban Elastis	`AWS/GatewayELB`	CloudWatch Metrik untuk Load Balancer Gateway Anda
Penyeimbang Beban Elastis	`AWS/ELB`	CloudWatch Metrik untuk Classic Load Balancer
Amazon Elastic Transcoder	`AWS/ElasticTranscoder`	Pemantauan dengan Amazon CloudWatch
Amazon ElastiCache untuk Memcached	`AWS/ElastiCache`	Pemantauan Penggunaan dengan CloudWatch Metrik
Amazon ElastiCache untuk Redis	`AWS/ElastiCache`	Pemantauan Penggunaan dengan CloudWatch Metrik

Layanan	Namespace	Dokumentasi
OpenSearch Layanan Amazon	AWS/ES	Memantau metrik OpenSearch klaster dengan Amazon CloudWatch
Amazon EMR	AWS/ElasticMapReduce	Monitor Metrik dengan CloudWatch
AWS Elemental MediaConnect	AWS/MediaConnect	Pemantauan MediaConnect dengan Amazon CloudWatch
AWS Elemental MediaConvert	AWS/MediaConvert	Menggunakan CloudWatch Metrik untuk Melihat Metrik untuk Sumber Daya AWS Elemental MediaConvert
AWS Elemental MediaLive	AWS/MediaLive	Memantau aktivitas menggunakan CloudWatch metrik Amazon
AWS Elemental MediaPackage	AWS/MediaPackage	Pemantauan AWS Elemental MediaPackage dengan Amazon CloudWatch Metrics
AWS Elemental MediaStore	AWS/MediaStore	Pemantauan AWS Elemental MediaStore dengan Amazon CloudWatch Metrics
AWS Elemental MediaTailor	AWS/MediaTailor	Pemantauan AWS Elemental MediaTailor dengan Amazon CloudWatch
Amazon EventBridge	AWS/Events	Memantau Amazon EventBridge
Amazon FinSpace		Pencatatan dan pemantauan
Amazon Forecast		CloudWatch Metrik untuk Amazon Forecast
Amazon Fraud Detector		Memantau Amazon Fraud Detector dengan Amazon CloudWatch

Layanan	Namespace	Dokumentasi
Amazon FSx for Lustre	AWS/FSx	Memantau Amazon FSx for Lustre
Amazon FSx for OpenZFS	AWS/FSx	Pemantauan dengan Amazon CloudWatch
Amazon FSx for Windows File Server	AWS/FSx	Memantau Amazon FSx for Windows File Server
Amazon FSx untuk ONTAP NetApp	AWS/FSx	Pemantauan dengan Amazon CloudWatch
Amazon FSx for OpenZFS	AWS/FSx	Pemantauan dengan Amazon CloudWatch
Amazon GameLift	AWS/GameLift	Pantau Amazon GameLift dengan CloudWatch
AWS Global Accelerator	AWS/GlobalAccelerator	Menggunakan Amazon CloudWatch dengan AWS Global Accelerator
AWS Glue	Glue	Pemantauan AWS Glue Menggunakan CloudWatch Metrik
AWS Ground Station	AWS/GroundStation	Metrik Menggunakan Amazon CloudWatch
AWS HealthLake	AWS/HealthLake	Monitoring HealthLake dengan CloudWatch
Amazon Inspector	AWS/Inspector	Memantau Amazon Inspector Menggunakan CloudWatch

Layanan	Namespace	Dokumentasi
Amazon Interactive Video Service	AWS/IVS	Memantau Amazon IVS dengan Amazon CloudWatch
Amazon Interactive Video Service Chat	AWS/IVSChat	Memantau Amazon IVS dengan Amazon CloudWatch
AWS IoT	AWS/IoT	AWS IoT Metrik dan Dimensi
AWS IoT Analytics	AWS/IoTAnalytics	Namespace, Metrik, dan Dimensi
AWS IoT FleetWise	AWS/IoTFleetWise	Memantau AWS IoT FleetWise dengan Amazon CloudWatch
AWS IoT SiteWise	AWS/IoTSiteWise	Pemantauan AWS IoT SiteWise dengan CloudWatch metrik Amazon
AWS IoT TwinMaker	AWS/IoTTwinMaker	Pemantauan AWS IoT TwinMaker dengan CloudWatch metrik Amazon
AWS IoT 1-Klik		Memantau AWS IoT 1-Klik dengan Amazon CloudWatch
AWS Key Management Service	AWS/KMS	Monitoring dengan CloudWatch
Amazon Keyspaces (untuk Apache Cassandra)	AWS/Cassandra	Metrik dan Dimensi Amazon Keyspaces
Amazon Kendra		Memantau Amazon Kendra dengan Amazon CloudWatch

Layanan	Namespace	Dokumentasi
Layanan Terkelola Amazon untuk Apache Flink	`AWS/KinesisAnalytics`	Layanan Terkelola untuk Apache Flink untuk Aplikasi SQL: Pemantauan dengan CloudWatch Layanan Terkelola untuk Apache Flink: Menampilkan Layanan Terkelola Amazon untuk Apache Flink, Metrik dan Dimensinya
Amazon Data Firehose	`AWS/Firehose`	Memantau Firehose Menggunakan Metrik CloudWatch
Amazon Kinesis Data Streams	`AWS/Kinesis`	Memantau Amazon Kinesis Data Streams dengan Amazon CloudWatch
Amazon Kinesis Video Streams	`AWS/KinesisVideo`	Memantau Metrik Kinesis Video Streams dengan CloudWatch
AWS Lambda	`AWS/Lambda`	AWS Lambda Metrik-metrik
Amazon Lex	`AWS/Lex`	Memantau Amazon Lex dengan Amazon CloudWatch

AWS License Manager	`AWSLicenseManager/licenseUsage`
	
	`AWS/LicenseManager/LinuxSubscriptions`
		Memantau penggunaan lisensi dengan Amazon CloudWatch
		
		Metrik penggunaan dan CloudWatch alarm Amazon untuk langganan Linux

Amazon Location Service	`AWS/Location`	Metrik Amazon Location Service diekspor ke Amazon CloudWatch
Amazon Lookout for Equipment	`AWS/lookoutequipment`	Memantau Lookout for Equipment dengan Amazon CloudWatch

Layanan	Namespace	Dokumentasi
Amazon Lookout for Metrics	`AWS/LookoutMetrics`	Memantau Lookout for Metrics dengan Amazon CloudWatch
Amazon Lookout for Vision	`AWS/LookoutVision`	Memantau Lookout for Vision dengan Amazon CloudWatch
AWS Modernisasi Mainframe		Memantau Modernisasi AWS Mainframe dengan Amazon CloudWatch
Amazon Machine Learning	`AWS/ML`	Memantau Amazon ML dengan CloudWatch Metrik
Amazon Managed Blockchain	`AWS/managedblockchain`	Gunakan Metrik Simpul Hyperledger Fabric Peer di Amazon Managed Blockchain
Layanan Terkelola Amazon untuk Prometheus	`AWS/Prometheus`	CloudWatch Metrik Amazon
Amazon Managed Streaming untuk Apache Kafka	`AWS/Kafka`	Memantau Amazon MSK dengan Amazon CloudWatch
Amazon Managed Streaming untuk Apache Kafka	`AWS/KafkaConnect`	Memantau MSK Connect

Layanan	Namespace	Dokumentasi
Amazon Managed Workflows for Apache Airflow (MWAA)	AWS/MWAA	Metrik kontainer, antrian, dan database untuk Amazon MWAA
Amazon MemoryDB for Redis	AWS/MemoryDB	CloudWatch Metrik pemantauan
Amazon MQ	AWS/AmazonMQ	Memantau Broker Amazon MQ Menggunakan Amazon CloudWatch
Amazon Neptune	AWS/Neptune	Memantau Neptunus dengan CloudWatch
AWS Network Firewall	AWS/NetworkFirewall	AWS Network Firewall metrik di Amazon CloudWatch
AWS Manajer Jaringan	AWS/NetworkManager	CloudWatch metrik untuk sumber daya lokal
Amazon Nimble Studio	AWS/NimbleStudio	Memantau Studio Nimble dengan Amazon CloudWatch
AWS HealthOmics	AWS/Omics	Pemantauan AWS HealthOmics dengan Amazon CloudWatch
AWS OpsWorks	AWS/OpsWorks	Memantau Tumpukan menggunakan Amazon CloudWatch
AWS Outposts	AWS/Outposts	CloudWatch metrik untuk AWS Outposts
AWS Panorama	AWS/PanoramaDeviceMetrics	Memantau peralatan dan aplikasi dengan Amazon CloudWatch

Layanan	Namespace	Dokumentasi
Amazon Personalize	`AWS/Personalize`	CloudWatch metrik untuk Amazon Personalisasi
Amazon Pinpoint	`AWS/Pinpoint`	Lihat Amazon Pinpoint metrik di CloudWatch
Amazon Polly	`AWS/Polly`	Integrasi CloudWatch dengan Amazon Polly
AWS PrivateLink	`AWS/PrivateLinkEndpoints`	CloudWatch metrik untuk AWS PrivateLink
AWS PrivateLink	`AWS/PrivateLinkServices`	CloudWatch metrik untuk AWS PrivateLink
AWS 5G pribadi	`AWS/Private5G`	CloudWatch Metrik Amazon
Amazon QLDB	`AWS/QLDB`	Memantau data di Amazon QuickSight
Amazon QuickSight	`AWS/QuickSight`	Pemantauan dengan Amazon CloudWatch
Amazon Redshift	`AWS/Redshift`	Data Kinerja Amazon Redshift
Amazon Relational Database Service	`AWS/RDS`	Memantau metrik Amazon RDS dengan Amazon CloudWatch
Amazon Rekognition	`AWS/Rekognition`	Memantau Rekognition dengan Amazon CloudWatch
AWS re:Post Pribadi	`AWS/rePostPrivate`	Pemantauan AWS re:Post Pribadi dengan Amazon CloudWatch

Layanan	Namespace	Dokumentasi
AWS RoboMaker	`AWS/Robomaker`	Pemantauan AWS RoboMaker dengan Amazon CloudWatch
Amazon Route 53	`AWS/Route53`	Memantau Amazon Route 53
Pengendali Pemulihan Aplikasi Route 53	`AWS/Route53RecoveryReadiness`	Menggunakan Amazon CloudWatch dengan Application Recovery Controller
Amazon SageMaker	`AWS/SageMaker`	Monitoring SageMaker dengan CloudWatch
Pipa Bangunan SageMaker Model Amazon	`AWS/SageMaker/ModelBuildingPipeline`	SageMaker Metrik Pipelines
AWS Secrets Manager	`AWS/SecretsManager`	Memantau Secrets Manager dengan Amazon CloudWatch
Amazon Security Lake	`AWS/SecurityLake`	CloudWatch metrik untuk Amazon Security Lake
Service Catalog	`AWS/ServiceCatalog`	CloudWatch Metrik Service Catalog
AWS Shield Advanced	`AWS/DDoSProtection`	Monitoring dengan CloudWatch
Layanan Email Sederhana Amazon	`AWS/SES`	Mengambil Data Acara Amazon SES dari CloudWatch
AWS SimSpace Weaver	`AWS/simspaceweaver`	Pemantauan AWS SimSpace Weaver dengan Amazon CloudWatch

Layanan	Namespace	Dokumentasi
Amazon Simple Notification Service	AWS/SNS	Memantau Amazon SNS dengan CloudWatch
Amazon Simple Queue Service	AWS/SQS	Memantau Antrian Amazon SQS Menggunakan CloudWatch
Amazon S3	AWS/S3	Memantau Metrik dengan Amazon CloudWatch
Lensa Penyimpanan S3	AWS/S3/Storage-Lens	Pantau metrik Lensa Penyimpanan S3 di CloudWatch
Layanan Alur Kerja Sederhana Amazon	AWS/SWF	Metrik Amazon SWF untuk CloudWatch
AWS Step Functions	AWS/States	Memantau Step Functions Menggunakan CloudWatch
AWS Storage Gateway	AWS/StorageGateway	Menggunakan CloudWatch metrik Amazon
AWS Systems Manager Jalankan Command	AWS/SSM-RunCommand	Pemantauan Jalankan Metrik Perintah Menggunakan CloudWatch
Amazon Textract	AWS/Textract	CloudWatch Metrik untuk Amazon Texttract
Amazon Timestream	AWS/Timestream	Metrik dan Dimensi Timestream
AWS Transfer for SFTP	AWS/Transfer	AWS SFTP CloudWatch Metrik

Layanan	Namespace	Dokumentasi
Amazon Transcribe	`AWS/Transcribe`	Pemantauan Amazon Transcribe dengan Amazon CloudWatch
Amazon Translate	`AWS/Translate`	CloudWatch Metrik dan Dimensi untuk Amazon Translate
AWS Trusted Advisor	`AWS/TrustedAdvisor`	Membuat Alarm Trusted Advisor Menggunakan CloudWatch
Amazon VPC	`AWS/NATGateway`	Memantau Gateway NAT Anda dengan CloudWatch
Amazon VPC	`AWS/TransitGateway`	CloudWatch Metrik untuk Gateway Transit Anda
Amazon VPC	`AWS/VPN`	Monitoring dengan CloudWatch
Pengelola Alamat IP Amazon VPC	`AWS/IPAM`	Buat alarm dengan Amazon CloudWatch
AWS WAF	`AWS/WAFV2` untuk AWS WAF sumber daya `WAF`untuk sumber daya AWS WAF klasik	Monitoring dengan CloudWatch
Amazon WorkMail	`AWS/WorkMail`	Pemantauan Amazon WorkMail dengan Amazon CloudWatch
Amazon WorkSpaces	`AWS/WorkSpaces`	Pantau CloudWatch Metrik WorkSpaces Penggunaan Anda

Layanan	Namespace	Dokumentasi
WorkSpaces Web Amazon	`AWS/WorkSpacesWeb`	Memantau WorkSpaces Web Amazon dengan Amazon CloudWatch

AWS metrik penggunaan

CloudWatch mengumpulkan metrik yang melacak penggunaan beberapa AWS sumber daya dan API. Metrik-metrik ini diterbitkan di ruang nama `AWS/Usage`. Metrik penggunaan CloudWatch memungkinkan Anda mengelola penggunaan secara proaktif dengan memvisualisasikan metrik di CloudWatch konsol, membuat dasbor khusus, mendeteksi perubahan aktivitas dengan deteksi CloudWatch anomali, dan mengonfigurasi alarm yang mengingatkan Anda saat penggunaan mendekati ambang batas.

Beberapa AWS layanan mengintegrasikan metrik penggunaan ini dengan Service Quotas. Untuk layanan ini, Anda dapat menggunakan CloudWatch untuk mengelola penggunaan akun Anda atas kuota layanan Anda. Untuk informasi selengkapnya, lihat Memvisualisasikan kuota layanan dan mengatur alarm Anda.

Topik

- Memvisualisasikan kuota layanan dan mengatur alarm Anda
- AWS Metrik penggunaan API
- Metrik-metrik penggunaan CloudWatch

Memvisualisasikan kuota layanan dan mengatur alarm Anda

Untuk beberapa AWS layanan, Anda dapat menggunakan metrik penggunaan untuk memvisualisasikan penggunaan layanan Anda saat ini pada CloudWatch grafik dan dasbor. Anda dapat menggunakan fungsi matematika CloudWatch metrik untuk menampilkan kuota layanan untuk sumber daya tersebut pada grafik Anda. Anda juga dapat mengonfigurasi alarm yang memperingatkan Anda ketika penggunaan mendekati kuota layanan. Untuk informasi selengkapnya tentang kuota layanan, silakan lihat Apa itu Kuota Layanan di Panduan Pengguna Kuota Layanan.

Jika Anda masuk ke akun yang disiapkan sebagai akun pemantauan dalam pengamatan CloudWatch lintas akun, Anda dapat menggunakan akun pemantauan tersebut untuk memvisualisasikan kuota layanan dan menyetel alarm untuk metrik di akun sumber yang ditautkan ke akun pemantauan tersebut. Untuk informasi selengkapnya, lihat CloudWatch observabilitas lintas akun.

Saat ini, layanan berikut mengintegrasikan metrik penggunaannya dengan Kuota Layanan:

- AWS CloudHSM
- Amazon Chime SDK

- Amazon CloudWatch
- CloudWatch Log Amazon
- Amazon DynamoDB
- Amazon EC2
- Amazon Elastic Container Registry
- Penyeimbang Beban Elastis
- AWS Fargate
- AWS Fault Injection Service
- AWS Layanan Video Interaktif
- AWS Key Management Service
- Amazon Data Firehose
- Amazon Location Service
- Kueri Amazon Managed Blockchain (AMB)
- AWS RoboMaker
- Amazon SageMaker

Untuk memvisualisasikan kuota layanan dan secara opsional mengatur alarm

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik.
3. Pada tab Semua metrik, pilih Penggunaan, lalu pilih Berdasarkan AWS Sumber Daya.

   Daftar metrik penggunaan kuota layanan muncul.
4. Pilih kotak centang di samping salah satu metrik.

   Grafik menampilkan penggunaan AWS sumber daya Anda saat ini.
5. Untuk menambahkan kuota layanan Anda ke grafik, lakukan hal berikut:

   a. Pilih tab Metrik bergrafik.

   b. Pilih Ekspresi matematika, Mulai dengan ekspresi kosong. Di baris baru, di bawah Detail, masukkan **`SERVICE_QUOTA(m1)`**.

      Baris baru ditambahkan ke grafik, yang menampilkan kuota layanan untuk sumber daya yang diwakili dalam metrik.

6. Untuk melihat penggunaan Anda saat ini sebagai persentase dari kuota, tambahkan ekspresi baru atau ubah ekspresi SERVICE_QUOTA saat ini. Ekspresi baru yang digunakan adalah **`m1/SERVICE_QUOTA(m1)*100`**.
7. (Opsional) Untuk mengatur alarm yang memberi tahu Anda jika mendekati kuota layanan, lakukan hal berikut ini:

   a. Pada baris dengan **`m1/SERVICE_QUOTA(m1)*100`**, di bawah Tindakan, pilih ikon alarm. Ini terlihat seperti lonceng.

      Halaman pembuatan alarm muncul.

   b. Di bawah Ketentuan, pastikan bahwa Jenis ambang batas bersifat Statis dan Setiap kali Expression1 diatur menjadi Lebih Besar. Di bawah dari, masukkan **`80`**. Tindakan ini akan membuat alarm masuk ke status ALARM ketika penggunaan Anda melebihi 80 persen dari kuota.

   c. Pilih Berikutnya.

   d. Di halaman berikutnya, pilih topik Amazon SNS atau buat topik baru, lalu pilih Berikutnya. Topik yang Anda pilih akan diberi tahu ketika alarm masuk ke status ALARM.

   e. Di halaman berikutnya, masukkan nama dan penjelasan untuk alarm, lalu pilih Berikutnya.

   f. Pilih Buat alarm.

AWS Metrik penggunaan API

Sebagian besar API yang mendukung AWS CloudTrail logging juga melaporkan metrik penggunaan ke CloudWatch. Metrik penggunaan API CloudWatch memungkinkan Anda mengelola penggunaan API secara proaktif dengan memvisualisasikan metrik di CloudWatch konsol, membuat dasbor khusus, mendeteksi perubahan aktivitas dengan Deteksi CloudWatch Anomali, dan mengonfigurasi alarm yang mengingatkan saat penggunaan mendekati ambang batas.

Tabel berikut mencantumkan layanan yang melaporkan metrik penggunaan API CloudWatch, dan nilai yang akan digunakan untuk `Service` dimensi untuk melihat metrik penggunaan dari layanan tersebut.

Layanan	Nilai untuk dimensi **`Service`**
AWS Identity and Access Management Access Analyzer	`Access Analyzer`
AWS Account Management	`Account Management`
Alexa for Business	`A4B`
Amazon API Gateway	`API Gateway`
AWS App Mesh	`App Mesh`
AWS AppConfig	`AWS AppConfig`
Amazon AppFlow	`AppFlow`
Application Auto Scaling	`Application Auto Scaling`
Application Discovery Service	`Application Discovery Service`
Amazon AppStream	`AppStream`
AppStream 2.0 Image Builder	`Image Builder`
Amazon Athena	`Athena`
AWS Audit Manager	`Audit Manager`
AWS Backup	`Backup`
AWS Batch	`Batch`
Amazon Braket	`Braket`
AWS Anggaran	`Budgets`
AWS Certificate Manager	`Certificate Manager`
Amazon Chime SDK	`ChimeSDK`

Layanan	Nilai untuk dimensi **`Service`**
Direktori Cloud Amazon	`Cloud Directory`
AWS Cloud Map	`Cloud Map`
AWS CloudFormation	`CloudFormation`
AWS CloudHSM	`CloudHSM`
Amazon CloudSearch	`CloudSearch`
AWS CloudShell	`CloudShell`
AWS CloudTrail	`CloudTrail`
Amazon CloudWatch	`CloudWatch`
CloudWatch Log Amazon	`Logs`
Wawasan CloudWatch Aplikasi Amazon	`CloudWatch Application Insights`
AWS CodeBuild	`CodeBuild`
AWS CodeCommit	`CodeCommit`
Amazon CodeGuru Profiler	`CodeGuru Profiler`
AWS CodePipeline	`CodePipeline`
AWS CodeStar	`CodeStar`
AWS CodeStar Pemberitahuan	`CodeStar Notifications`
AWS CodeStar Koneksi	`CodeStar Connections`
Kolam-kolam identitas Amazon Cognito	`Cognito Identity Pools`
Amazon Cognito Sync	`Cognito Sync`
Amazon Comprehend	`Comprehend`

Layanan	Nilai untuk dimensi **`Service`**
Amazon Comprehend Medical	`Comprehend Medical`
AWS Compute Optimizer	`ComputeOptimzier`
Amazon Connect	`Connect`
Amazon Connect Customer Profiles	`Customer Profiles`
AWS Laporan Biaya dan Penggunaan	`Cost and Usage Report`
AWS Cost Explorer	`Cost Explorer`
AWS Data Exchange	`Data Exchange`
AWS Manajer Siklus Hidup Data	`Data Lifecycle Manager`
AWS Database Migration Service	`Database Migration Service`
AWS DataSync	`DataSync`
AWS DeepLens	`AWS DeepLens`
Amazon Detective	`Detective`
Penasihat Perangkat	`Device Advisor`
AWS Direct Connect	`Direct Connect`
AWS Directory Service	`Directory Service`
DynamoDB Accelerator	`DynamoDBAccelerator`
Amazon EC2	`EC2`
Penskalaan Otomatis EC2	`EC2 Auto Scaling`
Amazon Elastic Container Registry	`ECR Public`
Amazon Elastic Container Service	`ECS`

Layanan	Nilai untuk dimensi **`Service`**
Amazon Elastic File System	`EFS`
Amazon Elastic Kubernetes Service	`EKS`
AWS Elastic Beanstalk	`Elastic Beanstalk`
Amazon Elastic Inference	`Elastic Inference`
Penyeimbang Beban Elastis	`Elastic Load Balancing`
Amazon EMR	`EMR Containers`
AWS Firewall Manager	`Firewall Manager`
Amazon FSx	`FSx`
Amazon GameLift	`GameLift`
AWS Glue DataBrew	`DataBrew`
Amazon Managed Grafana	`Grafana`
AWS IoT Greengrass	`Greengrass`
AWS Ground Station	`Ground Station`
AWS Health API Dan Pemberitahuan	`AWS Health APIs And Notifications`
Amazon Interactive Video Service	`IVS`
AWS IoT Core	`IoT`
AWS IoT 1-Klik	`IoT 1-Click`
AWS IoT Events	`IoT Events`
AWS IoT RoboRunner	`IoT RoboRunner`
AWS IoT SiteWise	`IoT Sitewise`

Layanan	Nilai untuk dimensi **`Service`**
AWS IoT Wireless	`IoT Wireless`
Amazon Kendra	`Kendra`
Amazon Keyspaces (untuk Apache Cassandra)	`Keyspaces`
Layanan Terkelola Amazon untuk Apache Flink	`Kinesis Analytics`
Amazon Data Firehose	`Firehose`
Kinesis Video Streams	`Kinesis Video Streams`
AWS Key Management Service	`KMS`
AWS Lambda	`Lambda`
AWS Launch Wizard	`Launch Wizard`
Amazon Lex	`Amazon Lex`
Amazon Lightsail	`Lightsail`
Amazon Location Service	`Location`
Amazon Lookout for Vision	`Lookout for Vision`
Amazon Machine Learning	`Amazon Machine Learning`
Amazon Macie	`Macie`
Kueri Amazon Managed Blockchain (AMB)	`Amazon Managed Blockchain Query`
AWS Managed Services	`AWS Managed Services`
AWS Marketplace Commerce Analytics	`Marketplace Analytics Service`
AWS Elemental MediaConnect	`MediaConnect`
AWS Elemental MediaConvert	`MediaConvert`

Layanan	Nilai untuk dimensi **`Service`**
AWS Elemental MediaLive	`MediaLive`
AWS Elemental MediaStore	`Mediastore`
AWS Elemental MediaTailor	`MediaTailor`
AWS Mobile Hub	`Mobile Hub`
AWS Network Firewall	`Network Firewall`
AWS OpsWorks	`OpsWorks`
AWS OpsWorks untuk Manajemen Konfigurasi	`OPsWorks CM`
AWS Outposts	`Outposts`
AWS Organizations	`Organizations`
Wawasan Performa Amazon RDS	`Performance Insights`
Amazon Pinpoint	`Pinpoint`
AWS Private Certificate Authority	`Private Certificate Authority`
Layanan Terkelola Amazon untuk Prometheus	`Prometheus`
AWS Proton	`Proton`
Amazon Quantum Ledger Database (Amazon QLDB)	`QLDB`
Amazon RDS	`RDS`
Amazon Redshift	`Redshift Data API`
Amazon Rekognition	`Rekognition`
AWS Resource Access Manager	`Resource Access Manager`
AWS Resource Groups	`Resource Groups`

Layanan	Nilai untuk dimensi **`Service`**
AWS Resource Groups Tagging API	`Resource Groups Tagging API`
AWS RoboMaker	`RoboMaker`
Domain Amazon Route 53	`Route 53 Domains`
Amazon Route 53 Resolver	`Route 53 Resolver`
Amazon S3	`S3`
Amazon S3 Glacier	`Amazon S3 Glacier`
SageMaker Runtime Amazon	`Sagemaker`
Savings Plans	`Savings Plans`
AWS Secrets Manager	`Secrets Manager`
AWS Security Hub	`Security Hub`
AWS Server Migration Service	`AWS Server Migration Service`
AWS Service Catalog AppRegistry	`Service Catalog AppRegistry`
Kuota Layanan	`Service Quotas`
AWS Shield	`Shield`
AWS Penandatangan	`Signer`
Layanan Notifikasi Sederhana Amazon	`SNS`
Layanan Email Sederhana Amazon	`SES`
Amazon Simple Queue Service	`SQS`
Penyimpanan Identitas	`Identity Store`
Storage Gateway	`Storage Gateway`

Layanan	Nilai untuk dimensi **`Service`**
AWS Support	`Support`
Layanan Alur Kerja Sederhana Amazon	`SWF`
Amazon Textract	`Textract`
AWS IoT Things Graph	`ThingsGraph`
Amazon Timestream	`Timestream`
Amazon Transcribe	`Transcribe`
Amazon Translate	`Translate`
Transkripsi streaming Amazon Transcribe	`Transcribe Streaming`
AWS Transfer Family	`Transfer`
AWS WAF	`WAF`
Amazon WorkDocs	`Amazon WorkDocs`
Amazon WorkLink	`WorkLink`
Amazon WorkMail	`Amazon WorkMail`
Amazon WorkSpaces	`Workspaces`
AWS X-Ray	`X-Ray`

Beberapa layanan juga melaporkan metrik penggunaan untuk API tambahan. Untuk melihat apakah API melaporkan metrik penggunaan CloudWatch, gunakan CloudWatch konsol untuk melihat metrik yang dilaporkan oleh layanan tersebut di namespace. `AWS/Usage`

Untuk melihat daftar API layanan yang melaporkan metrik penggunaan CloudWatch

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Metrik.

3. Pada tab Semua metrik, pilih Penggunaan, lalu pilih Berdasarkan AWS Sumber Daya.
4. Di kotak pencarian di dekat daftar metrik, masukkan nama layanan. Metrik difilter oleh layanan yang Anda masukkan.

Metrik-metrik penggunaan CloudWatch

CloudWatch mengumpulkan metrik-metrik yang menelusuri penggunaan beberapa sumber daya AWS. Metrik-metrik ini sesuai dengan kuota layanan AWS. Dengan menelusuri metrik-metrik tersebut dapat membantu Anda mengelola kuota secara proaktif. Untuk informasi selengkapnya, silakan lihat Memvisualisasikan kuota layanan dan mengatur alarm Anda.

Metrik penggunaan kuota layanan adalah ruang nama `AWS/Usage` dan dikumpulkan setiap menit.

Saat ini, satu-satunya nama metrik di ruang nama ini yang diterbitkan CloudWatch adalah `CallCount`. Metrik ini diterbitkan dengan dimensi `Resource`, `Service`, dan `Type`. Dimensi `Resource` menentukan nama operasi API yang sedang ditelusuri. Misalnya, `CallCount` metrik dengan dimensi `"Service": "CloudWatch"`, `"Type": "API"` dan `"Resource": "PutMetricData"` menunjukkan seberapa banyak operasi API `PutMetricData` CloudWatch telah dipanggil di akun Anda.

Metrik `CallCount` tidak memiliki unit tertentu. Statistik yang paling berguna untuk metrik adalah `SUM`, yang menunjukkan total operasi untuk periode 1 menit.

Metrik-metrik

Metrik	Deskripsi
`CallCount`	Jumlah operasi tertentu yang dilakukan di akun Anda.

Dimensi

Dimensi	Deskripsi
`Service`	Nama dari layanan AWS yang berisi sumber daya. Untuk metrik penggunaan CloudWatch, nilai untuk dimensi ini adalah `CloudWatch` .

Dimensi	Deskripsi
`Class`	Kelas sumber daya yang sedang ditelusuri. Metrik penggunaan API CloudWatch menggunakan dimensi ini dengan nilai `None`.
`Type`	Jenis sumber daya yang sedang ditelusuri. Saat ini, ketika dimensi `Service` adalah `CloudWatch` , satu-satunya nilai yang benar untuk `Type` adalah `API`.
`Resource`	Nama operasi API. Nilai yang valid mencakup hal berikut: `DeleteAlarms` , `DeleteDashboards` , `DescribeAlarmHistory` , `DescribeAlarms` , `GetDashboard` , `GetMetricData` , `GetMetricStatistics` , `ListMetrics` , `PutDashboard` , dan `PutMetricData`

Tutorial CloudWatch

Skenario-skenario berikut menggambarkan penggunaan Amazon CloudWatch. Dalam skenario pertama, Anda menggunakan konsol CloudWatch untuk membuat sebuah alarm penagihan yang menelusuri penggunaan AWS Anda dan memberikan Anda notifikasi ketika Anda telah melampaui ambang batas pengeluaran tertentu. Dalam skenario kedua yang lebih canggih, Anda menggunakan AWS Command Line Interface (AWS CLI) untuk menerbitkan satu metrik pada aplikasi hipotetis bernama GetStarted.

Skenario

- Memantau estimasi biaya Anda
- Menerbitkan metrik

Skenario: Memantau perkiraan biaya dengan menggunakan CloudWatch

Dalam skenario ini, Anda membuat alarm Amazon CloudWatch untuk memantau estimasi biaya. Ketika Anda mengaktifkan pemantauan estimasi biaya untuk akun AWS Anda, estimasi biaya akan dihitung dan dikirimkan beberapa kali sehari ke CloudWatch sebagai data metrik.

Data metrik penagihan disimpan di Wilayah AS Timur (Virginia Utara) dan mencerminkan biaya di seluruh dunia. Data ini mencakup estimasi biaya untuk setiap layanan di AWS yang Anda gunakan, serta estimasi total biaya AWS Anda.

Anda dapat memilih untuk menerima peringatan melalui email ketika biaya telah melampaui ambang batas tertentu. Peringatan-peringatan ini dipicu oleh CloudWatch dan pesan dikirimkan menggunakan Amazon Simple Notification Service (Amazon SNS).

> **Note**
>
> Untuk informasi tentang cara menganalisis biaya CloudWatch yang telah ditagihkan kepada Anda, silakan lihat CloudWatch penagihan dan biaya.

Tugas

- Langkah 1: Mengaktifkan peringatan penagihan

Langkah 1: Mengaktifkan peringatan penagihan

Sebelum dapat membuat sebuah alarm untuk membuat estimasi biaya, Anda harus mengaktifkan peringatan penagihan terlebih dahulu, sehingga Anda dapat memantau estimasi biaya AWS Anda dan membuat sebuah alarm dengan menggunakan data metrik penagihan. Setelah Anda mengaktifkan peringatan penagihan, Anda tidak dapat menonaktifkan pengumpulan data, tetapi Anda dapat menghapus alarm penagihan apa pun yang dibuat.

Setelah Anda mengaktifkan peringatan penagihan untuk pertama kalinya, diperlukan waktu sekitar 15 menit sebelum Anda dapat melihat data penagihan dan mengatur alarm penagihan tersebut.

Persyaratan

- Anda harus masuk menggunakan kredensial pengguna root atau sebagai pengguna yang telah diberi izin untuk melihat informasi penagihan.
- Untuk akun penagihan terkonsolidasi, data penagihan untuk masing-masing akun yang dikaitkan dapat Anda temukan dengan cara masuk sebagai akun pembayaran. Anda dapat melihat data penagihan untuk total estimasi biaya dan estimasi biaya berdasarkan layanan untuk setiap akun yang dikaitkan, selain akun konsolidasian tersebut.
- Dalam sebuah akun penagihan konsolidasian, metrik akun yang dikaitkan anggota hanya akan diambil datanya jika akun pembayar mengaktifkan pilihan Terima Pemberitahuan Penagihan. Jika Anda mengubah akun yang merupakan akun manajemen/pembayar, maka Anda harus mengaktifkan peringatan penagihan pada akun manajemen/pembayar yang baru.
- Akun ini tidak boleh menjadi bagian dari Amazon Partner Network (APN) karena metrik penagihan tidak diterbitkan ke CloudWatch untuk akun APN. Untuk informasi selengkapnya mengenai hal ini, silakan lihat Jaringan Partner AWS.

Cara mengaktifkan pemantauan estimasi biaya

1. Buka konsol AWS Billing di https://console.aws.amazon.com/billing/.
2. Pada panel navigasi, silakan pilih Preferensi Penagihan.

3. Pada Preferensi Peringatan pilih Sunting.
4. Pilih Terima Peringatan Penagihan CloudWatch.
5. Pilih Simpan preferensi.

Langkah 2: Membuat sebuah alarm penagihan

> **Important**
>
> Sebelum Anda membuat sebuah alarm penagihan, Anda harus mengatur Wilayah Anda ke AS Timur (Virginia Utara). Data metrik penagihan akan disimpan di Wilayah ini dan mewakili biaya di seluruh dunia. Anda juga harus mengaktifkan peringatan penagihan untuk akun Anda atau di akun manajemen/pembayar (jika Anda menggunakan penagihan konsolidasian). Untuk informasi selengkapnya, silakan lihat Langkah 1: Mengaktifkan peringatan penagihan.

Dalam prosedur ini, Anda membuat sebuah alarm yang akan mengirimkan notifikasi ketika estimasi biaya untuk AWS melebihi ambang batas yang ditentukan.

Cara membuat sebuah alarm penagihan menggunakan konsol CloudWatch

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Alarm, dan kemudian pilih Semua alarm.
3. Pilih Buat alarm.
4. Pilih Pilih metrik. Pada Jelajah, pilih Penagihan, kemudian pilih Total Estimasi Biaya.

   > **Note**
   >
   > Jika Anda tidak melihat metrik Penagihan/Total Estimasi Biaya, aktifkan peringatan penagihan, dan ubah Wilayah Anda ke AS Timur (Virginia Utara). Untuk informasi selengkapnya, silakan lihat Mengaktifkan peringatan penagihan.

5. Pilih kotak untuk metrik EstimatedCharges, dan kemudian pilih Pilih metrik.
6. Untuk Statistik, pilih Maksimum.
7. Untuk Periode, pilih 6 jam.
8. Untuk Jenis ambang batas, pilih Statis.

9. Untuk Setiap kali EstimatedCharges adalah . . ., pilih Lebih besar.
10. Untuk daripada . . ., Anda harus menentukan nilai yang ingin Anda pilih untuk memicu alarm. Sebagai contoh, **`200`** USD.

    Nilai metrik EstimatedCharges hanya boleh menggunakan dolar AS (USD), dan konversi mata uang akan disediakan oleh Amazon Services LLC. Untuk informasi selengkapnya tentang ini, silakan lihat Apa itu AWS Billing?.

    > **Note**
    >
    > Setelah Anda menentukan nilai ambang batas, grafik pratinjau akan menampilkan estimasi biaya untuk bulan berjalan.

11. Pilih Konfigurasi Tambahan dan lakukan hal-hal berikut:

    - Untuk Titik data untuk alarm, tentukan 1 dari 1.
    - Untuk Perlakuan data yang hilang, pilih Perlakukan data yang hilang sebagai hilang.
12. Pilih Berikutnya.
13. Pada Notifikasi, pastikan bahwa Dalam alarm yang dipilih. Kemudian Anda harus menentukan topik Amazon SNS yang akan dikirimkan notifikasinya saat alarm Anda berada dalam status `ALARM`. Topik Amazon SNS tersebut bisa menyertakan alamat email Anda sehingga Anda akan menerima email saat jumlah penagihan melewati ambang batas yang Anda tentukan.

    Anda dapat memilih topik Amazon SNS yang sudah ada, membuat topik Amazon SNS yang baru, atau menggunakan topik ARN untuk memberikan notifikasi ke akun lain. Jika Anda ingin alarm mengirimkan beberapa notifikasi untuk status alarm yang sama atau untuk status alarm yang berbeda, silakan pilih Tambahkan notifikasi.
14. Pilih Berikutnya.
15. Pada Nama dan deskripsi, masukkan nama untuk alarm Anda.

    - (Opsional) Masukkan deskripsi untuk alarm Anda.
16. Pilih Berikutnya.
17. Pada Pratinjau dan buat, periksa apakah konfigurasi Anda sudah benar, kemudian pilih Buat alarm.

Langkah 3: Memeriksa status alarm

Sekarang, periksa status alarm penagihan yang baru saja Anda buat.

Untuk memeriksa status alarm

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Jika perlu, ubah Wilayah ke AS Timur (Virginia Utara). Data metrik penagihan akan disimpan di Wilayah ini dan ini akan mencerminkan biaya di seluruh dunia.
3. Pada panel navigasi, silakan pilih Alarm.
4. Centang kotak di samping alarm. Sampai berlangganan dikonfirmasi, maka akan ditampilkan sebagai "Konfirmasi tertunda". Setelah berlangganan dikonfirmasi, segarkan konsol untuk menampilkan status yang diperbarui.

Langkah 4: Menyunting alarm penagihan

Misalnya, Anda mungkin ingin meningkatkan jumlah uang yang dibelanjakan dengan AWS setiap bulannya dari $200 menjadi $400. Anda dapat menyunting alarm penagihan yang ada dan meningkatkan jumlah uang yang harus terlampaui sebelum alarm dipicu.

Untuk menyunting alarm penagihan

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Jika perlu, ubah Wilayah ke AS Timur (Virginia Utara). Data metrik penagihan akan disimpan di Wilayah ini dan ini akan mencerminkan biaya di seluruh dunia.
3. Pada panel navigasi, silakan pilih Alarm.
4. Pilih kotak centang yang ada di samping alarm dan pilih Tindakan, Modifikasi.
5. Untuk Setiap kali total biaya AWS saya untuk bulan tersebut melebihi, tetapkan jumlah baru yang harus terlampaui untuk memicu alarm dan mengirimkan pemberitahuan email.
6. Pilih Simpan Perubahan.

Langkah 5: Menghapus alarm penagihan

Jika Anda tidak lagi memerlukan alarm penagihan, Anda dapat menghapusnya.

Cara menghapus sebuah alarm penagihan

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Jika perlu, ubah Wilayah ke AS Timur (Virginia Utara). Data metrik penagihan akan disimpan di Wilayah ini dan ini akan mencerminkan biaya di seluruh dunia.
3. Pada panel navigasi, silakan pilih Alarm.
4. Pilih kotak centang yang ada di samping alarm dan pilih Tindakan, Hapus.
5. Ketika diminta untuk mengonfirmasi, pilih Ya, Hapus.

Skenario: Terbitkan metrik ke CloudWatch

Dalam skenario ini, Anda menggunakan AWS Command Line Interface (AWS CLI) untuk menerbitkan satu metrik pada aplikasi hipotetis bernama GetStarted. Jika Anda belum melakukan instalasi dan mengkonfigurasi AWS CLI, silakan lihat Menyiapkan AWS Command Line Interface di Panduan Pengguna AWS Command Line Interface.

Tugas

- Langkah 1: Menentukan konfigurasi data
- Langkah 2: Menambahkan metrik ke CloudWatch
- Langkah 3: Mengambil statistik dari CloudWatch
- Langkah 4: Menampilkan grafik dengan konsol

Langkah 1: Menentukan konfigurasi data

Dalam skenario ini, Anda menerbitkan titik data yang menelusuri latensi permintaan untuk aplikasi tersebut. Pilih nama untuk metrik dan ruang nama yang sesuai bagi Anda. Untuk contoh ini, sebutkan RequestLatency metrik dan tempatkan semua titik data ke dalam ruang nama GetStarted.

Anda menerbitkan beberapa titik data yang secara bersma-sama mewakili tiga jam data latensi. Data mentah terdiri atas 15 pembacaan latensi permintaan yang didistribusikan selama tiga jam. Setiap pembacaan dalam milidetik:

- Jam pertama: 87, 51, 125, 235
- Jam kedua: 121, 113, 189, 65, 89
- Jam ketiga: 100, 47, 133, 98, 100, 328

Anda dapat menerbitkan data ke CloudWatch sebagai titik data tunggal atau gabungan rangkaian poin data yang disebut set statistik. Anda dapat menggabungkan metrik menjadi granularitas serendah satu menit. Anda dapat menerbitkan titik data gabungan ke CloudWatch sebagai set statistik dengan empat kunci yang telah ditetapkan sebelumnya: `Sum`, `Minimum`, `Maximum`, dan `SampleCount`.

Anda menerbitkan titik data dari satu jam sebagai satu titik data. Untuk data dari jam kedua dan ketiga, Anda menggabungkan titik data dan menerbitkan set statistik untuk setiap jam. Nilai-nilai utama ditunjukkan dalam tabel berikut.

Jam	Data mentah	Jumlah	Minimum	Maksimum	SampleCount
1	87				
1	51				
1	125				
1	235				
2	121, 113, 189, 65, 89	577	65	189	5
3	100, 47, 133, 98, 100, 328	806	47	328	6

Langkah 2: Menambahkan metrik ke CloudWatch

Setelah Anda menentukan konfigurasi data, Anda siap untuk menambahkan data.

Untuk menerbitkan titik data ke CloudWatch

1. Dengan perintah segera, jalankan perintah berikut put-metric-data untuk menambahkan data pada jam pertama. Ganti stempel waktu contoh dengan stempel waktu yang merupakan dua jam sebelumnya, dalam Waktu Terkoordinasi Universal (UTC).

```
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted
 \
--timestamp 2016-10-14T20:30:00Z --value 87 --unit Milliseconds
```

```
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted
 \
--timestamp 2016-10-14T20:30:00Z --value 51 --unit Milliseconds
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted
 \
--timestamp 2016-10-14T20:30:00Z --value 125 --unit Milliseconds
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted
 \
--timestamp 2016-10-14T20:30:00Z --value 235 --unit Milliseconds
```

2. Tambahkan data untuk jam kedua, menggunakan stempel waktu satu jam lebih lambat dari jam pertama.

```
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted
 \
--timestamp 2016-10-14T21:30:00Z --statistic-values
 Sum=577,Minimum=65,Maximum=189,SampleCount=5 --unit Milliseconds
```

3. Tambahkan data untuk jam ketiga, hilangkan stempel waktu ke default dengan waktu saat ini.

```
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted
 \
--statistic-values Sum=806,Minimum=47,Maximum=328,SampleCount=6 --unit Milliseconds
```

Langkah 3: Mengambil statistik dari CloudWatch

Sekarang setelah Anda menerbitkan metrik ke CloudWatch, Anda dapat mengambil statistik didasarkan pada metrik tersebut menggunakan perintah get-metric-statistics sebagai berikut. Pastikan untuk menentukan `--start-time` dan `--end-time` cukup jauh di masa lalu untuk membahas stempel waktu paling awal yang Anda terbitkan.

```
aws cloudwatch get-metric-statistics --namespace GetStarted --metric-name
 RequestLatency --statistics Average \
--start-time 2016-10-14T00:00:00Z --end-time 2016-10-15T00:00:00Z --period 60
```

Berikut ini output contohnya:

```
{
 "Datapoints": [],
 "Label": "Request:Latency"
```

```
}
```

Langkah 4: Menampilkan grafik dengan konsol

Setelah Anda menerbitkan metrik ke CloudWatch, Anda dapat menggunakan konsol CloudWatch untuk melihat grafik statistik.

Untuk melihat grafik statistik Anda di konsol

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel Navigasi, silakan pilih Metrik.
3. Di tab Semua metrik, dalam kotak pencarian, ketik RequestLatency dan tekan Enter.
4. Pilih kotak centang untuk metrik RequestLatency. Grafik data metrik ditampilkan di panel atas.

Untuk informasi selengkapnya, silakan lihat Membuat grafik metrik.

Menggunakan CloudWatch dengan AWS SDK

AWS kit pengembangan perangkat lunak (SDK) tersedia untuk banyak bahasa pemrograman populer. Setiap SDK menyediakan API, contoh kode, dan dokumentasi yang memudahkan developer untuk membangun aplikasi dalam bahasa pilihan mereka.

Dokumentasi SDK	Contoh kode
AWS SDK for C++	AWS SDK for C++ contoh kode
AWS SDK for Go	AWS SDK for Go contoh kode
AWS SDK for Java	AWS SDK for Java contoh kode
AWS SDK for JavaScript	AWS SDK for JavaScript contoh kode
AWS SDK for Kotlin	AWS SDK for Kotlin contoh kode
AWS SDK for .NET	AWS SDK for .NET contoh kode
AWS SDK for PHP	AWS SDK for PHP contoh kode
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) contoh kode
AWS SDK for Ruby	AWS SDK for Ruby contoh kode
AWS SDK for Rust	AWS SDK for Rust contoh kode
AWS SDK untuk SAP ABAP	AWS SDK untuk SAP ABAP contoh kode
AWS SDK for Swift	AWS SDK for Swift contoh kode

Untuk contoh khusus untuk CloudWatch, lihatContoh kode untuk CloudWatch menggunakan AWS SDK.

Ketersediaan contoh

Tidak menemukan yang Anda cari? Minta contoh kode menggunakan tautan Berikan umpan balik di bagian bawah halaman ini.

Contoh kode untuk CloudWatch menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan CloudWatch kit pengembangan AWS perangkat lunak (SDK).

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Meskipun tindakan menunjukkan cara memanggil setiap fungsi layanan, Anda dapat melihat tindakan dalam konteks pada skenario yang terkait dan contoh lintas layanan.

Skenario adalah contoh kode yang menunjukkan cara untuk menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan yang sama.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan CloudWatch dengan AWS SDK. Topik ini juga berisi informasi tentang cara memulai dan detail tentang versi SDK sebelumnya.

Memulai

Halo CloudWatch

Contoh kode berikut menunjukkan cara untuk mulai menggunakan CloudWatch.

.NET

AWS SDK for .NET

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace CloudWatchActions;
```

```
public static class HelloCloudWatch
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the Amazon CloudWatch service.
        // Use your AWS profile name, or leave it blank to use the default
 profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonCloudWatch>()
            ).Build();

        // Now the client is available for injection.
        var cloudWatchClient =
 host.Services.GetRequiredService<IAmazonCloudWatch>();

        // You can use await and any of the async methods to get a response.
        var metricNamespace = "AWS/Billing";
        var response = await cloudWatchClient.ListMetricsAsync(new
 ListMetricsRequest
        {
            Namespace = metricNamespace
        });
        Console.WriteLine($"Hello Amazon CloudWatch! Following are some metrics
 available in the {metricNamespace} namespace:");
        Console.WriteLine();
        foreach (var metric in response.Metrics.Take(5))
        {
            Console.WriteLine($"\tMetric: {metric.MetricName}");
            Console.WriteLine($"\tNamespace: {metric.Namespace}");
            Console.WriteLine($"\tDimensions: {string.Join(", ",
 metric.Dimensions.Select(m => $"{m.Name}:{m.Value}"))}");
            Console.WriteLine();
        }
    }
}
```

- Untuk detail API, lihat ListMetricsdi Referensi AWS SDK for .NET API.

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.paginators.ListMetricsIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloService {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <namespace>\s

                Where:
                  namespace - The namespace to filter against (for example, AWS/
EC2).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String namespace = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        listMets(cw, namespace);
        cw.close();
    }

    public static void listMets(CloudWatchClient cw, String namespace) {
        try {
            ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

            ListMetricsIterable listRes = cw.listMetricsPaginator(request);
            listRes.stream()
                    .flatMap(r -> r.metrics().stream())
                    .forEach(metrics -> System.out.println(" Retrieved metric is:
 " + metrics.metricName()));

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat ListMetricsdi Referensi AWS SDK for Java 2.x API.

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
           <namespace>
        Where:
           namespace - The namespace to filter against (for example, AWS/EC2).
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }

    val namespace = args[0]
    listAllMets(namespace)
}

suspend fun listAllMets(namespaceVal: String?) {
    val request = ListMetricsRequest {
        namespace = namespaceVal
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.listMetricsPaginated(request)
            .transform { it.metrics?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.metricName}")
                println("Namespace is ${obj.namespace}")
            }
    }
}
```

- Untuk detail API, lihat ListMetricsdi AWS SDK untuk referensi API Kotlin.

Contoh kode

Tindakan untuk CloudWatch menggunakan AWS SDK

Contoh kode berikut menunjukkan cara melakukan CloudWatch tindakan individual dengan AWS SDK. Kutipan ini memanggil CloudWatch API dan merupakan kutipan kode dari program yang lebih

besar yang harus dijalankan dalam konteks. Setiap contoh menyertakan tautan ke GitHub, di mana Anda dapat menemukan instruksi untuk mengatur dan menjalankan kode.

Contoh berikut hanya mencakup tindakan yang paling umum digunakan. Untuk daftar lengkapnya, lihat Referensi Amazon CloudWatch API.

Contoh-contoh

- Buat CloudWatch dasbor
- Membuat alarm CloudWatch metrik menggunakan AWS SDK
- Buat detektor CloudWatch anomali
- Hapus CloudWatch alarm menggunakan SDK AWS
- Menghapus detektor CloudWatch anomali menggunakan SDK AWS
- Hapus CloudWatch dasbor menggunakan SDK AWS
- Jelaskan riwayat CloudWatch alarm menggunakan AWS SDK
- Jelaskan CloudWatch alarm menggunakan SDK AWS
- Jelaskan CloudWatch alarm untuk metrik menggunakan SDK AWS
- Jelaskan detektor CloudWatch anomali menggunakan SDK AWS
- Nonaktifkan tindakan CloudWatch alarm menggunakan AWS SDK
- Aktifkan tindakan CloudWatch alarm menggunakan AWS SDK
- Dapatkan gambar data CloudWatch metrik
- Dapatkan detail CloudWatch dasbor
- Dapatkan nilai data CloudWatch metrik
- Dapatkan statistik CloudWatch metrik menggunakan AWS SDK
- Daftar CloudWatch dasbor
- Daftar CloudWatch metrik menggunakan SDK AWS
- Masukkan satu set data ke dalam CloudWatch metrik menggunakan AWS SDK
- Masukkan data ke dalam CloudWatch metrik menggunakan AWS SDK

Buat CloudWatch dasbor

Contoh kode berikut menunjukkan cara membuat CloudWatch dasbor Amazon.

Contoh-contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan di dalam konteks. Anda dapat melihat tindakan ini dalam konteks pada contoh kode berikut:

- Mulai metrik, dasbor, dan alarm CloudWatch

.NET

AWS SDK for .NET

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /// <summary>
    /// Set up a dashboard using a call to the wrapper class.
    /// </summary>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="dashboardName">The name of the dashboard.</param>
    /// <returns>A list of validation messages.</returns>
    private static async Task<List<DashboardValidationMessage>> SetupDashboard(
        string customMetricNamespace, string customMetricName, string
dashboardName)
    {
        // Get the dashboard model from configuration.
        var newDashboard = new DashboardModel();
        _configuration.GetSection("dashboardExampleBody").Bind(newDashboard);

        // Add a new metric to the dashboard.
        newDashboard.Widgets.Add(new Widget
        {
            Height = 8,
            Width = 8,
            Y = 8,
            X = 0,
            Type = "metric",
            Properties = new Properties
            {
                Metrics = new List<List<object>>
                    { new() { customMetricNamespace, customMetricName } },
                View = "timeSeries",
                Region = "us-east-1",
```

```
                Stat = "Sum",
                Period = 86400,
                YAxis = new YAxis { Left = new Left { Min = 0, Max = 100 } },
                Title = "Custom Metric Widget",
                LiveData = true,
                Sparkline = true,
                Trend = true,
                Stacked = false,
                SetPeriodToTimeRange = false
            }
        });

        var newDashboardString = JsonSerializer.Serialize(newDashboard,
            new JsonSerializerOptions
            { DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull });
        var validationMessages =
            await _cloudWatchWrapper.PutDashboard(dashboardName,
 newDashboardString);

        return validationMessages;
    }

    /// <summary>
    /// Wrapper to create or add to a dashboard with metrics.
    /// </summary>
    /// <param name="dashboardName">The name for the dashboard.</param>
    /// <param name="dashboardBody">The metric data in JSON for the dashboard.</
param>
    /// <returns>A list of validation messages for the dashboard.</returns>
    public async Task<List<DashboardValidationMessage>> PutDashboard(string
 dashboardName,
        string dashboardBody)
    {
        // Updating a dashboard replaces all contents.
        // Best practice is to include a text widget indicating this dashboard
 was created programmatically.
        var dashboardResponse = await _amazonCloudWatch.PutDashboardAsync(
            new PutDashboardRequest()
            {
                DashboardName = dashboardName,
                DashboardBody = dashboardBody
            });

        return dashboardResponse.DashboardValidationMessages;
```

```
    }
```

- Untuk detail API, lihat PutDashboarddi Referensi AWS SDK for .NET API.

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
   public static void createDashboardWithMetrics(CloudWatchClient cw, String
dashboardName, String fileName) {
       try {
           PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
                   .dashboardName(dashboardName)
                   .dashboardBody(readFileAsString(fileName))
                   .build();

           PutDashboardResponse response = cw.putDashboard(dashboardRequest);
           System.out.println(dashboardName + " was successfully created.");
           List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
           if (messages.isEmpty()) {
               System.out.println("There are no messages in the new Dashboard");
           } else {
               for (DashboardValidationMessage message : messages) {
                   System.out.println("Message is: " + message.message());
               }
           }

       } catch (CloudWatchException | IOException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- Untuk detail API, lihat PutDashboarddi Referensi AWS SDK for Java 2.x API.

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
suspend fun createDashboardWithMetrics(dashboardNameVal: String, fileNameVal:
 String) {
    val dashboardRequest = PutDashboardRequest {
        dashboardName = dashboardNameVal
        dashboardBody = readFileAsString(fileNameVal)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully created.")
        val messages = response.dashboardValidationMessages
        if (messages != null) {
            if (messages.isEmpty()) {
                println("There are no messages in the new Dashboard")
            } else {
                for (message in messages) {
                    println("Message is: ${message.message}")
                }
            }
        }
    }
}
```

- Untuk detail API, lihat PutDashboarddi AWS SDK untuk referensi API Kotlin.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan CloudWatch dengan AWS SDK. Topik ini juga mencakup informasi tentang cara memulai dan detail versi-versi SDK sebelumnya.

Membuat alarm CloudWatch metrik menggunakan AWS SDK

Contoh kode berikut menunjukkan cara membuat atau memperbarui CloudWatch alarm Amazon dan mengaitkannya dengan metrik yang ditentukan, ekspresi matematika metrik, model deteksi anomali, atau kueri Wawasan Metrik.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Memulai menggunakan alarm
- Mulai metrik, dasbor, dan alarm CloudWatch
- Mengelola metrik dan alarm

.NET

AWS SDK for .NET

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /// <summary>
    /// Add a metric alarm to send an email when the metric passes a threshold.
    /// </summary>
    /// <param name="alarmDescription">A description of the alarm.</param>
    /// <param name="alarmName">The name for the alarm.</param>
    /// <param name="comparison">The type of comparison to use.</param>
    /// <param name="metricName">The name of the metric for the alarm.</param>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="threshold">The threshold value for the alarm.</param>
    /// <param name="alarmActions">Optional actions to execute when in an alarm
state.</param>
    /// <returns>True if successful.</returns>
```

```
   public async Task<bool> PutMetricEmailAlarm(string alarmDescription, string
alarmName, ComparisonOperator comparison,
       string metricName, string metricNamespace, double threshold, List<string>
alarmActions = null!)
   {
       try
       {
           var putEmailAlarmResponse = await
_amazonCloudWatch.PutMetricAlarmAsync(
               new PutMetricAlarmRequest()
               {
                   AlarmActions = alarmActions,
                   AlarmDescription = alarmDescription,
                   AlarmName = alarmName,
                   ComparisonOperator = comparison,
                   Threshold = threshold,
                   Namespace = metricNamespace,
                   MetricName = metricName,
                   EvaluationPeriods = 1,
                   Period = 10,
                   Statistic = new Statistic("Maximum"),
                   DatapointsToAlarm = 1,
                   TreatMissingData = "ignore"
               });
           return putEmailAlarmResponse.HttpStatusCode == HttpStatusCode.OK;
       }
       catch (LimitExceededException lex)
       {
           _logger.LogError(lex, $"Unable to add alarm {alarmName}. Alarm quota
has already been reached.");
       }

       return false;
   }

   /// <summary>
   /// Add specific email actions to a list of action strings for a CloudWatch
alarm.
   /// </summary>
   /// <param name="accountId">The AccountId for the alarm.</param>
   /// <param name="region">The region for the alarm.</param>
   /// <param name="emailTopicName">An Amazon Simple Notification Service (SNS)
topic for the alarm email.</param>
```

```
    /// <param name="alarmActions">Optional list of existing alarm actions to
 append to.</param>
    /// <returns>A list of string actions for an alarm.</returns>
    public List<string> AddEmailAlarmAction(string accountId, string region,
        string emailTopicName, List<string>? alarmActions = null)
    {
        alarmActions ??= new List<string>();
        var snsAlarmAction = $"arn:aws:sns:{region}:{accountId}:
{emailTopicName}";
        alarmActions.Add(snsAlarmAction);
        return alarmActions;
    }
```

- Untuk detail API, lihat PutMetricAlarmdi Referensi AWS SDK for .NET API.

C++

SDK for C++

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di Repositori Contoh Kode AWS.

Sertakan file-file yang diperlukan.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

Buat peringatan alarm untuk mengamati metrik.

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::PutMetricAlarmRequest request;
        request.SetAlarmName(alarm_name);
        request.SetComparisonOperator(
            Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
```

```
        request.SetEvaluationPeriods(1);
        request.SetMetricName("CPUUtilization");
        request.SetNamespace("AWS/EC2");
        request.SetPeriod(60);
        request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
        request.SetThreshold(70.0);
        request.SetActionsEnabled(false);
        request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
        request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);

        Aws::CloudWatch::Model::Dimension dimension;
        dimension.SetName("InstanceId");
        dimension.SetValue(instanceId);

        request.AddDimensions(dimension);

        auto outcome = cw.PutMetricAlarm(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to create CloudWatch alarm:" <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully created CloudWatch alarm " << alarm_name
                << std::endl;
        }
```

- Untuk detail API, lihat PutMetricAlarmdi Referensi AWS SDK for C++ API.

CLI

AWS CLI

Untuk mengirim sebuah pesan email Layanan Notifkasi Sederhana Amazon saat pemanfaatan CPU melebihi 70 persen

Contoh berikut menggunakan perintah `put-metric-alarm` untuk mengirim sebuah pesan email Amazon Simple Notification Service ketika penggunaan CPU melebihi 70 persen:

```
aws cloudwatch put-metric-alarm --alarm-name cpu-mon --alarm-description "Alarm
 when CPU exceeds 70 percent" --metric-name CPUUtilization --namespace AWS/
```

```
EC2 --statistic Average --period 300 --threshold 70 --comparison-operator
 GreaterThanThreshold  --dimensions "Name=InstanceId,Value=i-12345678" --
evaluation-periods 2 --alarm-actions arn:aws:sns:us-east-1:111122223333:MyTopic
 --unit Percent
```

Perintah ini akan kembali ke prompt jika berhasil. Jika suatu alarm yang memiliki nama yang sama sudah ada, alarm itu akan ditimpa oleh alarm yang baru.

Cara menentukan beberapa dimensi

Contoh berikut menggambarkan cara menentukan beberapa dimensi. Masing-masing dimensi ditentukan sebagai sebuah pasangan Nama/Nilai, yang menggunakan koma antara nama dan nilai tersebut. Beberapa dimensi dipisahkan dengan satu spasi:

```
aws cloudwatch put-metric-alarm --alarm-name "Default_Test_Alarm3" --alarm-
description "The default example alarm" --namespace "CW EXAMPLE METRICS" --
metric-name Default_Test --statistic Average --period 60 --evaluation-periods 3
 --threshold 50 --comparison-operator GreaterThanOrEqualToThreshold --dimensions
 Name=key1,Value=value1 Name=key2,Value=value2
```

- Untuk detail API, lihat PutMetricAlarmdi Referensi AWS CLI Perintah.

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    public static String createAlarm(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
```

```
            String customMetricName =
rootNode.findValue("customMetricName").asText();
            String alarmName = rootNode.findValue("exampleAlarmName").asText();
            String emailTopic = rootNode.findValue("emailTopic").asText();
            String accountId = rootNode.findValue("accountId").asText();
            String region = rootNode.findValue("region").asText();

            // Create a List for alarm actions.
            List<String> alarmActions = new ArrayList<>();
            alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);
            PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
                    .alarmActions(alarmActions)
                    .alarmDescription("Example metric alarm")
                    .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
                    .threshold(100.00)
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .evaluationPeriods(1)
                    .period(10)
                    .statistic("Maximum")
                    .datapointsToAlarm(1)
                    .treatMissingData("ignore")
                    .build();

            cw.putMetricAlarm(alarmRequest);
            System.out.println(alarmName + " was successfully created!");
            return alarmName;

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- Untuk detail API, lihat PutMetricAlarmdi Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di Repositori Contoh Kode AWS.

Mengimpor modul SDK dan klien dan memanggil API.

```
import { PutMetricAlarmCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  // This alarm triggers when CPUUtilization exceeds 70% for one minute.
  const command = new PutMetricAlarmCommand({
    AlarmName: process.env.CLOUDWATCH_ALARM_NAME, // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
    ComparisonOperator: "GreaterThanThreshold",
    EvaluationPeriods: 1,
    MetricName: "CPUUtilization",
    Namespace: "AWS/EC2",
    Period: 60,
    Statistic: "Average",
    Threshold: 70.0,
    ActionsEnabled: false,
    AlarmDescription: "Alarm when server CPU exceeds 70%",
    Dimensions: [
      {
        Name: "InstanceId",
        Value: process.env.EC2_INSTANCE_ID, // Set the value of EC_INSTANCE_ID to
 the Id of an existing Amazon EC2 instance.
      },
    ],
    Unit: "Percent",
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
```

```
  }
};

export default run();
```

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat PutMetricAlarmdi Referensi AWS SDK for JavaScript API.

SDK untuk JavaScript (v2)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  AlarmName: "Web_Server_CPU_Utilization",
  ComparisonOperator: "GreaterThanThreshold",
  EvaluationPeriods: 1,
  MetricName: "CPUUtilization",
  Namespace: "AWS/EC2",
  Period: 60,
  Statistic: "Average",
  Threshold: 70.0,
  ActionsEnabled: false,
```

```
  AlarmDescription: "Alarm when server CPU exceeds 70%",
  Dimensions: [
    {
      Name: "InstanceId",
      Value: "INSTANCE_ID",
    },
  ],
  Unit: "Percent",
};

cw.putMetricAlarm(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat PutMetricAlarmdi Referensi AWS SDK for JavaScript API.

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
suspend fun putMetricAlarm(alarmNameVal: String, instanceIdVal: String) {

    val dimensionOb = Dimension {
        name = "InstanceId"
        value = instanceIdVal
    }

    val request = PutMetricAlarmRequest {
        alarmName = alarmNameVal
```

```
        comparisonOperator = ComparisonOperator.GreaterThanThreshold
        evaluationPeriods = 1
        metricName = "CPUUtilization"
        namespace = "AWS/EC2"
        period = 60
        statistic = Statistic.fromValue("Average")
        threshold = 70.0
        actionsEnabled = false
        alarmDescription = "An Alarm created by the Kotlin SDK when server CPU
 utilization exceeds 70%"
        unit = StandardUnit.fromValue("Seconds")
        dimensions = listOf(dimensionOb)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(request)
        println("Successfully created an alarm with name $alarmNameVal")
    }
}
```

- Untuk detail API, lihat PutMetricAlarmdi AWS SDK untuk referensi API Kotlin.

Python

SDK for Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource
```

```
   def create_metric_alarm(
       self,
       metric_namespace,
       metric_name,
       alarm_name,
       stat_type,
       period,
       eval_periods,
       threshold,
       comparison_op,
   ):
       """
       Creates an alarm that watches a metric.

       :param metric_namespace: The namespace of the metric.
       :param metric_name: The name of the metric.
       :param alarm_name: The name of the alarm.
       :param stat_type: The type of statistic the alarm watches.
       :param period: The period in which metric data are grouped to calculate
                      statistics.
       :param eval_periods: The number of periods that the metric must be over
the
                            alarm threshold before the alarm is set into an
alarmed
                            state.
       :param threshold: The threshold value to compare against the metric
statistic.
       :param comparison_op: The comparison operation used to compare the
threshold
                             against the metric.
       :return: The newly created alarm.
       """
       try:
           metric = self.cloudwatch_resource.Metric(metric_namespace,
metric_name)
           alarm = metric.put_alarm(
               AlarmName=alarm_name,
               Statistic=stat_type,
               Period=period,
               EvaluationPeriods=eval_periods,
               Threshold=threshold,
               ComparisonOperator=comparison_op,
           )
```

```
            logger.info(
                "Added alarm %s to track metric %s.%s.",
                alarm_name,
                metric_namespace,
                metric_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't add alarm %s to metric %s.%s",
                alarm_name,
                metric_namespace,
                metric_name,
            )
            raise
        else:
            return alarm
```

- Untuk detail API, lihat PutMetricAlarmdi AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK for Ruby

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
# Creates or updates an alarm in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm.
# @param alarm_description [String] A description about the alarm.
# @param metric_name [String] The name of the metric associated with the alarm.
# @param alarm_actions [Array] A list of Strings representing the
#   Amazon Resource Names (ARNs) to execute when the alarm transitions to the
#   ALARM state.
# @param namespace [String] The namespace for the metric to alarm on.
```

```
# @param statistic [String] The statistic for the metric.
# @param dimensions [Array] A list of dimensions for the metric, specified as
#   Aws::CloudWatch::Types::Dimension.
# @param period [Integer] The number of seconds before re-evaluating the metric.
# @param unit [String] The unit of measure for the statistic.
# @param evaluation_periods [Integer] The number of periods over which data is
#   compared to the specified threshold.
# @param theshold [Float] The value against which the specified statistic is
 compared.
# @param comparison_operator [String] The arithmetic operation to use when
#   comparing the specified statistic and threshold.
# @return [Boolean] true if the alarm was created or updated; otherwise, false.
# @example
#   exit 1 unless alarm_created_or_updated?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket',
#     'Objects exist in this bucket for more than 1 day.',
#     'NumberOfObjects',
#     ['arn:aws:sns:us-east-1:111111111111:Default_CloudWatch_Alarms_Topic'],
#     'AWS/S3',
#     'Average',
#     [
#       {
#         name: 'BucketName',
#         value: 'doc-example-bucket'
#       },
#       {
#         name: 'StorageType',
#         value: 'AllStorageTypes'
#       }
#     ],
#     86_400,
#     'Count',
#     1,
#     1,
#     'GreaterThanThreshold'
#   )
def alarm_created_or_updated?(
  cloudwatch_client,
  alarm_name,
  alarm_description,
  metric_name,
  alarm_actions,
  namespace,
```

```
  statistic,
  dimensions,
  period,
  unit,
  evaluation_periods,
  threshold,
  comparison_operator
)
  cloudwatch_client.put_metric_alarm(
    alarm_name: alarm_name,
    alarm_description: alarm_description,
    metric_name: metric_name,
    alarm_actions: alarm_actions,
    namespace: namespace,
    statistic: statistic,
    dimensions: dimensions,
    period: period,
    unit: unit,
    evaluation_periods: evaluation_periods,
    threshold: threshold,
    comparison_operator: comparison_operator
  )
  return true
rescue StandardError => e
  puts "Error creating alarm: #{e.message}"
  return false
end
```

- Untuk detail API, lihat PutMetricAlarmdi Referensi AWS SDK for Ruby API.

SAP ABAP

SDK untuk SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    TRY.
```

```
        lo_cwt->putmetricalarm(
          iv_alarmname                = iv_alarm_name
          iv_comparisonoperator       = iv_comparison_operator
          iv_evaluationperiods        = iv_evaluation_periods
          iv_metricname               = iv_metric_name
          iv_namespace                = iv_namespace
          iv_statistic                = iv_statistic
          iv_threshold                = iv_threshold
          iv_actionsenabled           = iv_actions_enabled
          iv_alarmdescription         = iv_alarm_description
          iv_unit                     = iv_unit
          iv_period                   = iv_period
          it_dimensions               = it_dimensions
        ).
        MESSAGE 'Alarm created.' TYPE 'I'.
      CATCH /aws1/cx_cwtlimitexceededfault.
        MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
    ENDTRY.
```

- Untuk detail API, lihat PutMetricAlarmdi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan CloudWatch dengan AWS SDK. Topik ini juga mencakup informasi tentang cara memulai dan detail versi-versi SDK sebelumnya.

Buat detektor CloudWatch anomali

Contoh kode berikut menunjukkan cara membuat detektor CloudWatch anomali Amazon.

Contoh-contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan di dalam konteks. Anda dapat melihat tindakan ini dalam konteks pada contoh kode berikut:

- Mulai metrik, dasbor, dan alarm CloudWatch

.NET

AWS SDK for .NET

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /// <summary>
    /// Add an anomaly detector for a single metric.
    /// </summary>
    /// <param name="anomalyDetector">A single metric anomaly detector.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
    {
        var putAlarmDetectorResult = await
_amazonCloudWatch.PutAnomalyDetectorAsync(
            new PutAnomalyDetectorRequest()
            {
                SingleMetricAnomalyDetector = anomalyDetector
            });

        return putAlarmDetectorResult.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Untuk detail API, lihat PutAnomalyDetectordi Referensi AWS SDK for .NET API.

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
   public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
       try {
           // Read values from the JSON file.
           JsonParser parser = new JsonFactory().createParser(new
File(fileName));
           com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
           String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
           String customMetricName =
rootNode.findValue("customMetricName").asText();

           SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                   .metricName(customMetricName)
                   .namespace(customMetricNamespace)
                   .stat("Maximum")
                   .build();

           PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
                   .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                   .build();

           cw.putAnomalyDetector(anomalyDetectorRequest);
           System.out.println("Added anomaly detector for metric " +
customMetricName + ".");

       } catch (CloudWatchException | IOException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- Untuk detail API, lihat PutAnomalyDetectordi Referensi AWS SDK for Java 2.x API.

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
suspend fun addAnomalyDetector(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal = SingleMetricAnomalyDetector {
        metricName = customMetricName
        namespace = customMetricNamespace
        stat = "Maximum"
    }

    val anomalyDetectorRequest = PutAnomalyDetectorRequest {
        singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putAnomalyDetector(anomalyDetectorRequest)
        println("Added anomaly detector for metric $customMetricName.")
    }
}
```

- Untuk detail API, lihat PutAnomalyDetectordi AWS SDK untuk referensi API Kotlin.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan CloudWatch dengan AWS SDK. Topik ini juga mencakup informasi tentang cara memulai dan detail versi-versi SDK sebelumnya.

Hapus CloudWatch alarm menggunakan SDK AWS

Contoh kode berikut menunjukkan cara menghapus CloudWatch alarm Amazon.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Memulai menggunakan alarm
- Mulai metrik, dasbor, dan alarm CloudWatch
- Mengelola metrik dan alarm

.NET

AWS SDK for .NET

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
/// <summary>
/// Delete a list of alarms from CloudWatch.
/// </summary>
/// <param name="alarmNames">A list of names of alarms to delete.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteAlarms(List<string> alarmNames)
{
    var deleteAlarmsResult = await _amazonCloudWatch.DeleteAlarmsAsync(
        new DeleteAlarmsRequest()
        {
            AlarmNames = alarmNames
        });

    return deleteAlarmsResult.HttpStatusCode == HttpStatusCode.OK;
}
```

- Untuk detail API, lihat DeleteAlarmsdi Referensi AWS SDK for .NET API.

C++

SDK for C++

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di Repositori Contoh Kode AWS.

Sertakan file-file yang diperlukan.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DeleteAlarmsRequest.h>
#include <iostream>
```

Menghapus alarm.

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::DeleteAlarmsRequest request;
        request.AddAlarmNames(alarm_name);

        auto outcome = cw.DeleteAlarms(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to delete CloudWatch alarm:" <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully deleted CloudWatch alarm " << alarm_name
                << std::endl;
        }
```

- Untuk detail API, lihat DeleteAlarmsdi Referensi AWS SDK for C++ API.

CLI

AWS CLI

Cara menghapus sebuah alarm

Contoh berikut menggunakan `delete-alarms` perintah untuk menghapus CloudWatch alarm Amazon bernama “myalarm”:

```
aws cloudwatch delete-alarms --alarm-names myalarm
```

Output:

```
This command returns to the prompt if successful.
```

- Untuk detail API, lihat DeleteAlarmsdi Referensi AWS CLI Perintah.

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
```

```
public class DeleteAlarm {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <alarmName>

                Where:
                  alarmName - An alarm name to delete (for example, MyAlarm).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarmName = args[0];
        Region region = Region.US_EAST_2;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        deleteCWAlarm(cw, alarmName);
        cw.close();
    }

    public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
        try {
            DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
                    .alarmNames(alarmName)
                    .build();

            cw.deleteAlarms(request);
            System.out.printf("Successfully deleted alarm %s", alarmName);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat DeleteAlarmsdi Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di Repositori Contoh Kode AWS.

Mengimpor modul SDK dan klien dan memanggil API.

```
import { DeleteAlarmsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DeleteAlarmsCommand({
    AlarmNames: [process.env.CLOUDWATCH_ALARM_NAME], // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for JavaScript.

- Untuk detail API, lihat DeleteAlarmsdi Referensi AWS SDK for JavaScript API.

SDK untuk JavaScript (v2)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di Repositori Contoh Kode AWS.

Mengimpor modul SDK dan klien dan memanggil API.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  AlarmNames: ["Web_Server_CPU_Utilization"],
};

cw.deleteAlarms(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat DeleteAlarmsdi Referensi AWS SDK for JavaScript API.

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
suspend fun deleteAlarm(alarmNameVal: String) {
    val request = DeleteAlarmsRequest {
        alarmNames = listOf(alarmNameVal)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAlarms(request)
        println("Successfully deleted alarm $alarmNameVal")
    }
}
```

- Untuk detail API, lihat DeleteAlarmsdi AWS SDK untuk referensi API Kotlin.

Python

SDK for Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
```

```
        """
        self.cloudwatch_resource = cloudwatch_resource


    def delete_metric_alarms(self, metric_namespace, metric_name):
        """
        Deletes all of the alarms that are currently watching the specified
metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(metric_namespace,
metric_name)
            metric.alarms.delete()
            logger.info(
                "Deleted alarms for metric %s.%s.", metric_namespace, metric_name
            )
        except ClientError:
            logger.exception(
                "Couldn't delete alarms for metric %s.%s.",
                metric_namespace,
                metric_name,
            )
            raise
```

- Untuk detail API, lihat DeleteAlarmsdi AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK untuk SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    TRY.
        lo_cwt->deletealarms(
          it_alarmnames = it_alarm_names
        ).
        MESSAGE 'Alarms deleted.' TYPE 'I'.
      CATCH /aws1/cx_cwtresourcenotfound .
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
    ENDTRY.
```

- Untuk detail API, lihat DeleteAlarmsdi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan CloudWatch dengan AWS SDK. Topik ini juga mencakup informasi tentang cara memulai dan detail versi-versi SDK sebelumnya.

Menghapus detektor CloudWatch anomali menggunakan SDK AWS

Contoh kode berikut menunjukkan cara menghapus detektor CloudWatch anomali Amazon.

Contoh-contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan di dalam konteks. Anda dapat melihat tindakan ini dalam konteks pada contoh kode berikut:

- Mulai metrik, dasbor, dan alarm CloudWatch

.NET

AWS SDK for .NET

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /// <summary>
    /// Delete a single metric anomaly detector.
    /// </summary>
    /// <param name="anomalyDetector">The anomaly detector to delete.</param>
```

```
   /// <returns>True if successful.</returns>
   public async Task<bool> DeleteAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
   {
       var deleteAnomalyDetectorResponse = await
_amazonCloudWatch.DeleteAnomalyDetectorAsync(
           new DeleteAnomalyDetectorRequest()
           {
               SingleMetricAnomalyDetector = anomalyDetector
           });

       return deleteAnomalyDetectorResponse.HttpStatusCode == HttpStatusCode.OK;
   }
```

- Untuk detail API, lihat DeleteAnomalyDetectordi Referensi AWS SDK for .NET API.

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
   public static void deleteAnomalyDetector(CloudWatchClient cw, String
fileName) {
       try {
           // Read values from the JSON file.
           JsonParser parser = new JsonFactory().createParser(new
File(fileName));
           com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
           String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
           String customMetricName =
rootNode.findValue("customMetricName").asText();

           SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
```

```
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .stat("Maximum")
                    .build();

            DeleteAnomalyDetectorRequest request =
 DeleteAnomalyDetectorRequest.builder()
                    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                    .build();

            cw.deleteAnomalyDetector(request);
            System.out.println("Successfully deleted the Anomaly Detector.");

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

- Untuk detail API, lihat DeleteAnomalyDetectordi Referensi AWS SDK for Java 2.x API.

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
suspend fun deleteAnomalyDetector(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
```

```
    val singleMetricAnomalyDetectorVal = SingleMetricAnomalyDetector {
        metricName = customMetricName
        namespace = customMetricNamespace
        stat = "Maximum"
    }

    val request = DeleteAnomalyDetectorRequest {
        singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAnomalyDetector(request)
        println("Successfully deleted the Anomaly Detector.")
    }
}
```

- Untuk detail API, lihat DeleteAnomalyDetectordi AWS SDK untuk referensi API Kotlin.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan CloudWatch dengan AWS SDK. Topik ini juga mencakup informasi tentang cara memulai dan detail versi-versi SDK sebelumnya.

Hapus CloudWatch dasbor menggunakan SDK AWS

Contoh kode berikut menunjukkan cara menghapus CloudWatch dasbor Amazon.

Contoh-contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan di dalam konteks. Anda dapat melihat tindakan ini dalam konteks pada contoh kode berikut:

- Mulai metrik, dasbor, dan alarm CloudWatch

.NET

AWS SDK for .NET

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /// <summary>
    /// Delete a list of CloudWatch dashboards.
    /// </summary>
    /// <param name="dashboardNames">List of dashboard names to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDashboards(List<string> dashboardNames)
    {
        var deleteDashboardsResponse = await
_amazonCloudWatch.DeleteDashboardsAsync(
            new DeleteDashboardsRequest()
            {
                DashboardNames = dashboardNames
            });

        return deleteDashboardsResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Untuk detail API, lihat DeleteDashboardsdi Referensi AWS SDK for .NET API.

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    public static void deleteDashboard(CloudWatchClient cw, String dashboardName)
{
        try {
            DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
                    .dashboardNames(dashboardName)
                    .build();
            cw.deleteDashboards(dashboardsRequest);
            System.out.println(dashboardName + " was successfully deleted.");

        } catch (CloudWatchException e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Untuk detail API, lihat DeleteDashboardsdi Referensi AWS SDK for Java 2.x API.

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
suspend fun deleteDashboard(dashboardName: String) {
    val dashboardsRequest = DeleteDashboardsRequest {
        dashboardNames = listOf(dashboardName)
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteDashboards(dashboardsRequest)
        println("$dashboardName was successfully deleted.")
    }
}
```

- Untuk detail API, lihat DeleteDashboardsdi AWS SDK untuk referensi API Kotlin.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan CloudWatch dengan AWS SDK. Topik ini juga mencakup informasi tentang cara memulai dan detail versi-versi SDK sebelumnya.

Jelaskan riwayat CloudWatch alarm menggunakan AWS SDK

Contoh kode berikut menunjukkan cara mendeskripsikan riwayat CloudWatch alarm Amazon.

Contoh-contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan di dalam konteks. Anda dapat melihat tindakan ini dalam konteks pada contoh kode berikut:

- Mulai metrik, dasbor, dan alarm CloudWatch

.NET

AWS SDK for .NET

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /// <summary>
    /// Describe the history of an alarm for a number of days in the past.
    /// </summary>
    /// <param name="alarmName">The name of the alarm.</param>
    /// <param name="historyDays">The number of days in the past.</param>
    /// <returns>The list of alarm history data.</returns>
    public async Task<List<AlarmHistoryItem>> DescribeAlarmHistory(string
alarmName, int historyDays)
    {
        List<AlarmHistoryItem> alarmHistory = new List<AlarmHistoryItem>();
        var paginatedAlarmHistory =
_amazonCloudWatch.Paginators.DescribeAlarmHistory(
            new DescribeAlarmHistoryRequest()
            {
                AlarmName = alarmName,
                EndDateUtc = DateTime.UtcNow,
                HistoryItemType = HistoryItemType.StateUpdate,
                StartDateUtc = DateTime.UtcNow.AddDays(-historyDays)
            });

        await foreach (var data in paginatedAlarmHistory.AlarmHistoryItems)
        {
            alarmHistory.Add(data);
        }
        return alarmHistory;
    }
```

- Untuk detail API, lihat DescribeAlarmHistorydi Referensi AWS SDK for .NET API.

CLI

AWS CLI

Cara mengambil riwayat untuk sebuah alarm

Contoh berikut menggunakan `describe-alarm-history` perintah untuk mengambil riwayat untuk CloudWatch alarm Amazon bernama “myalarm”:

```
aws cloudwatch describe-alarm-history --alarm-name "myalarm" --history-item-type
 StateUpdate
```

Output:

```
{
    "AlarmHistoryItems": [
        {
            "Timestamp": "2014-04-09T18:59:06.442Z",
            "HistoryItemType": "StateUpdate",
            "AlarmName": "myalarm",
            "HistoryData": "{\"version\":\"1.0\",\"oldState\":{\"stateValue
\":\"ALARM\",\"stateReason\":\"testing purposes\"},\"newState\":{\"stateValue
\":\"OK\",\"stateReason\":\"Threshold Crossed: 2 datapoints were not
 greater than the threshold (70.0). The most recent datapoints: [38.958,
 40.292].\",\"stateReasonData\":{\"version\":\"1.0\",\"queryDate\":
\"2014-04-09T18:59:06.419+0000\",\"startDate\":\"2014-04-09T18:44:00.000+0000\",
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":[38.958,40.292],
\"threshold\":70.0}}",
            "HistorySummary": "Alarm updated from ALARM to OK"
        },
        {
            "Timestamp": "2014-04-09T18:59:05.805Z",
            "HistoryItemType": "StateUpdate",
            "AlarmName": "myalarm",
            "HistoryData": "{\"version\":\"1.0\",\"oldState\":{\"stateValue
\":\"OK\",\"stateReason\":\"Threshold Crossed: 2 datapoints were
 not greater than the threshold (70.0). The most recent datapoints:
 [38.839999999999996, 39.714].\",\"stateReasonData\":{\"version\":
```

```
\"1.0\",\"queryDate\":\"2014-03-11T22:45:41.569+0000\",\"startDate\":
\"2014-03-11T22:30:00.000+0000\",\"statistic\":\"Average\",\"period\":300,
\"recentDatapoints\":[38.839999999999996,39.714],\"threshold\":70.0}},\"newState
\":{\"stateValue\":\"ALARM\",\"stateReason\":\"testing purposes\"}}",
            "HistorySummary": "Alarm updated from OK to ALARM"
        }
    ]
}
```

- Untuk detail API, lihat DescribeAlarmHistorydi Referensi AWS CLI Perintah.

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
   public static void getAlarmHistory(CloudWatchClient cw, String fileName,
String date) {
       try {
           // Read values from the JSON file.
           JsonParser parser = new JsonFactory().createParser(new
File(fileName));
           com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
           String alarmName = rootNode.findValue("exampleAlarmName").asText();

           Instant start = Instant.parse(date);
           Instant endDate = Instant.now();
           DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
                   .startDate(start)
                   .endDate(endDate)
                   .alarmName(alarmName)
                   .historyItemType(HistoryItemType.ACTION)
                   .build();
```

```
            DescribeAlarmHistoryResponse response =
 cw.describeAlarmHistory(historyRequest);
            List<AlarmHistoryItem> historyItems = response.alarmHistoryItems();
            if (historyItems.isEmpty()) {
                System.out.println("No alarm history data found for " + alarmName
 + ".");
            } else {
                for (AlarmHistoryItem item : historyItems) {
                    System.out.println("History summary: " +
 item.historySummary());
                    System.out.println("Time stamp: " + item.timestamp());
                }
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Untuk detail API, lihat DescribeAlarmHistorydi Referensi AWS SDK for Java 2.x API.

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
suspend fun getAlarmHistory(fileName: String, date: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val start = Instant.parse(date)
    val endDateVal = Instant.now()

    val historyRequest = DescribeAlarmHistoryRequest {
```

```
        startDate = aws.smithy.kotlin.runtime.time.Instant(start)
        endDate = aws.smithy.kotlin.runtime.time.Instant(endDateVal)
        alarmName = alarmNameVal
        historyItemType = HistoryItemType.Action
    }

    CloudWatchClient { credentialsProvider = EnvironmentCredentialsProvider();
 region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarmHistory(historyRequest)
        val historyItems = response.alarmHistoryItems
        if (historyItems != null) {
            if (historyItems.isEmpty()) {
                println("No alarm history data found for $alarmNameVal.")
            } else {
                for (item in historyItems) {
                    println("History summary ${item.historySummary}")
                    println("Time stamp: ${item.timestamp}")
                }
            }
        }
    }
}
```

- Untuk detail API, lihat DescribeAlarmHistorydi AWS SDK untuk referensi API Kotlin.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan CloudWatch dengan AWS SDK. Topik ini juga mencakup informasi tentang cara memulai dan detail versi-versi SDK sebelumnya.

Jelaskan CloudWatch alarm menggunakan SDK AWS

Contoh kode berikut menunjukkan cara mendeskripsikan CloudWatch alarm Amazon.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Memulai menggunakan alarm
- Mulai metrik, dasbor, dan alarm CloudWatch

.NET

AWS SDK for .NET

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /// <summary>
    /// Describe the current alarms, optionally filtered by state.
    /// </summary>
    /// <param name="stateValue">Optional filter for alarm state.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarms(StateValue? stateValue =
null)
    {
        List<MetricAlarm> alarms = new List<MetricAlarm>();
        var paginatedDescribeAlarms =
_amazonCloudWatch.Paginators.DescribeAlarms(
            new DescribeAlarmsRequest()
            {
                StateValue = stateValue
            });

        await foreach (var data in paginatedDescribeAlarms.MetricAlarms)
        {
            alarms.Add(data);
        }
        return alarms;
    }
```

- Untuk detail API, lihat DescribeAlarmsdi Referensi AWS SDK for .NET API.

CLI

AWS CLI

Cara membuat daftar informasi tentang sebuah alarm

Contoh berikut menggunakan perintah `describe-alarms` untuk memberikan informasi tentang alarm yang bernama "myalarm":

```
aws cloudwatch describe-alarms --alarm-names "myalarm"
```

Output:

```
{
    "MetricAlarms": [
        {
            "EvaluationPeriods": 2,
            "AlarmArn": "arn:aws:cloudwatch:us-
east-1:123456789012:alarm:myalarm",
            "StateUpdatedTimestamp": "2014-04-09T18:59:06.442Z",
            "AlarmConfigurationUpdatedTimestamp": "2012-12-27T00:49:54.032Z",
            "ComparisonOperator": "GreaterThanThreshold",
            "AlarmActions": [
                "arn:aws:sns:us-east-1:123456789012:myHighCpuAlarm"
            ],
            "Namespace": "AWS/EC2",
            "AlarmDescription": "CPU usage exceeds 70 percent",
            "StateReasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2014-04-09T18:59:06.419+0000\",\"startDate\":\"2014-04-09T18:44:00.000+0000\",
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":[38.958,40.292],
\"threshold\":70.0}",
            "Period": 300,
            "StateValue": "OK",
            "Threshold": 70.0,
            "AlarmName": "myalarm",
            "Dimensions": [
                {
                    "Name": "InstanceId",
                    "Value": "i-0c986c72"
                }
            ],
            "Statistic": "Average",
            "StateReason": "Threshold Crossed: 2 datapoints were not greater than
 the threshold (70.0). The most recent datapoints: [38.958, 40.292].",
            "InsufficientDataActions": [],
            "OKActions": [],
            "ActionsEnabled": true,
            "MetricName": "CPUUtilization"
        }
```

```
    ]
}
```

- Untuk detail API, lihat DescribeAlarmsdi Referensi AWS CLI Perintah.

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    public static void describeAlarms(CloudWatchClient cw) {
        try {
            List<AlarmType> typeList = new ArrayList<>();
            typeList.add(AlarmType.METRIC_ALARM);

            DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
                    .alarmTypes(typeList)
                    .maxRecords(10)
                    .build();

            DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
            List<MetricAlarm> alarmList = response.metricAlarms();
            for (MetricAlarm alarm : alarmList) {
                System.out.println("Alarm name: " + alarm.alarmName());
                System.out.println("Alarm description: " +
alarm.alarmDescription());
            }
        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Untuk detail API, lihat DescribeAlarmsdi Referensi AWS SDK for Java 2.x API.

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
suspend fun describeAlarms() {
    val typeList = ArrayList<AlarmType>()
    typeList.add(AlarmType.MetricAlarm)
    val alarmsRequest = DescribeAlarmsRequest {
        alarmTypes = typeList
        maxRecords = 10
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarms(alarmsRequest)
        response.metricAlarms?.forEach { alarm ->
            println("Alarm name: ${alarm.alarmName}")
            println("Alarm description: ${alarm.alarmDescription}")
        }
    }
}
```

- Untuk detail API, lihat DescribeAlarmsdi AWS SDK untuk referensi API Kotlin.

Ruby

SDK for Ruby

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
require "aws-sdk-cloudwatch"

# Lists the names of available Amazon CloudWatch alarms.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @example
#   list_alarms(Aws::CloudWatch::Client.new(region: 'us-east-1'))
def list_alarms(cloudwatch_client)
  response = cloudwatch_client.describe_alarms
  if response.metric_alarms.count.positive?
    response.metric_alarms.each do |alarm|
      puts alarm.alarm_name
    end
  else
    puts "No alarms found."
  end
rescue StandardError => e
  puts "Error getting information about alarms: #{e.message}"
end
```

- Untuk detail API, lihat DescribeAlarmsdi Referensi AWS SDK for Ruby API.

SAP ABAP

SDK untuk SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    TRY.
        oo_result = lo_cwt->describealarms(                " oo_result is
 returned for testing purposes. "
          it_alarmnames = it_alarm_names
        ).
        MESSAGE 'Alarms retrieved.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
```

```
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Untuk detail API, lihat DescribeAlarmsdi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan CloudWatch dengan AWS SDK. Topik ini juga mencakup informasi tentang cara memulai dan detail versi-versi SDK sebelumnya.

Jelaskan CloudWatch alarm untuk metrik menggunakan SDK AWS

Contoh kode berikut menunjukkan cara mendeskripsikan CloudWatch alarm Amazon untuk metrik.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Mulai metrik, dasbor, dan alarm CloudWatch
- Mengelola metrik dan alarm

.NET

AWS SDK for .NET

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /// <summary>
    /// Describe the current alarms for a specific metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarmsForMetric(string
metricNamespace, string metricName)
```

```
    {
        var alarmsResult = await _amazonCloudWatch.DescribeAlarmsForMetricAsync(
            new DescribeAlarmsForMetricRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName
            });

        return alarmsResult.MetricAlarms;
    }
```

- Untuk detail API, lihat DescribeAlarmsForMetricdi Referensi AWS SDK for .NET API.

C++

SDK for C++

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di Repositori Contoh Kode AWS.

Sertakan file-file yang diperlukan.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DescribeAlarmsRequest.h>
#include <aws/monitoring/model/DescribeAlarmsResult.h>
#include <iomanip>
#include <iostream>
```

Mendeskripsikan alarm.

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::DescribeAlarmsRequest request;
        request.SetMaxRecords(1);

        bool done = false;
```

```
        bool header = false;
        while (!done)
        {
            auto outcome = cw.DescribeAlarms(request);
            if (!outcome.IsSuccess())
            {
                std::cout << "Failed to describe CloudWatch alarms:" <<
                    outcome.GetError().GetMessage() << std::endl;
                break;
            }

            if (!header)
            {
                std::cout << std::left <<
                    std::setw(32) << "Name" <<
                    std::setw(64) << "Arn" <<
                    std::setw(64) << "Description" <<
                    std::setw(20) << "LastUpdated" <<
                    std::endl;
                header = true;
            }

            const auto &alarms = outcome.GetResult().GetMetricAlarms();
            for (const auto &alarm : alarms)
            {
                std::cout << std::left <<
                    std::setw(32) << alarm.GetAlarmName() <<
                    std::setw(64) << alarm.GetAlarmArn() <<
                    std::setw(64) << alarm.GetAlarmDescription() <<
                    std::setw(20) <<
                    alarm.GetAlarmConfigurationUpdatedTimestamp().ToGmtString(
                        SIMPLE_DATE_FORMAT_STR) <<
                    std::endl;
            }

            const auto &next_token = outcome.GetResult().GetNextToken();
            request.SetNextToken(next_token);
            done = next_token.empty();
        }
```

- Untuk detail API, lihat DescribeAlarmsForMetricdi Referensi AWS SDK for C++ API.

CLI

AWS CLI

Untuk menampilkan informasi tentang alarm-alarm yang terkait dengan sebuah metrik

Contoh berikut menggunakan perintah `describe-alarms-for-metric` untuk menampilkan informasi tentang setiap alarm yang terkait dengan metrik CPUUutilization Amazon EC2 dan instans dengan ID i-0c986c72. :

```
aws cloudwatch describe-alarms-for-metric --metric-name CPUUtilization --
namespace AWS/EC2 --dimensions Name=InstanceId,Value=i-0c986c72
```

Output:

```
{
    "MetricAlarms": [
        {
            "EvaluationPeriods": 10,
            "AlarmArn": "arn:aws:cloudwatch:us-
east-1:111122223333:alarm:myHighCpuAlarm2",
            "StateUpdatedTimestamp": "2013-10-30T03:03:51.479Z",
            "AlarmConfigurationUpdatedTimestamp": "2013-10-30T03:03:50.865Z",
            "ComparisonOperator": "GreaterThanOrEqualToThreshold",
            "AlarmActions": [
                "arn:aws:sns:us-east-1:111122223333:NotifyMe"
            ],
            "Namespace": "AWS/EC2",
            "AlarmDescription": "CPU usage exceeds 70 percent",
            "StateReasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2013-10-30T03:03:51.479+0000\",\"startDate\":\"2013-10-30T02:08:00.000+0000\",
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":
[40.698,39.612,42.432,39.796,38.816,42.28,42.854,40.088,40.760000000000005,41.316],
\"threshold\":70.0}",
            "Period": 300,
            "StateValue": "OK",
            "Threshold": 70.0,
            "AlarmName": "myHighCpuAlarm2",
            "Dimensions": [
                {
                    "Name": "InstanceId",
                    "Value": "i-0c986c72"
                }
```

```
            ],
            "Statistic": "Average",
            "StateReason": "Threshold Crossed: 10 datapoints were not
 greater than or equal to the threshold (70.0). The most recent datapoints:
 [40.760000000000005, 41.316].",
            "InsufficientDataActions": [],
            "OKActions": [],
            "ActionsEnabled": true,
            "MetricName": "CPUUtilization"
        },
        {
            "EvaluationPeriods": 2,
            "AlarmArn": "arn:aws:cloudwatch:us-
east-1:111122223333:alarm:myHighCpuAlarm",
            "StateUpdatedTimestamp": "2014-04-09T18:59:06.442Z",
            "AlarmConfigurationUpdatedTimestamp": "2014-04-09T22:26:05.958Z",
            "ComparisonOperator": "GreaterThanThreshold",
            "AlarmActions": [
                "arn:aws:sns:us-east-1:111122223333:HighCPUAlarm"
            ],
            "Namespace": "AWS/EC2",
            "AlarmDescription": "CPU usage exceeds 70 percent",
            "StateReasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2014-04-09T18:59:06.419+0000\",\"startDate\":\"2014-04-09T18:44:00.000+0000\",
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":[38.958,40.292],
\"threshold\":70.0}",
            "Period": 300,
            "StateValue": "OK",
            "Threshold": 70.0,
            "AlarmName": "myHighCpuAlarm",
            "Dimensions": [
                {
                    "Name": "InstanceId",
                    "Value": "i-0c986c72"
                }
            ],
            "Statistic": "Average",
            "StateReason": "Threshold Crossed: 2 datapoints were not greater than
 the threshold (70.0). The most recent datapoints: [38.958, 40.292].",
            "InsufficientDataActions": [],
            "OKActions": [],
            "ActionsEnabled": false,
            "MetricName": "CPUUtilization"
        }
```

```
    ]
}
```

- Untuk detail API, lihat DescribeAlarmsForMetricdi Referensi AWS CLI Perintah.

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
   public static void checkForMetricAlarm(CloudWatchClient cw, String fileName)
{
       try {
           // Read values from the JSON file.
           JsonParser parser = new JsonFactory().createParser(new
File(fileName));
           com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
           String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
           String customMetricName =
rootNode.findValue("customMetricName").asText();
           boolean hasAlarm = false;
           int retries = 10;

           DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
                   .metricName(customMetricName)
                   .namespace(customMetricNamespace)
                   .build();

           while (!hasAlarm && retries > 0) {
               DescribeAlarmsForMetricResponse response =
cw.describeAlarmsForMetric(metricRequest);
               hasAlarm = response.hasMetricAlarms();
               retries--;
               Thread.sleep(20000);
```

```
                System.out.println(".");
            }
            if (!hasAlarm)
                System.out.println("No Alarm state found for " + customMetricName
 + " after 10 retries.");
            else
                System.out.println("Alarm state found for " + customMetricName +
 ".");

        } catch (CloudWatchException | IOException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Untuk detail API, lihat DescribeAlarmsForMetricdi Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di Repositori Contoh Kode AWS.

Mengimpor modul SDK dan klien dan memanggil API.

```
import { DescribeAlarmsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DescribeAlarmsCommand({
    AlarmNames: [process.env.CLOUDWATCH_ALARM_NAME], // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
    return await client.send(command);
  } catch (err) {
```

```
    console.error(err);
  }
};

export default run();
```

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat DescribeAlarmsForMetricdi Referensi AWS SDK for JavaScript API.

SDK untuk JavaScript (v2)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

cw.describeAlarms({ StateValue: "INSUFFICIENT_DATA" }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    // List the names of all current alarms in the console
    data.MetricAlarms.forEach(function (item, index, array) {
      console.log(item.AlarmName);
    });
```

```
  }
});
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat DescribeAlarmsForMetricdi Referensi AWS SDK for JavaScript API.

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
suspend fun checkForMetricAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    var hasAlarm = false
    var retries = 10

    val metricRequest = DescribeAlarmsForMetricRequest {
        metricName = customMetricName
        namespace = customMetricNamespace
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        while (!hasAlarm && retries > 0) {
            val response = cwClient.describeAlarmsForMetric(metricRequest)
            if (response.metricAlarms?.count()!! > 0) {
                hasAlarm = true
            }
            retries--
            delay(20000)
            println(".")
        }
```

```
        if (!hasAlarm) println("No Alarm state found for $customMetricName after
 10 retries.") else println("Alarm state found for $customMetricName.")
    }
}
```

- Untuk detail API, lihat DescribeAlarmsForMetricdi AWS SDK untuk referensi API Kotlin.

Python

SDK for Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def get_metric_alarms(self, metric_namespace, metric_name):
        """
        Gets the alarms that are currently watching the specified metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :returns: An iterator that yields the alarms.
        """
        metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
        alarm_iter = metric.alarms.all()
        logger.info("Got alarms for metric %s.%s.", metric_namespace,
 metric_name)
        return alarm_iter
```

- Untuk detail API, lihat DescribeAlarmsForMetricdi AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK for Ruby

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @example
#   describe_metric_alarms(Aws::CloudWatch::Client.new(region: 'us-east-1'))
def describe_metric_alarms(cloudwatch_client)
  response = cloudwatch_client.describe_alarms

  if response.metric_alarms.count.positive?
    response.metric_alarms.each do |alarm|
      puts "-" * 16
      puts "Name:           " + alarm.alarm_name
      puts "State value:    " + alarm.state_value
      puts "State reason:   " + alarm.state_reason
      puts "Metric:         " + alarm.metric_name
      puts "Namespace:      " + alarm.namespace
      puts "Statistic:      " + alarm.statistic
      puts "Period:         " + alarm.period.to_s
      puts "Unit:           " + alarm.unit.to_s
      puts "Eval. periods:  " + alarm.evaluation_periods.to_s
      puts "Threshold:      " + alarm.threshold.to_s
      puts "Comp. operator: " + alarm.comparison_operator

      if alarm.key?(:ok_actions) && alarm.ok_actions.count.positive?
        puts "OK actions:"
```

```
        alarm.ok_actions.each do |a|
          puts "  " + a
        end
      end

      if alarm.key?(:alarm_actions) && alarm.alarm_actions.count.positive?
        puts "Alarm actions:"
        alarm.alarm_actions.each do |a|
          puts "  " + a
        end
      end

      if alarm.key?(:insufficient_data_actions) &&
          alarm.insufficient_data_actions.count.positive?
        puts "Insufficient data actions:"
        alarm.insufficient_data_actions.each do |a|
          puts "  " + a
        end
      end

      puts "Dimensions:"
      if alarm.key?(:dimensions) && alarm.dimensions.count.positive?
        alarm.dimensions.each do |d|
          puts "  Name: " + d.name + ", Value: " + d.value
        end
      else
        puts "  None for this alarm."
      end
    end
  else
    puts "No alarms found."
  end
rescue StandardError => e
  puts "Error getting information about alarms: #{e.message}"
end

# Example usage:
def run_me
  region = ""

  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage:   ruby cw-ruby-example-show-alarms.rb REGION"
    puts "Example: ruby cw-ruby-example-show-alarms.rb us-east-1"
```

```
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    region = "us-east-1"
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)
  puts "Available alarms:"
  describe_metric_alarms(cloudwatch_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Untuk detail API, lihat DescribeAlarmsForMetricdi Referensi AWS SDK for Ruby API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan CloudWatch dengan AWS SDK. Topik ini juga mencakup informasi tentang cara memulai dan detail versi-versi SDK sebelumnya.

Jelaskan detektor CloudWatch anomali menggunakan SDK AWS

Contoh kode berikut menunjukkan cara mendeskripsikan detektor CloudWatch anomali Amazon.

Contoh-contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan di dalam konteks. Anda dapat melihat tindakan ini dalam konteks pada contoh kode berikut:

- Mulai metrik, dasbor, dan alarm CloudWatch

.NET

AWS SDK for .NET

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /// <summary>
    /// Describe anomaly detectors for a metric and namespace.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The metric of the anomaly detectors.</param>
    /// <returns>The list of detectors.</returns>
    public async Task<List<AnomalyDetector>> DescribeAnomalyDetectors(string
metricNamespace, string metricName)
    {
        List<AnomalyDetector> detectors = new List<AnomalyDetector>();
        var paginatedDescribeAnomalyDetectors =
_amazonCloudWatch.Paginators.DescribeAnomalyDetectors(
            new DescribeAnomalyDetectorsRequest()
            {
                MetricName = metricName,
                Namespace = metricNamespace
            });

        await foreach (var data in
paginatedDescribeAnomalyDetectors.AnomalyDetectors)
        {
            detectors.Add(data);
        }

        return detectors;
    }
```

- Untuk detail API, lihat DescribeAnomalyDetectorsdi Referensi AWS SDK for .NET API.

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
   public static void describeAnomalyDetectors(CloudWatchClient cw, String
fileName) {
       try {
           // Read values from the JSON file.
           JsonParser parser = new JsonFactory().createParser(new
File(fileName));
           com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
           String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
           String customMetricName =
rootNode.findValue("customMetricName").asText();
           DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
                   .maxResults(10)
                   .metricName(customMetricName)
                   .namespace(customMetricNamespace)
                   .build();

           DescribeAnomalyDetectorsResponse response =
cw.describeAnomalyDetectors(detectorsRequest);
           List<AnomalyDetector> anomalyDetectorList =
response.anomalyDetectors();
           for (AnomalyDetector detector : anomalyDetectorList) {
               System.out.println("Metric name: " +
detector.singleMetricAnomalyDetector().metricName());
               System.out.println("State: " + detector.stateValue());
           }

       } catch (CloudWatchException | IOException e) {
           System.err.println(e.getMessage());
           System.exit(1);
```

```
        }
    }
```

- Untuk detail API, lihat DescribeAnomalyDetectorsdi Referensi AWS SDK for Java 2.x API.

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
suspend fun describeAnomalyDetectors(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val detectorsRequest = DescribeAnomalyDetectorsRequest {
        maxResults = 10
        metricName = customMetricName
        namespace = customMetricNamespace
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAnomalyDetectors(detectorsRequest)
        response.anomalyDetectors?.forEach { detector ->
            println("Metric name:
 ${detector.singleMetricAnomalyDetector?.metricName}")
            println("State: ${detector.stateValue}")
        }
    }
}
```

- Untuk detail API, lihat DescribeAnomalyDetectorsdi AWS SDK untuk referensi API Kotlin.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan CloudWatch dengan AWS SDK. Topik ini juga mencakup informasi tentang cara memulai dan detail versi-versi SDK sebelumnya.

Nonaktifkan tindakan CloudWatch alarm menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menonaktifkan tindakan CloudWatch alarm Amazon.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Memulai menggunakan alarm
- Mengelola metrik dan alarm

.NET

AWS SDK for .NET

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /// <summary>
    /// Disable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DisableAlarmActions(List<string> alarmNames)
    {
        var disableAlarmActionsResult = await
_amazonCloudWatch.DisableAlarmActionsAsync(
            new DisableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return disableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Untuk detail API, lihat DisableAlarmActionsdi Referensi AWS SDK for .NET API.

C++

SDK for C++

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di Repositori Contoh Kode AWS.

Sertakan file-file yang diperlukan.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DisableAlarmActionsRequest.h>
#include <iostream>
```

Menonaktifkan tindakan alarm.

```
        Aws::CloudWatch::CloudWatchClient cw;

        Aws::CloudWatch::Model::DisableAlarmActionsRequest
 disableAlarmActionsRequest;
        disableAlarmActionsRequest.AddAlarmNames(alarm_name);

        auto disableAlarmActionsOutcome =
 cw.DisableAlarmActions(disableAlarmActionsRequest);
        if (!disableAlarmActionsOutcome.IsSuccess())
        {
            std::cout << "Failed to disable actions for alarm " << alarm_name <<
                ": " << disableAlarmActionsOutcome.GetError().GetMessage() <<
                std::endl;
        }
        else
        {
            std::cout << "Successfully disabled actions for alarm " <<
```

```
                alarm_name << std::endl;
        }
```

- Untuk detail API, lihat DisableAlarmActionsdi Referensi AWS SDK for C++ API.

CLI

AWS CLI

Cara menonaktifkan tindakan untuk sebuah alarm

Contoh berikut menggunakan perintah `disable-alarm-actions` untuk menonaktifkan semua tindakan untuk alarm bernama myalarm.:

```
aws cloudwatch disable-alarm-actions --alarm-names myalarm
```

Perintah ini akan kembali ke prompt jika berhasil.

- Untuk detail API, lihat DisableAlarmActionsdi Referensi AWS CLI Perintah.

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import
 software.amazon.awssdk.services.cloudwatch.model.DisableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DisableAlarmActions {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <alarmName>

                Where:
                  alarmName - An alarm name to disable (for example, MyAlarm).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarmName = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        disableActions(cw, alarmName);
        cw.close();
    }

    public static void disableActions(CloudWatchClient cw, String alarmName) {
        try {
            DisableAlarmActionsRequest request =
 DisableAlarmActionsRequest.builder()
                    .alarmNames(alarmName)
                    .build();

            cw.disableAlarmActions(request);
            System.out.printf("Successfully disabled actions on alarm %s",
 alarmName);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat DisableAlarmActionsdi Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di Repositori Contoh Kode AWS.

Mengimpor modul SDK dan klien dan memanggil API.

```
import { DisableAlarmActionsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DisableAlarmActionsCommand({
    AlarmNames: process.env.CLOUDWATCH_ALARM_NAME, // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat DisableAlarmActionsdi Referensi AWS SDK for JavaScript API.

SDK untuk JavaScript (v2)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di Repositori Contoh Kode AWS.

Mengimpor modul SDK dan klien dan memanggil API.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

cw.disableAlarmActions(
  { AlarmNames: ["Web_Server_CPU_Utilization"] },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
  }
);
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat DisableAlarmActionsdi Referensi AWS SDK for JavaScript API.

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
suspend fun disableActions(alarmName: String) {

    val request = DisableAlarmActionsRequest {
        alarmNames = listOf(alarmName)
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.disableAlarmActions(request)
        println("Successfully disabled actions on alarm $alarmName")
    }
}
```

- Untuk detail API, lihat DisableAlarmActionsdi AWS SDK untuk referensi API Kotlin.

Python

SDK for Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
```

```
        """
        self.cloudwatch_resource = cloudwatch_resource


    def enable_alarm_actions(self, alarm_name, enable):
        """
        Enables or disables actions on the specified alarm. Alarm actions can be
        used to send notifications or automate responses when an alarm enters a
        particular state.

        :param alarm_name: The name of the alarm.
        :param enable: When True, actions are enabled for the alarm. Otherwise,
they
                       disabled.
        """
        try:
            alarm = self.cloudwatch_resource.Alarm(alarm_name)
            if enable:
                alarm.enable_actions()
            else:
                alarm.disable_actions()
            logger.info(
                "%s actions for alarm %s.",
                "Enabled" if enable else "Disabled",
                alarm_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't %s actions alarm %s.",
                "enable" if enable else "disable",
                alarm_name,
            )
            raise
```

- Untuk detail API, lihat DisableAlarmActionsdi AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK for Ruby

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
# Disables an alarm in Amazon CloudWatch.
#
# Prerequisites.
#
# - The alarm to disable.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm to disable.
# @return [Boolean] true if the alarm was disabled; otherwise, false.
# @example
#   exit 1 unless alarm_actions_disabled?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket'
#   )
def alarm_actions_disabled?(cloudwatch_client, alarm_name)
  cloudwatch_client.disable_alarm_actions(alarm_names: [alarm_name])
  return true
rescue StandardError => e
  puts "Error disabling alarm actions: #{e.message}"
  return false
end

# Example usage:
def run_me
  alarm_name = "ObjectsInBucket"
  alarm_description = "Objects exist in this bucket for more than 1 day."
  metric_name = "NumberOfObjects"
  # Notify this Amazon Simple Notification Service (Amazon SNS) topic when
  # the alarm transitions to the ALARM state.
  alarm_actions = ["arn:aws:sns:us-
east-1:111111111111:Default_CloudWatch_Alarms_Topic"]
```

```
namespace = "AWS/S3"
statistic = "Average"
dimensions = [
  {
    name: "BucketName",
    value: "doc-example-bucket"
  },
  {
    name: "StorageType",
    value: "AllStorageTypes"
  }
]
period = 86_400 # Daily (24 hours * 60 minutes * 60 seconds = 86400 seconds).
unit = "Count"
evaluation_periods = 1 # More than one day.
threshold = 1 # One object.
comparison_operator = "GreaterThanThreshold" # More than one object.
# Replace us-west-2 with the AWS Region you're using for Amazon CloudWatch.
region = "us-east-1"

cloudwatch_client = Aws::CloudWatch::Client.new(region: region)

if alarm_created_or_updated?(
  cloudwatch_client,
  alarm_name,
  alarm_description,
  metric_name,
  alarm_actions,
  namespace,
  statistic,
  dimensions,
  period,
  unit,
  evaluation_periods,
  threshold,
  comparison_operator
)
  puts "Alarm '#{alarm_name}' created or updated."
else
  puts "Could not create or update alarm '#{alarm_name}'."
end

if alarm_actions_disabled?(cloudwatch_client, alarm_name)
  puts "Alarm '#{alarm_name}' disabled."
```

```
  else
    puts "Could not disable alarm '#{alarm_name}'."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Untuk detail API, lihat DisableAlarmActionsdi Referensi AWS SDK for Ruby API.

SAP ABAP

SDK untuk SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    "Disables actions on the specified alarm. "
    TRY.
        lo_cwt->disablealarmactions(
          it_alarmnames = it_alarm_names
        ).
        MESSAGE 'Alarm actions disabled.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Untuk detail API, lihat DisableAlarmActionsdi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan CloudWatch dengan AWS SDK. Topik ini juga mencakup informasi tentang cara memulai dan detail versi-versi SDK sebelumnya.

Aktifkan tindakan CloudWatch alarm menggunakan AWS SDK

Contoh kode berikut menunjukkan cara mengaktifkan tindakan CloudWatch alarm Amazon.

Contoh-contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan di dalam konteks. Anda dapat melihat tindakan ini dalam konteks pada contoh kode berikut:

- Mengelola metrik dan alarm

.NET

AWS SDK for .NET

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /// <summary>
    /// Enable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableAlarmActions(List<string> alarmNames)
    {
        var enableAlarmActionsResult = await
_amazonCloudWatch.EnableAlarmActionsAsync(
            new EnableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return enableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Untuk detail API, lihat EnableAlarmActionsdi Referensi AWS SDK for .NET API.

C++

SDK for C++

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di Repositori Contoh Kode AWS.

Sertakan file-file yang diperlukan.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/EnableAlarmActionsRequest.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

Mengaktifkan tindakan alarm.

```
    Aws::CloudWatch::CloudWatchClient cw;
    Aws::CloudWatch::Model::PutMetricAlarmRequest request;
    request.SetAlarmName(alarm_name);
    request.SetComparisonOperator(
        Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
    request.SetEvaluationPeriods(1);
    request.SetMetricName("CPUUtilization");
    request.SetNamespace("AWS/EC2");
    request.SetPeriod(60);
    request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
    request.SetThreshold(70.0);
    request.SetActionsEnabled(false);
    request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
    request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);
    request.AddAlarmActions(actionArn);

    Aws::CloudWatch::Model::Dimension dimension;
    dimension.SetName("InstanceId");
    dimension.SetValue(instanceId);
    request.AddDimensions(dimension);
```

```
    auto outcome = cw.PutMetricAlarm(request);
    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to create CloudWatch alarm:" <<
            outcome.GetError().GetMessage() << std::endl;
        return;
    }

    Aws::CloudWatch::Model::EnableAlarmActionsRequest enable_request;
    enable_request.AddAlarmNames(alarm_name);

    auto enable_outcome = cw.EnableAlarmActions(enable_request);
    if (!enable_outcome.IsSuccess())
    {
        std::cout << "Failed to enable alarm actions:" <<
            enable_outcome.GetError().GetMessage() << std::endl;
        return;
    }

    std::cout << "Successfully created alarm " << alarm_name <<
        " and enabled actions on it." << std::endl;
```

- Untuk detail API, lihat EnableAlarmActionsdi Referensi AWS SDK for C++ API.

CLI

AWS CLI

Cara mengaktifkan semua tindakan untuk sebuah alarm

Contoh berikut menggunakan perintah `enable-alarm-actions` untuk mengaktifkan semua tindakan untuk alarm bernama myalarm.:

```
aws cloudwatch enable-alarm-actions --alarm-names myalarm
```

Perintah ini akan kembali ke prompt jika berhasil.

- Untuk detail API, lihat EnableAlarmActionsdi Referensi AWS CLI Perintah.

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import
 software.amazon.awssdk.services.cloudwatch.model.EnableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class EnableAlarmActions {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <alarmName>

                Where:
                  alarmName - An alarm name to enable (for example, MyAlarm).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarm = args[0];
```

```
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        enableActions(cw, alarm);
        cw.close();
    }

    public static void enableActions(CloudWatchClient cw, String alarm) {
        try {
            EnableAlarmActionsRequest request =
 EnableAlarmActionsRequest.builder()
                    .alarmNames(alarm)
                    .build();

            cw.enableAlarmActions(request);
            System.out.printf("Successfully enabled actions on alarm %s", alarm);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat EnableAlarmActionsdi Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di Repositori Contoh Kode AWS.

Mengimpor modul SDK dan klien dan memanggil API.

```
import { EnableAlarmActionsCommand } from "@aws-sdk/client-cloudwatch";
```

```
import { client } from "../libs/client.js";

const run = async () => {
  const command = new EnableAlarmActionsCommand({
    AlarmNames: [process.env.CLOUDWATCH_ALARM_NAME], // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat EnableAlarmActionsdi Referensi AWS SDK for JavaScript API.

SDK untuk JavaScript (v2)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di Repositori Contoh Kode AWS.

Mengimpor modul SDK dan klien dan memanggil API.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });
```

```
// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  AlarmName: "Web_Server_CPU_Utilization",
  ComparisonOperator: "GreaterThanThreshold",
  EvaluationPeriods: 1,
  MetricName: "CPUUtilization",
  Namespace: "AWS/EC2",
  Period: 60,
  Statistic: "Average",
  Threshold: 70.0,
  ActionsEnabled: true,
  AlarmActions: ["ACTION_ARN"],
  AlarmDescription: "Alarm when server CPU exceeds 70%",
  Dimensions: [
    {
      Name: "InstanceId",
      Value: "INSTANCE_ID",
    },
  ],
  Unit: "Percent",
};

cw.putMetricAlarm(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Alarm action added", data);
    var paramsEnableAlarmAction = {
      AlarmNames: [params.AlarmName],
    };
    cw.enableAlarmActions(paramsEnableAlarmAction, function (err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Alarm action enabled", data);
      }
    });
  }
});
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat EnableAlarmActionsdi Referensi AWS SDK for JavaScript API.

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
suspend fun enableActions(alarm: String) {

    val request = EnableAlarmActionsRequest {
        alarmNames = listOf(alarm)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.enableAlarmActions(request)
        println("Successfully enabled actions on alarm $alarm")
    }
}
```

- Untuk detail API, lihat EnableAlarmActionsdi AWS SDK untuk referensi API Kotlin.

Python

SDK for Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
class CloudWatchWrapper:
```

```
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def enable_alarm_actions(self, alarm_name, enable):
        """
        Enables or disables actions on the specified alarm. Alarm actions can be
        used to send notifications or automate responses when an alarm enters a
        particular state.

        :param alarm_name: The name of the alarm.
        :param enable: When True, actions are enabled for the alarm. Otherwise,
they
                       disabled.
        """
        try:
            alarm = self.cloudwatch_resource.Alarm(alarm_name)
            if enable:
                alarm.enable_actions()
            else:
                alarm.disable_actions()
            logger.info(
                "%s actions for alarm %s.",
                "Enabled" if enable else "Disabled",
                alarm_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't %s actions alarm %s.",
                "enable" if enable else "disable",
                alarm_name,
            )
            raise
```

- Untuk detail API, lihat EnableAlarmActionsdi AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK untuk SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    "Enable actions on the specified alarm."
    TRY.
        lo_cwt->enablealarmactions(
          it_alarmnames = it_alarm_names
        ).
        MESSAGE 'Alarm actions enabled.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Untuk detail API, lihat EnableAlarmActionsdi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan CloudWatch dengan AWS SDK. Topik ini juga mencakup informasi tentang cara memulai dan detail versi-versi SDK sebelumnya.

Dapatkan gambar data CloudWatch metrik

Contoh kode berikut menunjukkan cara mendapatkan gambar data CloudWatch metrik Amazon.

Contoh-contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan di dalam konteks. Anda dapat melihat tindakan ini dalam konteks pada contoh kode berikut:

- Mulai metrik, dasbor, dan alarm CloudWatch

.NET

AWS SDK for .NET

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /// <summary>
    /// Get an image for a metric graphed over time.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metric">The name of the metric.</param>
    /// <param name="stat">The name of the stat to chart.</param>
    /// <param name="period">The period to use for the chart.</param>
    /// <returns>A memory stream for the chart image.</returns>
    public async Task<MemoryStream> GetTimeSeriesMetricImage(string
metricNamespace, string metric, string stat, int period)
    {
        var metricImageWidget = new
        {
            title = "Example Metric Graph",
            view = "timeSeries",
            stacked = false,
            period = period,
            width = 1400,
            height = 600,
            metrics = new List<List<object>>
                { new() { metricNamespace, metric, new { stat } } }
        };

        var metricImageWidgetString =
JsonSerializer.Serialize(metricImageWidget);
        var imageResponse = await _amazonCloudWatch.GetMetricWidgetImageAsync(
            new GetMetricWidgetImageRequest()
            {
                MetricWidget = metricImageWidgetString
            });

        return imageResponse.MetricWidgetImage;
```

```
    }

    /// <summary>
    /// Save a metric image to a file.
    /// </summary>
    /// <param name="memoryStream">The MemoryStream for the metric image.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The path to the file.</returns>
    public string SaveMetricImage(MemoryStream memoryStream, string metricName)
    {
        var metricFileName = $"{metricName}_{DateTime.Now.Ticks}.png";
        using var sr = new StreamReader(memoryStream);
        // Writes the memory stream to a file.
        File.WriteAllBytes(metricFileName, memoryStream.ToArray());
        var filePath = Path.Join(AppDomain.CurrentDomain.BaseDirectory,
            metricFileName);
        return filePath;
    }
```

- Untuk detail API, lihat GetMetricWidgetImagedi Referensi AWS SDK for .NET API.

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
   public static void getAndOpenMetricImage(CloudWatchClient cw, String
fileName) {
       System.out.println("Getting Image data for custom metric.");
       try {
           String myJSON = "{\n" +
                   "  \"title\": \"Example Metric Graph\",\n" +
                   "  \"view\": \"timeSeries\",\n" +
                   "  \"stacked \": false,\n" +
                   "  \"period\": 10,\n" +
                   "  \"width\": 1400,\n" +
```

```
                    "  \"height\": 600,\n" +
                    "  \"metrics\": [\n" +
                    "    [\n" +
                    "      \"AWS/Billing\",\n" +
                    "      \"EstimatedCharges\",\n" +
                    "      \"Currency\",\n" +
                    "      \"USD\"\n" +
                    "    ]\n" +
                    "  ]\n" +
                    "}";

            GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
                    .metricWidget(myJSON)
                    .build();

            GetMetricWidgetImageResponse response =
cw.getMetricWidgetImage(imageRequest);
            SdkBytes sdkBytes = response.metricWidgetImage();
            byte[] bytes = sdkBytes.asByteArray();
            File outputFile = new File(fileName);
            try (FileOutputStream outputStream = new
FileOutputStream(outputFile)) {
                outputStream.write(bytes);
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Untuk detail API, lihat GetMetricWidgetImagedi Referensi AWS SDK for Java 2.x API.

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
suspend fun getAndOpenMetricImage(fileName: String) {
    println("Getting Image data for custom metric.")
    val myJSON = """{
        "title": "Example Metric Graph",
        "view": "timeSeries",
        "stacked ": false,
        "period": 10,
        "width": 1400,
        "height": 600,
        "metrics": [
            [
            "AWS/Billing",
            "EstimatedCharges",
            "Currency",
            "USD"
            ]
        ]
        }"""

    val imageRequest = GetMetricWidgetImageRequest {
        metricWidget = myJSON
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricWidgetImage(imageRequest)
        val bytes = response.metricWidgetImage
        if (bytes != null) {
            File(fileName).writeBytes(bytes)
        }
    }
    println("You have successfully written data to $fileName")
}
```

- Untuk detail API, lihat GetMetricWidgetImagedi AWS SDK untuk referensi API Kotlin.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan CloudWatch dengan AWS SDK. Topik ini juga mencakup informasi tentang cara memulai dan detail versi-versi SDK sebelumnya.

Dapatkan detail CloudWatch dasbor

Contoh kode berikut menunjukkan cara mendapatkan detail CloudWatch dasbor Amazon.

.NET

AWS SDK for .NET

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
/// <summary>
/// Get information on a dashboard.
/// </summary>
/// <param name="dashboardName">The name of the dashboard.</param>
/// <returns>A JSON object with dashboard information.</returns>
public async Task<string> GetDashboard(string dashboardName)
{
    var dashboardResponse = await _amazonCloudWatch.GetDashboardAsync(
        new GetDashboardRequest()
        {
            DashboardName = dashboardName
        });

    return dashboardResponse.DashboardBody;
}
```

- Untuk detail API, lihat GetDashboarddi Referensi AWS SDK for .NET API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan CloudWatch dengan AWS SDK. Topik ini juga mencakup informasi tentang cara memulai dan detail versi-versi SDK sebelumnya.

Dapatkan nilai data CloudWatch metrik

Contoh kode berikut menunjukkan cara mendapatkan data CloudWatch metrik Amazon.

Contoh-contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan di dalam konteks. Anda dapat melihat tindakan ini dalam konteks pada contoh kode berikut:

- Mulai metrik, dasbor, dan alarm CloudWatch

.NET

AWS SDK for .NET

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /// <summary>
    /// Get data for CloudWatch metrics.
    /// </summary>
    /// <param name="minutesOfData">The number of minutes of data to include.</
param>
    /// <param name="useDescendingTime">True to return the data descending by
 time.</param>
    /// <param name="endDateUtc">The end date for the data, in UTC.</param>
    /// <param name="maxDataPoints">The maximum data points to include.</param>
    /// <param name="dataQueries">Optional data queries to include.</param>
    /// <returns>A list of the requested metric data.</returns>
    public async Task<List<MetricDataResult>> GetMetricData(int minutesOfData,
 bool useDescendingTime, DateTime? endDateUtc = null,
        int maxDataPoints = 0, List<MetricDataQuery>? dataQueries = null)
    {
        var metricData = new List<MetricDataResult>();
        // If no end time is provided, use the current time for the end time.
        endDateUtc ??= DateTime.UtcNow;
```

```
        var timeZoneOffset =
 TimeZoneInfo.Local.GetUtcOffset(endDateUtc.Value.ToLocalTime());
        var startTimeUtc = endDateUtc.Value.AddMinutes(-minutesOfData);
        // The timezone string should be in the format +0000, so use the timezone
 offset to format it correctly.
        var timeZoneString = $"{timeZoneOffset.Hours:D2}
{timeZoneOffset.Minutes:D2}";
        var paginatedMetricData = _amazonCloudWatch.Paginators.GetMetricData(
            new GetMetricDataRequest()
            {
                StartTimeUtc = startTimeUtc,
                EndTimeUtc = endDateUtc.Value,
                LabelOptions = new LabelOptions { Timezone = timeZoneString },
                ScanBy = useDescendingTime ? ScanBy.TimestampDescending :
 ScanBy.TimestampAscending,
                MaxDatapoints = maxDataPoints,
                MetricDataQueries = dataQueries,
            });

        await foreach (var data in paginatedMetricData.MetricDataResults)
        {
            metricData.Add(data);
        }
        return metricData;
    }
```

- Untuk detail API, lihat GetMetricDatadi Referensi AWS SDK for .NET API.

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    public static void getCustomMetricData(CloudWatchClient cw, String fileName)
 {
        try {
```

```
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            // Set the date.
            Instant nowDate = Instant.now();

            long hours = 1;
            long minutes = 30;
            Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
                    ChronoUnit.MINUTES);

            Metric met = Metric.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

            MetricStat metStat = MetricStat.builder()
                    .stat("Maximum")
                    .period(1)
                    .metric(met)
                    .build();

            MetricDataQuery dataQUery = MetricDataQuery.builder()
                    .metricStat(metStat)
                    .id("foo2")
                    .returnData(true)
                    .build();

            List<MetricDataQuery> dq = new ArrayList<>();
            dq.add(dataQUery);

            GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
                    .maxDatapoints(10)
                    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
                    .startTime(nowDate)
                    .endTime(date2)
                    .metricDataQueries(dq)
```

```
                    .build();

            GetMetricDataResponse response = cw.getMetricData(getMetReq);
            List<MetricDataResult> data = response.metricDataResults();
            for (MetricDataResult item : data) {
                System.out.println("The label is " + item.label());
                System.out.println("The status code is " +
 item.statusCode().toString());
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Untuk detail API, lihat GetMetricDatadi Referensi AWS SDK for Java 2.x API.

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
suspend fun getCustomMetricData(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set the date.
    val nowDate = Instant.now()
    val hours: Long = 1
    val minutes: Long = 30
    val date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(
```

```
        minutes,
        ChronoUnit.MINUTES
    )

    val met = Metric {
        metricName = customMetricName
        namespace = customMetricNamespace
    }

    val metStat = MetricStat {
        stat = "Maximum"
        period = 1
        metric = met
    }

    val dataQUery = MetricDataQuery {
        metricStat = metStat
        id = "foo2"
        returnData = true
    }

    val dq = ArrayList<MetricDataQuery>()
    dq.add(dataQUery)
    val getMetReq = GetMetricDataRequest {
        maxDatapoints = 10
        scanBy = ScanBy.TimestampDescending
        startTime = aws.smithy.kotlin.runtime.time.Instant(nowDate)
        endTime = aws.smithy.kotlin.runtime.time.Instant(date2)
        metricDataQueries = dq
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricData(getMetReq)
        response.metricDataResults?.forEach { item ->
            println("The label is ${item.label}")
            println("The status code is ${item.statusCode}")
        }
    }
}
```

- Untuk detail API, lihat GetMetricDatadi AWS SDK untuk referensi API Kotlin.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan CloudWatch dengan AWS SDK. Topik ini juga mencakup informasi tentang cara memulai dan detail versi-versi SDK sebelumnya.

Dapatkan statistik CloudWatch metrik menggunakan AWS SDK

Contoh kode berikut menunjukkan cara mendapatkan statistik CloudWatch metrik Amazon.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Mulai metrik, dasbor, dan alarm CloudWatch
- Mengelola metrik dan alarm

.NET

AWS SDK for .NET

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /// <summary>
    /// Get billing statistics using a call to a wrapper class.
    /// </summary>
    /// <returns>A collection of billing statistics.</returns>
    private static async Task<List<Datapoint>> SetupBillingStatistics()
    {
        // Make a request for EstimatedCharges with a period of one day for the
past seven days.
        var billingStatistics = await _cloudWatchWrapper.GetMetricStatistics(
            "AWS/Billing",
            "EstimatedCharges",
            new List<string>() { "Maximum" },
            new List<Dimension>() { new Dimension { Name = "Currency", Value =
"USD" } },
            7,
            86400);
```

```
        billingStatistics = billingStatistics.OrderBy(n => n.Timestamp).ToList();

        return billingStatistics;
    }

    /// <summary>
    /// Wrapper to get statistics for a specific CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <param name="statistics">The list of statistics to include.</param>
    /// <param name="dimensions">The list of dimensions to include.</param>
    /// <param name="days">The number of days in the past to include.</param>
    /// <param name="period">The period for the data.</param>
    /// <returns>A list of DataPoint objects for the statistics.</returns>
    public async Task<List<Datapoint>> GetMetricStatistics(string
metricNamespace,
        string metricName, List<string> statistics, List<Dimension> dimensions,
int days, int period)
    {
        var metricStatistics = await _amazonCloudWatch.GetMetricStatisticsAsync(
            new GetMetricStatisticsRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName,
                Dimensions = dimensions,
                Statistics = statistics,
                StartTimeUtc = DateTime.UtcNow.AddDays(-days),
                EndTimeUtc = DateTime.UtcNow,
                Period = period
            });

        return metricStatistics.Datapoints;
    }
```

- Untuk detail API, lihat GetMetricStatisticsdi Referensi AWS SDK for .NET API.

CLI

AWS CLI

Cara mendapatkan pemanfaatan CPU per instans EC2

Contoh berikut menggunakan perintah `get-metric-statistics` untuk mendapatkan pemanfaatan CPU untuk instans EC2 dengan ID i-abcdef.

```
aws cloudwatch get-metric-statistics --metric-name CPUUtilization --start-time
 2014-04-08T23:18:00Z --end-time 2014-04-09T23:18:00Z --period 3600 --namespace
 AWS/EC2 --statistics Maximum --dimensions Name=InstanceId,Value=i-abcdef
```

Output:

```
{
    "Datapoints": [
        {
            "Timestamp": "2014-04-09T11:18:00Z",
            "Maximum": 44.79,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T20:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T19:18:00Z",
            "Maximum": 50.85,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T09:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T03:18:00Z",
            "Maximum": 76.84,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T21:18:00Z",
            "Maximum": 48.96,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T14:18:00Z",
```

```
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T08:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T16:18:00Z",
            "Maximum": 45.55,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T06:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T13:18:00Z",
            "Maximum": 45.08,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T05:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T18:18:00Z",
            "Maximum": 46.88,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T17:18:00Z",
            "Maximum": 52.08,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T07:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
```

```
            "Timestamp": "2014-04-09T02:18:00Z",
            "Maximum": 51.23,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T12:18:00Z",
            "Maximum": 47.67,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-08T23:18:00Z",
            "Maximum": 46.88,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T10:18:00Z",
            "Maximum": 51.91,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T04:18:00Z",
            "Maximum": 47.13,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T15:18:00Z",
            "Maximum": 48.96,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T00:18:00Z",
            "Maximum": 48.16,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T01:18:00Z",
            "Maximum": 49.18,
            "Unit": "Percent"
        }
    ],
    "Label": "CPUUtilization"
}
```

Menentukan beberapa dimensi

Contoh berikut menggambarkan cara menentukan beberapa dimensi. Masing-masing dimensi ditentukan sebagai sebuah pasangan Nama/Nilai, yang menggunakan koma antara nama dan nilai tersebut. Beberapa dimensi dipisahkan dengan satu spasi. Jika sebuah metrik mencakup beberapa dimensi, Anda harus menetapkan sebuah nilai untuk masing-masing dimensi yang ditetapkan.

Untuk contoh lainnya menggunakan `get-metric-statistics` perintah, lihat Mendapatkan Statistik untuk Metrik di Panduan CloudWatch Pengembang Amazon.

```
aws cloudwatch get-metric-statistics --metric-name Buffers --
namespace MyNameSpace --dimensions Name=InstanceID,Value=i-abcdef
 Name=InstanceType,Value=m1.small --start-time 2016-10-15T04:00:00Z --end-time
 2016-10-19T07:00:00Z --statistics Average --period 60
```

- Untuk detail API, lihat GetMetricStatisticsdi Referensi AWS CLI Perintah.

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
   public static void getAndDisplayMetricStatistics(CloudWatchClient cw, String
nameSpace, String metVal,
           String metricOption, String date, Dimension myDimension) {
       try {
           Instant start = Instant.parse(date);
           Instant endDate = Instant.now();

           GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
                   .endTime(endDate)
                   .startTime(start)
                   .dimensions(myDimension)
                   .metricName(metVal)
```

```
                    .namespace(nameSpace)
                    .period(86400)
                    .statistics(Statistic.fromValue(metricOption))
                    .build();

            GetMetricStatisticsResponse response =
 cw.getMetricStatistics(statisticsRequest);
            List<Datapoint> data = response.datapoints();
            if (!data.isEmpty()) {
                for (Datapoint datapoint : data) {
                    System.out
                            .println("Timestamp: " + datapoint.timestamp() + "
 Maximum value: " + datapoint.maximum());
                }
            } else {
                System.out.println("The returned data list is empty");
            }

        } catch (CloudWatchException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Untuk detail API, lihat GetMetricStatisticsdi Referensi AWS SDK for Java 2.x API.

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
suspend fun getAndDisplayMetricStatistics(nameSpaceVal: String, metVal: String,
 metricOption: String, date: String, myDimension: Dimension) {
    val start = Instant.parse(date)
    val endDate = Instant.now()
    val statisticsRequest = GetMetricStatisticsRequest {
```

```
        endTime = aws.smithy.kotlin.runtime.time.Instant(endDate)
        startTime = aws.smithy.kotlin.runtime.time.Instant(start)
        dimensions = listOf(myDimension)
        metricName = metVal
        namespace = nameSpaceVal
        period = 86400
        statistics = listOf(Statistic.fromValue(metricOption))
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data = response.datapoints
        if (data != null) {
            if (data.isNotEmpty()) {
                for (datapoint in data) {
                    println("Timestamp: ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}
```

- Untuk detail API, lihat GetMetricStatisticsdi AWS SDK untuk referensi API Kotlin.

Python

SDK for Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
```

```
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def get_metric_statistics(self, namespace, name, start, end, period,
stat_types):
        """
        Gets statistics for a metric within a specified time span. Metrics are
grouped
        into the specified period.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param start: The UTC start time of the time span to retrieve.
        :param end: The UTC end time of the time span to retrieve.
        :param period: The period, in seconds, in which to group metrics. The
period
                       must match the granularity of the metric, which depends on
                       the metric's age. For example, metrics that are older than
                       three hours have a one-minute granularity, so the period
must
                       be at least 60 and must be a multiple of 60.
        :param stat_types: The type of statistics to retrieve, such as average
value
                           or maximum value.
        :return: The retrieved statistics for the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            stats = metric.get_statistics(
                StartTime=start, EndTime=end, Period=period,
Statistics=stat_types
            )
            logger.info(
                "Got %s statistics for %s.", len(stats["Datapoints"]),
stats["Label"]
            )
        except ClientError:
            logger.exception("Couldn't get statistics for %s.%s.", namespace,
name)
            raise
        else:
```

```
            return stats
```

- Untuk detail API, lihat GetMetricStatisticsdi AWS SDK for Python (Boto3) Referensi API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan CloudWatch dengan AWS SDK. Topik ini juga mencakup informasi tentang cara memulai dan detail versi-versi SDK sebelumnya.

Daftar CloudWatch dasbor

Contoh kode berikut menunjukkan cara membuat daftar CloudWatch dasbor Amazon.

.NET

AWS SDK for .NET

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
/// <summary>
/// Get a list of dashboards.
/// </summary>
/// <returns>A list of DashboardEntry objects.</returns>
public async Task<List<DashboardEntry>> ListDashboards()
{
    var results = new List<DashboardEntry>();
    var paginateDashboards = _amazonCloudWatch.Paginators.ListDashboards(
        new ListDashboardsRequest());
    // Get the entire list using the paginator.
    await foreach (var data in paginateDashboards.DashboardEntries)
    {
        results.Add(data);
    }

    return results;
}
```

- Untuk detail API, lihat ListDashboardsdi Referensi AWS SDK for .NET API.

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    public static void listDashboards(CloudWatchClient cw) {
        try {
            ListDashboardsIterable listRes = cw.listDashboardsPaginator();
            listRes.stream()
                    .flatMap(r -> r.dashboardEntries().stream())
                    .forEach(entry -> {
                        System.out.println("Dashboard name is: " +
entry.dashboardName());
                        System.out.println("Dashboard ARN is: " +
entry.dashboardArn());
                    });

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Untuk detail API, lihat ListDashboardsdi Referensi AWS SDK for Java 2.x API.

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
suspend fun listDashboards() {
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.listDashboardsPaginated({})
            .transform { it.dashboardEntries?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.dashboardName}")
                println("Dashboard ARN is ${obj.dashboardArn}")
            }
    }
}
```

- Untuk detail API, lihat ListDashboardsdi AWS SDK untuk referensi API Kotlin.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan CloudWatch dengan AWS SDK. Topik ini juga mencakup informasi tentang cara memulai dan detail versi-versi SDK sebelumnya.

Daftar CloudWatch metrik menggunakan SDK AWS

Contoh kode berikut menunjukkan cara membuat daftar metadata untuk metrik Amazon CloudWatch . Untuk mendapatkan data untuk metrik, gunakan GetMetricStatistics tindakan GetMetricData atau.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Mulai metrik, dasbor, dan alarm CloudWatch
- Mengelola metrik dan alarm

.NET

AWS SDK for .NET

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /// <summary>
    /// List metrics available, optionally within a namespace.
    /// </summary>
    /// <param name="metricNamespace">Optional CloudWatch namespace to use when
listing metrics.</param>
    /// <param name="filter">Optional dimension filter.</param>
    /// <param name="metricName">Optional metric name filter.</param>
    /// <returns>The list of metrics.</returns>
    public async Task<List<Metric>> ListMetrics(string? metricNamespace = null,
DimensionFilter? filter = null, string? metricName = null)
    {
        var results = new List<Metric>();
        var paginateMetrics = _amazonCloudWatch.Paginators.ListMetrics(
            new ListMetricsRequest
            {
                Namespace = metricNamespace,
                Dimensions = filter != null ? new List<DimensionFilter>
{ filter } : null,
                MetricName = metricName
            });
        // Get the entire list using the paginator.
        await foreach (var metric in paginateMetrics.Metrics)
        {
            results.Add(metric);
        }

        return results;
    }
```

- Untuk detail API, lihat ListMetricsdi Referensi AWS SDK for .NET API.

C++

SDK for C++

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di Repositori Contoh Kode AWS.

Sertakan file-file yang diperlukan.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/ListMetricsRequest.h>
#include <aws/monitoring/model/ListMetricsResult.h>
#include <iomanip>
#include <iostream>
```

Membuat daftar metrik.

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::ListMetricsRequest request;

        if (argc > 1)
        {
            request.SetMetricName(argv[1]);
        }

        if (argc > 2)
        {
            request.SetNamespace(argv[2]);
        }

        bool done = false;
        bool header = false;
        while (!done)
        {
            auto outcome = cw.ListMetrics(request);
            if (!outcome.IsSuccess())
            {
```

```
                std::cout << "Failed to list CloudWatch metrics:" <<
                    outcome.GetError().GetMessage() << std::endl;
                break;
            }

            if (!header)
            {
                std::cout << std::left << std::setw(48) << "MetricName" <<
                    std::setw(32) << "Namespace" << "DimensionNameValuePairs" <<
                    std::endl;
                header = true;
            }

            const auto &metrics = outcome.GetResult().GetMetrics();
            for (const auto &metric : metrics)
            {
                std::cout << std::left << std::setw(48) <<
                    metric.GetMetricName() << std::setw(32) <<
                    metric.GetNamespace();
                const auto &dimensions = metric.GetDimensions();
                for (auto iter = dimensions.cbegin();
                    iter != dimensions.cend(); ++iter)
                {
                    const auto &dimkv = *iter;
                    std::cout << dimkv.GetName() << " = " << dimkv.GetValue();
                    if (iter + 1 != dimensions.cend())
                    {
                        std::cout << ", ";
                    }
                }
                std::cout << std::endl;
            }

            const auto &next_token = outcome.GetResult().GetNextToken();
            request.SetNextToken(next_token);
            done = next_token.empty();
        }
```

- Untuk detail API, lihat ListMetricsdi Referensi AWS SDK for C++ API.

CLI

AWS CLI

Cara membuat daftar metrik untuk Amazon SNS

Contoh `list-metrics` berikut menampilkan metrik-metrik untuk Amazon SNS.

```
aws cloudwatch list-metrics \
    --namespace "AWS/SNS"
```

Output:

```
{
    "Metrics": [
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "NotifyMe"
                }
            ],
            "MetricName": "PublishSize"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "CFO"
                }
            ],
            "MetricName": "PublishSize"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "NotifyMe"
                }
            ],
```

```
            "MetricName": "NumberOfNotificationsFailed"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "NotifyMe"
                }
            ],
            "MetricName": "NumberOfNotificationsDelivered"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "NotifyMe"
                }
            ],
            "MetricName": "NumberOfMessagesPublished"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "CFO"
                }
            ],
            "MetricName": "NumberOfMessagesPublished"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "CFO"
                }
            ],
            "MetricName": "NumberOfNotificationsDelivered"
        },
        {
            "Namespace": "AWS/SNS",
```

```
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "CFO"
                }
            ],
            "MetricName": "NumberOfNotificationsFailed"
        }
    ]
}
```

- Untuk detail API, lihat ListMetricsdi Referensi AWS CLI Perintah.

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListMetrics {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:
                  <namespace>\s

                Where:
                  namespace - The namespace to filter against (for example, AWS/
EC2).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String namespace = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        listMets(cw, namespace);
        cw.close();
    }

    public static void listMets(CloudWatchClient cw, String namespace) {
        boolean done = false;
        String nextToken = null;

        try {
            while (!done) {

                ListMetricsResponse response;
                if (nextToken == null) {
                    ListMetricsRequest request = ListMetricsRequest.builder()
                            .namespace(namespace)
                            .build();

                    response = cw.listMetrics(request);
                } else {
                    ListMetricsRequest request = ListMetricsRequest.builder()
                            .namespace(namespace)
                            .nextToken(nextToken)
                            .build();
```

```
                    response = cw.listMetrics(request);
                }

                for (Metric metric : response.metrics()) {
                    System.out.printf("Retrieved metric %s",
 metric.metricName());
                    System.out.println();
                }

                if (response.nextToken() == null) {
                    done = true;
                } else {
                    nextToken = response.nextToken();
                }
            }

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Untuk detail API, lihat ListMetricsdi Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di Repositori Contoh Kode AWS.

Mengimpor modul SDK dan klien dan memanggil API.

```
import { ListMetricsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

export const main = () => {
```

```
  // Use the AWS console to see available namespaces and metric names. Custom
 metrics can also be created.
  // https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
viewing_metrics_with_cloudwatch.html
  const command = new ListMetricsCommand({
    Dimensions: [
      {
        Name: "LogGroupName",
      },
    ],
    MetricName: "IncomingLogEvents",
    Namespace: "AWS/Logs",
  });

  return client.send(command);
};
```

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat ListMetricsdi Referensi AWS SDK for JavaScript API.

SDK untuk JavaScript (v2)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
```

```
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  Dimensions: [
    {
      Name: "LogGroupName" /* required */,
    },
  ],
  MetricName: "IncomingLogEvents",
  Namespace: "AWS/Logs",
};

cw.listMetrics(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Metrics", JSON.stringify(data.Metrics));
  }
});
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat ListMetricsdi Referensi AWS SDK for JavaScript API.

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
suspend fun listMets(namespaceVal: String?): ArrayList<String>? {
    val metList = ArrayList<String>()
    val request = ListMetricsRequest {
        namespace = namespaceVal
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val reponse = cwClient.listMetrics(request)
```

```
        reponse.metrics?.forEach { metrics ->
            val data = metrics.metricName
            if (!metList.contains(data)) {
                metList.add(data!!)
            }
        }
    }
    return metList
}
```

- Untuk detail API, lihat ListMetricsdi AWS SDK untuk referensi API Kotlin.

Python

SDK for Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def list_metrics(self, namespace, name, recent=False):
        """
        Gets the metrics within a namespace that have the specified name.
        If the metric has no dimensions, a single metric is returned.
        Otherwise, metrics for all dimensions are returned.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param recent: When True, only metrics that have been active in the last
```

```
                    three hours are returned.
        :return: An iterator that yields the retrieved metrics.
        """
        try:
            kwargs = {"Namespace": namespace, "MetricName": name}
            if recent:
                kwargs["RecentlyActive"] = "PT3H"  # List past 3 hours only
            metric_iter = self.cloudwatch_resource.metrics.filter(**kwargs)
            logger.info("Got metrics for %s.%s.", namespace, name)
        except ClientError:
            logger.exception("Couldn't get metrics for %s.%s.", namespace, name)
            raise
        else:
            return metric_iter
```

- Untuk detail API, lihat ListMetricsdi AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK for Ruby

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
# Lists available metrics for a metric namespace in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param metric_namespace [String] The namespace of the metric.
# @example
#   list_metrics_for_namespace(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC'
#   )
def list_metrics_for_namespace(cloudwatch_client, metric_namespace)
  response = cloudwatch_client.list_metrics(namespace: metric_namespace)
```

```
  if response.metrics.count.positive?
    response.metrics.each do |metric|
      puts "  Metric name: #{metric.metric_name}"
      if metric.dimensions.count.positive?
        puts "    Dimensions:"
        metric.dimensions.each do |dimension|
          puts "      Name: #{dimension.name}, Value: #{dimension.value}"
        end
      else
        puts "No dimensions found."
      end
    end
  else
    puts "No metrics found for namespace '#{metric_namespace}'. " \
      "Note that it could take up to 15 minutes for recently-added metrics " \
      "to become available."
  end
end

# Example usage:
def run_me
  metric_namespace = "SITE/TRAFFIC"
  # Replace us-west-2 with the AWS Region you're using for Amazon CloudWatch.
  region = "us-east-1"

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)

  # Add three datapoints.
  puts "Continuing..." unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    "UniqueVisitors",
    "SiteName",
    "example.com",
    5_885.0,
    "Count"
  )

  puts "Continuing..." unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    "UniqueVisits",
    "SiteName",
    "example.com",
```

```
    8_628.0,
    "Count"
  )

  puts "Continuing..." unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    "PageViews",
    "PageURL",
    "example.html",
    18_057.0,
    "Count"
  )

  puts "Metrics for namespace '#{metric_namespace}':"
  list_metrics_for_namespace(cloudwatch_client, metric_namespace)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Untuk detail API, lihat ListMetricsdi Referensi AWS SDK for Ruby API.

SAP ABAP

SDK untuk SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
   "The following list-metrics example displays the metrics for Amazon
CloudWatch."
   TRY.
       oo_result = lo_cwt->listmetrics(           " oo_result is returned for
testing purposes. "
         iv_namespace = iv_namespace
       ).
       DATA(lt_metrics) = oo_result->get_metrics( ).
       MESSAGE 'Metrics retrieved.' TYPE 'I'.
```

```
      CATCH /aws1/cx_cwtinvparamvalueex .
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
    ENDTRY.
```

- Untuk detail API, lihat ListMetricsdi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan CloudWatch dengan AWS SDK. Topik ini juga mencakup informasi tentang cara memulai dan detail versi-versi SDK sebelumnya.

Masukkan satu set data ke dalam CloudWatch metrik menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menempatkan satu set data ke dalam CloudWatch metrik Amazon.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks di contoh-contoh kode berikut:

- Mulai metrik, dasbor, dan alarm CloudWatch
- Mengelola metrik dan alarm

Python

SDK for Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
```

```
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def put_metric_data_set(self, namespace, name, timestamp, unit, data_set):
        """
        Sends a set of data to CloudWatch for a metric. All of the data in the
set
        have the same timestamp and unit.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param timestamp: The UTC timestamp for the metric.
        :param unit: The unit of the metric.
        :param data_set: The set of data to send. This set is a dictionary that
                         contains a list of values and a list of corresponding
counts.
                         The value and count lists must be the same length.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[
                    {
                        "MetricName": name,
                        "Timestamp": timestamp,
                        "Values": data_set["values"],
                        "Counts": data_set["counts"],
                        "Unit": unit,
                    }
                ],
            )
            logger.info("Put data set for metric %s.%s.", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data set for metric %s.%s.",
namespace, name)
            raise
```

- Untuk detail API, lihat PutMetricDatadi AWS SDK for Python (Boto3) Referensi API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan CloudWatch dengan AWS SDK. Topik ini juga mencakup informasi tentang cara memulai dan detail versi-versi SDK sebelumnya.

Masukkan data ke dalam CloudWatch metrik menggunakan AWS SDK

Contoh kode berikut menunjukkan cara mempublikasikan titik data metrik ke Amazon CloudWatch.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Mulai metrik, dasbor, dan alarm CloudWatch
- Mengelola metrik dan alarm

.NET

AWS SDK for .NET

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
   /// <summary>
   /// Add some metric data using a call to a wrapper class.
   /// </summary>
   /// <param name="customMetricName">The metric name.</param>
   /// <param name="customMetricNamespace">The metric namespace.</param>
   /// <returns></returns>
   private static async Task<List<MetricDatum>> PutRandomMetricData(string
customMetricName,
       string customMetricNamespace)
   {
       List<MetricDatum> customData = new List<MetricDatum>();
       Random rnd = new Random();

       // Add 10 random values up to 100, starting with a timestamp 15 minutes
in the past.
```

```
        var utcNowMinus15 = DateTime.UtcNow.AddMinutes(-15);
        for (int i = 0; i < 10; i++)
        {
            var metricValue = rnd.Next(0, 100);
            customData.Add(
                new MetricDatum
                {
                    MetricName = customMetricName,
                    Value = metricValue,
                    TimestampUtc = utcNowMinus15.AddMinutes(i)
                }
            );
        }

        await _cloudWatchWrapper.PutMetricData(customMetricNamespace,
customData);
        return customData;
    }

    /// <summary>
    /// Wrapper to add metric data to a CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricData">A data object for the metric data.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutMetricData(string metricNamespace,
        List<MetricDatum> metricData)
    {
        var putDataResponse = await _amazonCloudWatch.PutMetricDataAsync(
            new PutMetricDataRequest()
            {
                MetricData = metricData,
                Namespace = metricNamespace,
            });

        return putDataResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Untuk detail API, lihat PutMetricDatadi Referensi AWS SDK for .NET API.

C++

SDK for C++

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di Repositori Contoh Kode AWS.

Sertakan file-file yang diperlukan.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricDataRequest.h>
#include <iostream>
```

Masukkan data ke dalam metrik.

```
        Aws::CloudWatch::CloudWatchClient cw;

        Aws::CloudWatch::Model::Dimension dimension;
        dimension.SetName("UNIQUE_PAGES");
        dimension.SetValue("URLS");

        Aws::CloudWatch::Model::MetricDatum datum;
        datum.SetMetricName("PAGES_VISITED");
        datum.SetUnit(Aws::CloudWatch::Model::StandardUnit::None);
        datum.SetValue(data_point);
        datum.AddDimensions(dimension);

        Aws::CloudWatch::Model::PutMetricDataRequest request;
        request.SetNamespace("SITE/TRAFFIC");
        request.AddMetricData(datum);

        auto outcome = cw.PutMetricData(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to put sample metric data:" <<
                outcome.GetError().GetMessage() << std::endl;
        }
```

```
        else
        {
            std::cout << "Successfully put sample metric data" << std::endl;
        }
```

- Untuk detail API, lihat PutMetricDatadi Referensi AWS SDK for C++ API.

CLI

AWS CLI

Untuk mempublikasikan metrik khusus ke Amazon CloudWatch

Contoh berikut menggunakan `put-metric-data` perintah untuk menerbitkan metrik khusus ke Amazon CloudWatch:

```
aws cloudwatch put-metric-data --namespace "Usage Metrics" --metric-data file://
metric.json
```

Nilai-nilai untuk metrik itu sendiri disimpan dalam file JSON, `metric.json`.

Berikut adalah isi dari file tersebut:

```
[
  {
    "MetricName": "New Posts",
    "Timestamp": "Wednesday, June 12, 2013 8:28:20 PM",
    "Value": 0.50,
    "Unit": "Count"
  }
]
```

Untuk informasi selengkapnya, lihat Menerbitkan Metrik Kustom di Panduan CloudWatch Pengembang Amazon.

Cara menentukan beberapa dimensi

Contoh berikut menggambarkan cara menentukan beberapa dimensi. Masing-masing dimensi ditentukan sebagai sebuah pasangan Name=Value. Beberapa dimensi dipisahkan menggunakan koma.:

```
aws cloudwatch put-metric-data --metric-name Buffers --namespace
 MyNameSpace --unit Bytes --value 231434333 --dimensions
 InstanceID=1-23456789,InstanceType=m1.small
```

- Untuk detail API, lihat PutMetricDatadi Referensi AWS CLI Perintah.

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
   public static void addMetricDataForAlarm(CloudWatchClient cw, String
fileName) {
       try {
           // Read values from the JSON file.
           JsonParser parser = new JsonFactory().createParser(new
File(fileName));
           com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
           String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
           String customMetricName =
rootNode.findValue("customMetricName").asText();

           // Set an Instant object.
           String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
           Instant instant = Instant.parse(time);

           MetricDatum datum = MetricDatum.builder()
                   .metricName(customMetricName)
                   .unit(StandardUnit.NONE)
                   .value(1001.00)
                   .timestamp(instant)
                   .build();
```

```
            MetricDatum datum2 = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1002.00)
                    .timestamp(instant)
                    .build();

            List<MetricDatum> metricDataList = new ArrayList<>();
            metricDataList.add(datum);
            metricDataList.add(datum2);

            PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace(customMetricNamespace)
                    .metricData(metricDataList)
                    .build();

            cw.putMetricData(request);
            System.out.println("Added metric values for for metric " +
customMetricName);

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Untuk detail API, lihat PutMetricDatadi Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di Repositori Contoh Kode AWS.

Mengimpor modul SDK dan klien dan memanggil API.

```
import { PutMetricDataCommand } from "@aws-sdk/client-cloudwatch";
```

```
import { client } from "../libs/client.js";

const run = async () => {
  // See https://docs.aws.amazon.com/AmazonCloudWatch/latest/APIReference/
API_PutMetricData.html#API_PutMetricData_RequestParameters
  // and https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
publishingMetrics.html
  // for more information about the parameters in this command.
  const command = new PutMetricDataCommand({
    MetricData: [
      {
        MetricName: "PAGES_VISITED",
        Dimensions: [
          {
            Name: "UNIQUE_PAGES",
            Value: "URLS",
          },
        ],
        Unit: "None",
        Value: 1.0,
      },
    ],
    Namespace: "SITE/TRAFFIC",
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for JavaScript.

- Untuk detail API, lihat PutMetricDatadi Referensi AWS SDK for JavaScript API.

SDK untuk JavaScript (v2)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

// Create parameters JSON for putMetricData
var params = {
  MetricData: [
    {
      MetricName: "PAGES_VISITED",
      Dimensions: [
        {
          Name: "UNIQUE_PAGES",
          Value: "URLS",
        },
      ],
      Unit: "None",
      Value: 1.0,
    },
  ],
  Namespace: "SITE/TRAFFIC",
};

cw.putMetricData(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", JSON.stringify(data));
  }
});
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for JavaScript.
- Untuk detail API, lihat PutMetricDatadi Referensi AWS SDK for JavaScript API.

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
suspend fun addMetricDataForAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum = MetricDatum {
        metricName = customMetricName
        unit = StandardUnit.None
        value = 1001.00
        timestamp = aws.smithy.kotlin.runtime.time.Instant(instant)
    }

    val datum2 = MetricDatum {
        metricName = customMetricName
        unit = StandardUnit.None
        value = 1002.00
        timestamp = aws.smithy.kotlin.runtime.time.Instant(instant)
    }
```

```
    val metricDataList = ArrayList<MetricDatum>()
    metricDataList.add(datum)
    metricDataList.add(datum2)

    val request = PutMetricDataRequest {
        namespace = customMetricNamespace
        metricData = metricDataList
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric $customMetricName")
    }
}
```

- Untuk detail API, lihat PutMetricDatadi AWS SDK untuk referensi API Kotlin.

Python

SDK for Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def put_metric_data(self, namespace, name, value, unit):
        """
        Sends a single data value to CloudWatch for a metric. This metric is
 given
```

```
        a timestamp of the current UTC time.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param value: The value of the metric.
        :param unit: The unit of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[{"MetricName": name, "Value": value, "Unit": unit}],
            )
            logger.info("Put data for metric %s.%s", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data for metric %s.%s", namespace,
name)
            raise
```

- Untuk detail API, lihat PutMetricDatadi AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK for Ruby

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
require "aws-sdk-cloudwatch"

# Adds a datapoint to a metric in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param metric_namespace [String] The namespace of the metric to add the
#   datapoint to.
# @param metric_name [String] The name of the metric to add the datapoint to.
```

```
# @param dimension_name [String] The name of the dimension to add the
#   datapoint to.
# @param dimension_value [String] The value of the dimension to add the
#   datapoint to.
# @param metric_value [Float] The value of the datapoint.
# @param metric_unit [String] The unit of measurement for the datapoint.
# @return [Boolean]
# @example
#   exit 1 unless datapoint_added_to_metric?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC',
#     'UniqueVisitors',
#     'SiteName',
#     'example.com',
#     5_885.0,
#     'Count'
#   )
def datapoint_added_to_metric?(
  cloudwatch_client,
  metric_namespace,
  metric_name,
  dimension_name,
  dimension_value,
  metric_value,
  metric_unit
)
  cloudwatch_client.put_metric_data(
    namespace: metric_namespace,
    metric_data: [
      {
        metric_name: metric_name,
        dimensions: [
          {
            name: dimension_name,
            value: dimension_value
          }
        ],
        value: metric_value,
        unit: metric_unit
      }
    ]
  )
  puts "Added data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}'."
```

```
  return true
rescue StandardError => e
  puts "Error adding data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}': #{e.message}"
  return false
end
```

- Untuk detail API, lihat PutMetricDatadi Referensi AWS SDK for Ruby API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan CloudWatch dengan AWS SDK. Topik ini juga mencakup informasi tentang cara memulai dan detail versi-versi SDK sebelumnya.

Skenario untuk CloudWatch menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menerapkan skenario umum CloudWatch dengan AWS SDK. Skenario ini menunjukkan cara menyelesaikan tugas tertentu dengan memanggil beberapa fungsi di dalamnya CloudWatch. Setiap skenario menyertakan tautan ke GitHub, di mana Anda dapat menemukan petunjuk tentang cara mengatur dan menjalankan kode.

Contoh-contoh

- Memulai CloudWatch alarm menggunakan SDK AWS
- Memulai CloudWatch metrik, dasbor, dan alarm menggunakan SDK AWS
- Mengelola CloudWatch metrik dan alarm menggunakan SDK AWS

Memulai CloudWatch alarm menggunakan SDK AWS

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Membuat alarm.
- Menonaktifkan tindakan alarm.
- Menjelaskan maksud alarm.
- Menghapus alarm.

SAP ABAP

SDK untuk SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
DATA lt_alarmnames TYPE /aws1/cl_cwtalarmnames_w=>tt_alarmnames.
DATA lo_alarmname TYPE REF TO /aws1/cl_cwtalarmnames_w.

"Create an alarm"
TRY.
    lo_cwt->putmetricalarm(
      iv_alarmname                = iv_alarm_name
      iv_comparisonoperator       = iv_comparison_operator
      iv_evaluationperiods        = iv_evaluation_periods
      iv_metricname               = iv_metric_name
      iv_namespace                = iv_namespace
      iv_statistic                = iv_statistic
      iv_threshold                = iv_threshold
      iv_actionsenabled           = iv_actions_enabled
      iv_alarmdescription         = iv_alarm_description
      iv_unit                     = iv_unit
      iv_period                   = iv_period
      it_dimensions               = it_dimensions
    ).
    MESSAGE 'Alarm created' TYPE 'I'.
  CATCH /aws1/cx_cwtlimitexceededfault.
    MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
ENDTRY.

"Create an ABAP internal table for the created alarm."
CREATE OBJECT lo_alarmname EXPORTING iv_value = iv_alarm_name.
INSERT lo_alarmname INTO TABLE lt_alarmnames.

"Disable alarm actions."
TRY.
    lo_cwt->disablealarmactions(
```

```
          it_alarmnames               = lt_alarmnames
        ).
        MESSAGE 'Alarm actions disabled' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_disablealarm_exception).
        DATA(lv_disablealarm_error) = |"{ lo_disablealarm_exception-
>av_err_code }" - { lo_disablealarm_exception->av_err_msg }|.
        MESSAGE lv_disablealarm_error TYPE 'E'.
    ENDTRY.

    "Describe alarm using the same ABAP internal table."
    TRY.
        oo_result = lo_cwt->describealarms(                      " oo_result is
 returned for testing purpose "
          it_alarmnames               = lt_alarmnames
        ).
        MESSAGE 'Alarms retrieved' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_describealarms_exception).
        DATA(lv_describealarms_error) = |"{ lo_describealarms_exception-
>av_err_code }" - { lo_describealarms_exception->av_err_msg }|.
        MESSAGE lv_describealarms_error TYPE 'E'.
    ENDTRY.

    "Delete alarm."
    TRY.
        lo_cwt->deletealarms(
          it_alarmnames = lt_alarmnames
        ).
        MESSAGE 'Alarms deleted' TYPE 'I'.
      CATCH /aws1/cx_cwtresourcenotfound .
        MESSAGE 'Resource being access is not found.' TYPE 'E'.
    ENDTRY.
```

- Untuk mengetahui hal detail mengenai API, silakan lihat topik-topik berikut di referensi API SDK AWS untuk ABAP SAP.
  - DeleteAlarms
  - DescribeAlarms
  - DisableAlarmActions
  - PutMetricAlarm

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan CloudWatch dengan AWS SDK. Topik ini juga mencakup informasi tentang cara memulai dan detail versi-versi SDK sebelumnya.

Memulai CloudWatch metrik, dasbor, dan alarm menggunakan SDK AWS

Contoh-contoh kode berikut menunjukkan cara:

- Daftar CloudWatch ruang nama dan metrik.
- Ambil statistik untuk metrik dan estimasi penagihan.
- Membuat dan memperbarui sebuah dasbor.
- Membuat dan menambahkan data ke metrik.
- Membuat dan memicu alarm, lalu lihat riwayat alarm.
- Menambahkan detektor anomali.
- Ambil gambar metrik, lalu bersihkan sumber daya.

.NET

AWS SDK for .NET

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara pengaturan dan menjalankannya di Repositori Contoh Kode AWS.

Jalankan skenario interaktif di penggugah/prompt perintah.

```
public class CloudWatchScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    To enable billing metrics and statistics for this example, make sure billing
 alerts are enabled for your account:
    https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics
```

```
    This .NET example performs the following tasks:
        1. List and select a CloudWatch namespace.
        2. List and select a CloudWatch metric.
        3. Get statistics for a CloudWatch metric.
        4. Get estimated billing statistics for the last week.
        5. Create a new CloudWatch dashboard with two metrics.
        6. List current CloudWatch dashboards.
        7. Create a CloudWatch custom metric and add metric data.
        8. Add the custom metric to the dashboard.
        9. Create a CloudWatch alarm for the custom metric.
       10. Describe current CloudWatch alarms.
       11. Get recent data for the custom metric.
       12. Add data to the custom metric to trigger the alarm.
       13. Wait for an alarm state.
       14. Get history for the CloudWatch alarm.
       15. Add an anomaly detector.
       16. Describe current anomaly detectors.
       17. Get and display a metric image.
       18. Clean up resources.
    */

    private static ILogger logger = null!;
    private static CloudWatchWrapper _cloudWatchWrapper = null!;
    private static IConfiguration _configuration = null!;
    private static readonly List<string> _statTypes = new List<string>
{ "SampleCount", "Average", "Sum", "Minimum", "Maximum" };
    private static SingleMetricAnomalyDetector? anomalyDetector = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonCloudWatch>()
            .AddTransient<CloudWatchWrapper>()
        )
        .Build();
```

```
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<CloudWatchScenario>();

        _cloudWatchWrapper =
host.Services.GetRequiredService<CloudWatchWrapper>();

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon CloudWatch example scenario.");
        Console.WriteLine(new string('-', 80));

        try
        {
            var selectedNamespace = await SelectNamespace();
            var selectedMetric = await SelectMetric(selectedNamespace);
            await GetAndDisplayMetricStatistics(selectedNamespace,
selectedMetric);
            await GetAndDisplayEstimatedBilling();
            await CreateDashboardWithMetrics();
            await ListDashboards();
            await CreateNewCustomMetric();
            await AddMetricToDashboard();
            await CreateMetricAlarm();
            await DescribeAlarms();
            await GetCustomMetricData();
            await AddMetricDataForAlarm();
            await CheckForMetricAlarm();
            await GetAlarmHistory();
            anomalyDetector = await AddAnomalyDetector();
            await DescribeAnomalyDetectors();
            await GetAndOpenMetricImage();
            await CleanupResources();
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
            await CleanupResources();
        }
```

```
    }

    /// <summary>
    /// Select a namespace.
    /// </summary>
    /// <returns>The selected namespace.</returns>
    private static async Task<string> SelectNamespace()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"1. Select a CloudWatch Namespace from a list of
Namespaces.");
        var metrics = await _cloudWatchWrapper.ListMetrics();
        // Get a distinct list of namespaces.
        var namespaces = metrics.Select(m => m.Namespace).Distinct().ToList();
        for (int i = 0; i < namespaces.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {namespaces[i]}");
        }

        var namespaceChoiceNumber = 0;
        while (namespaceChoiceNumber < 1 || namespaceChoiceNumber >
namespaces.Count)
        {
            Console.WriteLine(
                "Select a namespace by entering a number from the preceding
list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out namespaceChoiceNumber);
        }

        var selectedNamespace = namespaces[namespaceChoiceNumber - 1];

        Console.WriteLine(new string('-', 80));

        return selectedNamespace;
    }

    /// <summary>
    /// Select a metric from a namespace.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <returns>The metric name.</returns>
    private static async Task<Metric> SelectMetric(string metricNamespace)
```

```
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"2. Select a CloudWatch metric from a namespace.");

        var namespaceMetrics = await
_cloudWatchWrapper.ListMetrics(metricNamespace);

        for (int i = 0; i < namespaceMetrics.Count && i < 15; i++)
        {
            var dimensionsWithValues = namespaceMetrics[i].Dimensions
                .Where(d => !string.Equals("None", d.Value));
            Console.WriteLine($"\t{i + 1}. {namespaceMetrics[i].MetricName} " +
                              $"{string.Join(", :", dimensionsWithValues.Select(d
=> d.Value))}");
        }

        var metricChoiceNumber = 0;
        while (metricChoiceNumber < 1 || metricChoiceNumber >
namespaceMetrics.Count)
        {
            Console.WriteLine(
                "Select a metric by entering a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out metricChoiceNumber);
        }

        var selectedMetric = namespaceMetrics[metricChoiceNumber - 1];

        Console.WriteLine(new string('-', 80));

        return selectedMetric;
    }

    /// <summary>
    /// Get and display metric statistics for a specific metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task GetAndDisplayMetricStatistics(string
metricNamespace, Metric metric)
    {
        Console.WriteLine(new string('-', 80));
```

```
        Console.WriteLine($"3. Get CloudWatch metric statistics for the last
day.");

        for (int i = 0; i < _statTypes.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {_statTypes[i]}");
        }

        var statisticChoiceNumber = 0;
        while (statisticChoiceNumber < 1 || statisticChoiceNumber >
_statTypes.Count)
        {
            Console.WriteLine(
                "Select a metric statistic by entering a number from the
preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out statisticChoiceNumber);
        }

        var selectedStatistic = _statTypes[statisticChoiceNumber - 1];
        var statisticsList = new List<string> { selectedStatistic };

        var metricStatistics = await
_cloudWatchWrapper.GetMetricStatistics(metricNamespace, metric.MetricName,
statisticsList, metric.Dimensions, 1, 60);

        if (!metricStatistics.Any())
        {
            Console.WriteLine($"No {selectedStatistic} statistics found for
{metric} in namespace {metricNamespace}.");
        }

        metricStatistics = metricStatistics.OrderBy(s => s.Timestamp).ToList();
        for (int i = 0; i < metricStatistics.Count && i < 10; i++)
        {
            var metricStat = metricStatistics[i];
            var statValue =
metricStat.GetType().GetProperty(selectedStatistic)!.GetValue(metricStat, null);
            Console.WriteLine($"\t{i + 1}. Timestamp
{metricStatistics[i].Timestamp:G} {selectedStatistic}: {statValue}");
        }

        Console.WriteLine(new string('-', 80));
    }
```

```
   /// <summary>
   /// Get and display estimated billing statistics.
   /// </summary>
   /// <param name="metricNamespace">The namespace for metrics.</param>
   /// <param name="metric">The CloudWatch metric.</param>
   /// <returns>Async task.</returns>
   private static async Task GetAndDisplayEstimatedBilling()
   {
       Console.WriteLine(new string('-', 80));
       Console.WriteLine($"4. Get CloudWatch estimated billing for the last
week.");

       var billingStatistics = await SetupBillingStatistics();

       for (int i = 0; i < billingStatistics.Count; i++)
       {
           Console.WriteLine($"\t{i + 1}. Timestamp
{billingStatistics[i].Timestamp:G} : {billingStatistics[i].Maximum}");
       }

       Console.WriteLine(new string('-', 80));
   }

   /// <summary>
   /// Get billing statistics using a call to a wrapper class.
   /// </summary>
   /// <returns>A collection of billing statistics.</returns>
   private static async Task<List<Datapoint>> SetupBillingStatistics()
   {
       // Make a request for EstimatedCharges with a period of one day for the
past seven days.
       var billingStatistics = await _cloudWatchWrapper.GetMetricStatistics(
           "AWS/Billing",
           "EstimatedCharges",
           new List<string>() { "Maximum" },
           new List<Dimension>() { new Dimension { Name = "Currency", Value =
"USD" } },
           7,
           86400);

       billingStatistics = billingStatistics.OrderBy(n => n.Timestamp).ToList();

       return billingStatistics;
```

```
    }

    /// <summary>
    /// Create a dashboard with metrics.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task CreateDashboardWithMetrics()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"5. Create a new CloudWatch dashboard with metrics.");
        var dashboardName = _configuration["dashboardName"];
        var newDashboard = new DashboardModel();
        _configuration.GetSection("dashboardExampleBody").Bind(newDashboard);
        var newDashboardString = JsonSerializer.Serialize(
            newDashboard,
            new JsonSerializerOptions
            {
                DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull
            });
        var validationMessages =
            await _cloudWatchWrapper.PutDashboard(dashboardName,
newDashboardString);

        Console.WriteLine(validationMessages.Any() ? $"\tValidation messages:" :
null);
        for (int i = 0; i < validationMessages.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {validationMessages[i].Message}");
        }
        Console.WriteLine($"\tDashboard {dashboardName} was created.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List dashboards.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListDashboards()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"6. List the CloudWatch dashboards in the current
account.");
```

```
        var dashboards = await _cloudWatchWrapper.ListDashboards();

        for (int i = 0; i < dashboards.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {dashboards[i].DashboardName}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Create and add data for a new custom metric.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CreateNewCustomMetric()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"7. Create and add data for a new custom metric.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var customData = await PutRandomMetricData(customMetricName,
customMetricNamespace);

        var valuesString = string.Join(',', customData.Select(d => d.Value));
        Console.WriteLine($"\tAdded metric values for for metric
{customMetricName}: \n\t{valuesString}");

        Console.WriteLine(new string('-', 80));
    }


    /// <summary>
    /// Add some metric data using a call to a wrapper class.
    /// </summary>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <returns></returns>
    private static async Task<List<MetricDatum>> PutRandomMetricData(string
customMetricName,
        string customMetricNamespace)
    {
```

```
        List<MetricDatum> customData = new List<MetricDatum>();
        Random rnd = new Random();

        // Add 10 random values up to 100, starting with a timestamp 15 minutes
in the past.
        var utcNowMinus15 = DateTime.UtcNow.AddMinutes(-15);
        for (int i = 0; i < 10; i++)
        {
            var metricValue = rnd.Next(0, 100);
            customData.Add(
                new MetricDatum
                {
                    MetricName = customMetricName,
                    Value = metricValue,
                    TimestampUtc = utcNowMinus15.AddMinutes(i)
                }
            );
        }

        await _cloudWatchWrapper.PutMetricData(customMetricNamespace,
customData);
        return customData;
    }

    /// <summary>
    /// Add the custom metric to the dashboard.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task AddMetricToDashboard()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"8. Add the new custom metric to the dashboard.");

        var dashboardName = _configuration["dashboardName"];

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var validationMessages = await SetupDashboard(customMetricNamespace,
customMetricName, dashboardName);

        Console.WriteLine(validationMessages.Any() ? $"\tValidation messages:" :
null);
        for (int i = 0; i < validationMessages.Count; i++)
```

```
        {
            Console.WriteLine($"\t{i + 1}. {validationMessages[i].Message}");
        }
        Console.WriteLine($"\tDashboard {dashboardName} updated with metric
{customMetricName}.");
        Console.WriteLine(new string('-', 80));
    }


    /// <summary>
    /// Set up a dashboard using a call to the wrapper class.
    /// </summary>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="dashboardName">The name of the dashboard.</param>
    /// <returns>A list of validation messages.</returns>
    private static async Task<List<DashboardValidationMessage>> SetupDashboard(
        string customMetricNamespace, string customMetricName, string
dashboardName)
    {
        // Get the dashboard model from configuration.
        var newDashboard = new DashboardModel();
        _configuration.GetSection("dashboardExampleBody").Bind(newDashboard);

        // Add a new metric to the dashboard.
        newDashboard.Widgets.Add(new Widget
        {
            Height = 8,
            Width = 8,
            Y = 8,
            X = 0,
            Type = "metric",
            Properties = new Properties
            {
                Metrics = new List<List<object>>
                    { new() { customMetricNamespace, customMetricName } },
                View = "timeSeries",
                Region = "us-east-1",
                Stat = "Sum",
                Period = 86400,
                YAxis = new YAxis { Left = new Left { Min = 0, Max = 100 } },
                Title = "Custom Metric Widget",
                LiveData = true,
                Sparkline = true,
```

```
                Trend = true,
                Stacked = false,
                SetPeriodToTimeRange = false
            }
        });

        var newDashboardString = JsonSerializer.Serialize(newDashboard,
            new JsonSerializerOptions
            { DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull });
        var validationMessages =
            await _cloudWatchWrapper.PutDashboard(dashboardName,
newDashboardString);

        return validationMessages;
    }

    /// <summary>
    /// Create a CloudWatch alarm for the new metric.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CreateMetricAlarm()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"9. Create a CloudWatch alarm for the new metric.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var alarmName = _configuration["exampleAlarmName"];
        var accountId = _configuration["accountId"];
        var region = _configuration["region"];
        var emailTopic = _configuration["emailTopic"];
        var alarmActions = new List<string>();

        if (GetYesNoResponse(
                $"\tAdd an email action for topic {emailTopic} to alarm
{alarmName}? (y/n)"))
        {
            _cloudWatchWrapper.AddEmailAlarmAction(accountId, region, emailTopic,
alarmActions);
        }

        await _cloudWatchWrapper.PutMetricEmailAlarm(
            "Example metric alarm",
```

```
            alarmName,
            ComparisonOperator.GreaterThanOrEqualToThreshold,
            customMetricName,
            customMetricNamespace,
            100,
            alarmActions);

        Console.WriteLine($"\tAlarm {alarmName} added for metric
{customMetricName}.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Describe Alarms.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task DescribeAlarms()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"10. Describe CloudWatch alarms in the current
account.");

        var alarms = await _cloudWatchWrapper.DescribeAlarms();
        alarms = alarms.OrderByDescending(a => a.StateUpdatedTimestamp).ToList();

        for (int i = 0; i < alarms.Count && i < 10; i++)
        {
            var alarm = alarms[i];
            Console.WriteLine($"\t{i + 1}. {alarm.AlarmName}");
            Console.WriteLine($"\tState: {alarm.StateValue} for
{alarm.MetricName} {alarm.ComparisonOperator} {alarm.Threshold}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get the recent data for the metric.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetCustomMetricData()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"11. Get current data for new custom metric.");
```

```
        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];
        var accountId = _configuration["accountId"];

        var query = new List<MetricDataQuery>
        {
            new MetricDataQuery
            {
                AccountId = accountId,
                Id = "m1",
                Label = "Custom Metric Data",
                MetricStat = new MetricStat
                {
                    Metric = new Metric
                    {
                        MetricName = customMetricName,
                        Namespace = customMetricNamespace,
                    },
                    Period = 1,
                    Stat = "Maximum"
                }
            }
        };

        var metricData = await _cloudWatchWrapper.GetMetricData(
            20,
            true,
            DateTime.UtcNow.AddMinutes(1),
            20,
            query);

        for (int i = 0; i < metricData.Count; i++)
        {
            for (int j = 0; j < metricData[i].Values.Count; j++)
            {
                Console.WriteLine(
                    $"\tTimestamp {metricData[i].Timestamps[j]:G} Value:
{metricData[i].Values[j]}");
            }
        }

        Console.WriteLine(new string('-', 80));
    }
```

```
    /// <summary>
    /// Add metric data to trigger an alarm.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task AddMetricDataForAlarm()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"12. Add metric data to the custom metric to trigger
an alarm.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];
        var nowUtc = DateTime.UtcNow;
        List<MetricDatum> customData = new List<MetricDatum>
        {
            new MetricDatum
            {
                MetricName = customMetricName,
                Value = 101,
                TimestampUtc = nowUtc.AddMinutes(-2)
            },
            new MetricDatum
            {
                MetricName = customMetricName,
                Value = 101,
                TimestampUtc = nowUtc.AddMinutes(-1)
            },
            new MetricDatum
            {
                MetricName = customMetricName,
                Value = 101,
                TimestampUtc = nowUtc
            }
        };
        var valuesString = string.Join(',', customData.Select(d => d.Value));
        Console.WriteLine($"\tAdded metric values for for metric
{customMetricName}: \n\t{valuesString}");
        await _cloudWatchWrapper.PutMetricData(customMetricNamespace,
customData);

        Console.WriteLine(new string('-', 80));
    }
```

```
    /// <summary>
    /// Check for a metric alarm using the DescribeAlarmsForMetric action.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CheckForMetricAlarm()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"13. Checking for an alarm state.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];
        var hasAlarm = false;
        var retries = 10;
        while (!hasAlarm && retries > 0)
        {
            var alarms = await
_cloudWatchWrapper.DescribeAlarmsForMetric(customMetricNamespace,
customMetricName);
            hasAlarm = alarms.Any(a => a.StateValue == StateValue.ALARM);
            retries--;
            Thread.Sleep(20000);
        }

        Console.WriteLine(hasAlarm
            ? $"\tAlarm state found for {customMetricName}."
            : $"\tNo Alarm state found for {customMetricName} after 10
retries.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get history for an alarm.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetAlarmHistory()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"14. Get alarm history.");

        var exampleAlarmName = _configuration["exampleAlarmName"];

        var alarmHistory = await
_cloudWatchWrapper.DescribeAlarmHistory(exampleAlarmName, 2);
```

```
        for (int i = 0; i < alarmHistory.Count; i++)
        {
            var history = alarmHistory[i];
            Console.WriteLine($"\t{i + 1}. {history.HistorySummary}, time
{history.Timestamp:g}");
        }
        if (!alarmHistory.Any())
        {
            Console.WriteLine($"\tNo alarm history data found for
{exampleAlarmName}.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Add an anomaly detector.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<SingleMetricAnomalyDetector> AddAnomalyDetector()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"15. Add an anomaly detector.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var detector = new SingleMetricAnomalyDetector
        {
            MetricName = customMetricName,
            Namespace = customMetricNamespace,
            Stat = "Maximum"
        };
        await _cloudWatchWrapper.PutAnomalyDetector(detector);
        Console.WriteLine($"\tAdded anomaly detector for metric
{customMetricName}.");

        Console.WriteLine(new string('-', 80));
        return detector;
    }

    /// <summary>
    /// Describe anomaly detectors.
```

```
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task DescribeAnomalyDetectors()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"16. Describe anomaly detectors in the current
account.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var detectors = await
_cloudWatchWrapper.DescribeAnomalyDetectors(customMetricNamespace,
customMetricName);

        for (int i = 0; i < detectors.Count; i++)
        {
            var detector = detectors[i];
            Console.WriteLine($"\t{i + 1}.
{detector.SingleMetricAnomalyDetector.MetricName}, state
{detector.StateValue}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Fetch and open a metrics image for a CloudWatch metric and namespace.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetAndOpenMetricImage()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("17. Get a metric image from CloudWatch.");

        Console.WriteLine($"\tGetting Image data for custom metric.");
        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var memoryStream = await
_cloudWatchWrapper.GetTimeSeriesMetricImage(customMetricNamespace,
customMetricName, "Maximum", 10);
        var file = _cloudWatchWrapper.SaveMetricImage(memoryStream,
"MetricImages");
```

```
        ProcessStartInfo info = new ProcessStartInfo();

        Console.WriteLine($"\tFile saved as {Path.GetFileName(file)}.");
        Console.WriteLine($"\tPress enter to open the image.");
        Console.ReadLine();
        info.FileName = Path.Combine("ms-photos://", file);
        info.UseShellExecute = true;
        info.CreateNoWindow = true;
        info.Verb = string.Empty;

        Process.Start(info);

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up created resources.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"18. Clean up resources.");

        var dashboardName = _configuration["dashboardName"];
        if (GetYesNoResponse($"\tDelete dashboard {dashboardName}? (y/n)"))
        {
            Console.WriteLine($"\tDeleting dashboard.");
            var dashboardList = new List<string> { dashboardName };
            await _cloudWatchWrapper.DeleteDashboards(dashboardList);
        }

        var alarmName = _configuration["exampleAlarmName"];
        if (GetYesNoResponse($"\tDelete alarm {alarmName}? (y/n)"))
        {
            Console.WriteLine($"\tCleaning up alarms.");
            var alarms = new List<string> { alarmName };
            await _cloudWatchWrapper.DeleteAlarms(alarms);
        }
```

```
        if (GetYesNoResponse($"\tDelete anomaly detector? (y/n)") &&
 anomalyDetector != null)
        {
            Console.WriteLine($"\tCleaning up anomaly detector.");

            await _cloudWatchWrapper.DeleteAnomalyDetector(
                anomalyDetector);
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
}
```

Metode pembungkus yang digunakan oleh skenario untuk CloudWatch tindakan.

```
/// <summary>
/// Wrapper class for Amazon CloudWatch methods.
/// </summary>
public class CloudWatchWrapper
{
    private readonly IAmazonCloudWatch _amazonCloudWatch;
    private readonly ILogger<CloudWatchWrapper> _logger;

    /// <summary>
    /// Constructor for the CloudWatch wrapper.
    /// </summary>
```

```
    /// <param name="amazonCloudWatch">The injected CloudWatch client.</param>
    /// <param name="logger">The injected logger for the wrapper.</param>
    public CloudWatchWrapper(IAmazonCloudWatch amazonCloudWatch,
ILogger<CloudWatchWrapper> logger)

    {
        _logger = logger;
        _amazonCloudWatch = amazonCloudWatch;
    }

    /// <summary>
    /// List metrics available, optionally within a namespace.
    /// </summary>
    /// <param name="metricNamespace">Optional CloudWatch namespace to use when
listing metrics.</param>
    /// <param name="filter">Optional dimension filter.</param>
    /// <param name="metricName">Optional metric name filter.</param>
    /// <returns>The list of metrics.</returns>
    public async Task<List<Metric>> ListMetrics(string? metricNamespace = null,
DimensionFilter? filter = null, string? metricName = null)
    {
        var results = new List<Metric>();
        var paginateMetrics = _amazonCloudWatch.Paginators.ListMetrics(
            new ListMetricsRequest
            {
                Namespace = metricNamespace,
                Dimensions = filter != null ? new List<DimensionFilter>
{ filter } : null,
                MetricName = metricName
            });
        // Get the entire list using the paginator.
        await foreach (var metric in paginateMetrics.Metrics)
        {
            results.Add(metric);
        }

        return results;
    }

    /// <summary>
    /// Wrapper to get statistics for a specific CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
```

```
    /// <param name="statistics">The list of statistics to include.</param>
    /// <param name="dimensions">The list of dimensions to include.</param>
    /// <param name="days">The number of days in the past to include.</param>
    /// <param name="period">The period for the data.</param>
    /// <returns>A list of DataPoint objects for the statistics.</returns>
    public async Task<List<Datapoint>> GetMetricStatistics(string
 metricNamespace,
        string metricName, List<string> statistics, List<Dimension> dimensions,
 int days, int period)
    {
        var metricStatistics = await _amazonCloudWatch.GetMetricStatisticsAsync(
            new GetMetricStatisticsRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName,
                Dimensions = dimensions,
                Statistics = statistics,
                StartTimeUtc = DateTime.UtcNow.AddDays(-days),
                EndTimeUtc = DateTime.UtcNow,
                Period = period
            });

        return metricStatistics.Datapoints;
    }

    /// <summary>
    /// Wrapper to create or add to a dashboard with metrics.
    /// </summary>
    /// <param name="dashboardName">The name for the dashboard.</param>
    /// <param name="dashboardBody">The metric data in JSON for the dashboard.</
param>
    /// <returns>A list of validation messages for the dashboard.</returns>
    public async Task<List<DashboardValidationMessage>> PutDashboard(string
 dashboardName,
        string dashboardBody)
    {
        // Updating a dashboard replaces all contents.
        // Best practice is to include a text widget indicating this dashboard
 was created programmatically.
        var dashboardResponse = await _amazonCloudWatch.PutDashboardAsync(
            new PutDashboardRequest()
            {
                DashboardName = dashboardName,
                DashboardBody = dashboardBody
```

```
            });

        return dashboardResponse.DashboardValidationMessages;
    }


    /// <summary>
    /// Get information on a dashboard.
    /// </summary>
    /// <param name="dashboardName">The name of the dashboard.</param>
    /// <returns>A JSON object with dashboard information.</returns>
    public async Task<string> GetDashboard(string dashboardName)
    {
        var dashboardResponse = await _amazonCloudWatch.GetDashboardAsync(
            new GetDashboardRequest()
            {
                DashboardName = dashboardName
            });

        return dashboardResponse.DashboardBody;
    }


    /// <summary>
    /// Get a list of dashboards.
    /// </summary>
    /// <returns>A list of DashboardEntry objects.</returns>
    public async Task<List<DashboardEntry>> ListDashboards()
    {
        var results = new List<DashboardEntry>();
        var paginateDashboards = _amazonCloudWatch.Paginators.ListDashboards(
            new ListDashboardsRequest());
        // Get the entire list using the paginator.
        await foreach (var data in paginateDashboards.DashboardEntries)
        {
            results.Add(data);
        }

        return results;
    }

    /// <summary>
    /// Wrapper to add metric data to a CloudWatch metric.
    /// </summary>
```

```
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricData">A data object for the metric data.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutMetricData(string metricNamespace,
        List<MetricDatum> metricData)
    {
        var putDataResponse = await _amazonCloudWatch.PutMetricDataAsync(
            new PutMetricDataRequest()
            {
                MetricData = metricData,
                Namespace = metricNamespace,
            });

        return putDataResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Get an image for a metric graphed over time.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metric">The name of the metric.</param>
    /// <param name="stat">The name of the stat to chart.</param>
    /// <param name="period">The period to use for the chart.</param>
    /// <returns>A memory stream for the chart image.</returns>
    public async Task<MemoryStream> GetTimeSeriesMetricImage(string
metricNamespace, string metric, string stat, int period)
    {
        var metricImageWidget = new
        {
            title = "Example Metric Graph",
            view = "timeSeries",
            stacked = false,
            period = period,
            width = 1400,
            height = 600,
            metrics = new List<List<object>>
                { new() { metricNamespace, metric, new { stat } } }
        };

        var metricImageWidgetString =
JsonSerializer.Serialize(metricImageWidget);
        var imageResponse = await _amazonCloudWatch.GetMetricWidgetImageAsync(
            new GetMetricWidgetImageRequest()
            {
```

```
                MetricWidget = metricImageWidgetString
            });

        return imageResponse.MetricWidgetImage;
    }

    /// <summary>
    /// Save a metric image to a file.
    /// </summary>
    /// <param name="memoryStream">The MemoryStream for the metric image.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The path to the file.</returns>
    public string SaveMetricImage(MemoryStream memoryStream, string metricName)
    {
        var metricFileName = $"{metricName}_{DateTime.Now.Ticks}.png";
        using var sr = new StreamReader(memoryStream);
        // Writes the memory stream to a file.
        File.WriteAllBytes(metricFileName, memoryStream.ToArray());
        var filePath = Path.Join(AppDomain.CurrentDomain.BaseDirectory,
            metricFileName);
        return filePath;
    }

    /// <summary>
    /// Get data for CloudWatch metrics.
    /// </summary>
    /// <param name="minutesOfData">The number of minutes of data to include.</
param>
    /// <param name="useDescendingTime">True to return the data descending by
 time.</param>
    /// <param name="endDateUtc">The end date for the data, in UTC.</param>
    /// <param name="maxDataPoints">The maximum data points to include.</param>
    /// <param name="dataQueries">Optional data queries to include.</param>
    /// <returns>A list of the requested metric data.</returns>
    public async Task<List<MetricDataResult>> GetMetricData(int minutesOfData,
 bool useDescendingTime, DateTime? endDateUtc = null,
        int maxDataPoints = 0, List<MetricDataQuery>? dataQueries = null)
    {
        var metricData = new List<MetricDataResult>();
        // If no end time is provided, use the current time for the end time.
        endDateUtc ??= DateTime.UtcNow;
        var timeZoneOffset =
 TimeZoneInfo.Local.GetUtcOffset(endDateUtc.Value.ToLocalTime());
        var startTimeUtc = endDateUtc.Value.AddMinutes(-minutesOfData);
```

```
        // The timezone string should be in the format +0000, so use the timezone
 offset to format it correctly.
        var timeZoneString = $"{timeZoneOffset.Hours:D2}
{timeZoneOffset.Minutes:D2}";
        var paginatedMetricData = _amazonCloudWatch.Paginators.GetMetricData(
            new GetMetricDataRequest()
            {
                StartTimeUtc = startTimeUtc,
                EndTimeUtc = endDateUtc.Value,
                LabelOptions = new LabelOptions { Timezone = timeZoneString },
                ScanBy = useDescendingTime ? ScanBy.TimestampDescending :
 ScanBy.TimestampAscending,
                MaxDatapoints = maxDataPoints,
                MetricDataQueries = dataQueries,
            });

        await foreach (var data in paginatedMetricData.MetricDataResults)
        {
            metricData.Add(data);
        }
        return metricData;
    }

    /// <summary>
    /// Add a metric alarm to send an email when the metric passes a threshold.
    /// </summary>
    /// <param name="alarmDescription">A description of the alarm.</param>
    /// <param name="alarmName">The name for the alarm.</param>
    /// <param name="comparison">The type of comparison to use.</param>
    /// <param name="metricName">The name of the metric for the alarm.</param>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="threshold">The threshold value for the alarm.</param>
    /// <param name="alarmActions">Optional actions to execute when in an alarm
 state.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutMetricEmailAlarm(string alarmDescription, string
 alarmName, ComparisonOperator comparison,
        string metricName, string metricNamespace, double threshold, List<string>
 alarmActions = null!)
    {
        try
        {
            var putEmailAlarmResponse = await
 _amazonCloudWatch.PutMetricAlarmAsync(
```

```
                new PutMetricAlarmRequest()
                {
                    AlarmActions = alarmActions,
                    AlarmDescription = alarmDescription,
                    AlarmName = alarmName,
                    ComparisonOperator = comparison,
                    Threshold = threshold,
                    Namespace = metricNamespace,
                    MetricName = metricName,
                    EvaluationPeriods = 1,
                    Period = 10,
                    Statistic = new Statistic("Maximum"),
                    DatapointsToAlarm = 1,
                    TreatMissingData = "ignore"
                });
            return putEmailAlarmResponse.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (LimitExceededException lex)
        {
            _logger.LogError(lex, $"Unable to add alarm {alarmName}. Alarm quota
 has already been reached.");
        }

        return false;
    }

    /// <summary>
    /// Add specific email actions to a list of action strings for a CloudWatch
 alarm.
    /// </summary>
    /// <param name="accountId">The AccountId for the alarm.</param>
    /// <param name="region">The region for the alarm.</param>
    /// <param name="emailTopicName">An Amazon Simple Notification Service (SNS)
 topic for the alarm email.</param>
    /// <param name="alarmActions">Optional list of existing alarm actions to
 append to.</param>
    /// <returns>A list of string actions for an alarm.</returns>
    public List<string> AddEmailAlarmAction(string accountId, string region,
        string emailTopicName, List<string>? alarmActions = null)
    {
        alarmActions ??= new List<string>();
        var snsAlarmAction = $"arn:aws:sns:{region}:{accountId}:
{emailTopicName}";
        alarmActions.Add(snsAlarmAction);
```

```
        return alarmActions;
    }

    /// <summary>
    /// Describe the current alarms, optionally filtered by state.
    /// </summary>
    /// <param name="stateValue">Optional filter for alarm state.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarms(StateValue? stateValue =
null)
    {
        List<MetricAlarm> alarms = new List<MetricAlarm>();
        var paginatedDescribeAlarms =
_amazonCloudWatch.Paginators.DescribeAlarms(
            new DescribeAlarmsRequest()
            {
                StateValue = stateValue
            });

        await foreach (var data in paginatedDescribeAlarms.MetricAlarms)
        {
            alarms.Add(data);
        }
        return alarms;
    }

    /// <summary>
    /// Describe the current alarms for a specific metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarmsForMetric(string
metricNamespace, string metricName)
    {
        var alarmsResult = await _amazonCloudWatch.DescribeAlarmsForMetricAsync(
            new DescribeAlarmsForMetricRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName
            });

        return alarmsResult.MetricAlarms;
    }
```

```
    /// <summary>
    /// Describe the history of an alarm for a number of days in the past.
    /// </summary>
    /// <param name="alarmName">The name of the alarm.</param>
    /// <param name="historyDays">The number of days in the past.</param>
    /// <returns>The list of alarm history data.</returns>
    public async Task<List<AlarmHistoryItem>> DescribeAlarmHistory(string
alarmName, int historyDays)
    {
        List<AlarmHistoryItem> alarmHistory = new List<AlarmHistoryItem>();
        var paginatedAlarmHistory =
_amazonCloudWatch.Paginators.DescribeAlarmHistory(
            new DescribeAlarmHistoryRequest()
            {
                AlarmName = alarmName,
                EndDateUtc = DateTime.UtcNow,
                HistoryItemType = HistoryItemType.StateUpdate,
                StartDateUtc = DateTime.UtcNow.AddDays(-historyDays)
            });

        await foreach (var data in paginatedAlarmHistory.AlarmHistoryItems)
        {
            alarmHistory.Add(data);
        }
        return alarmHistory;
    }

    /// <summary>
    /// Delete a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteAlarms(List<string> alarmNames)
    {
        var deleteAlarmsResult = await _amazonCloudWatch.DeleteAlarmsAsync(
            new DeleteAlarmsRequest()
            {
                AlarmNames = alarmNames
            });

        return deleteAlarmsResult.HttpStatusCode == HttpStatusCode.OK;
    }
```

```
    /// <summary>
    /// Disable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DisableAlarmActions(List<string> alarmNames)
    {
        var disableAlarmActionsResult = await
_amazonCloudWatch.DisableAlarmActionsAsync(
            new DisableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return disableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Enable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableAlarmActions(List<string> alarmNames)
    {
        var enableAlarmActionsResult = await
_amazonCloudWatch.EnableAlarmActionsAsync(
            new EnableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return enableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Add an anomaly detector for a single metric.
    /// </summary>
    /// <param name="anomalyDetector">A single metric anomaly detector.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
    {
        var putAlarmDetectorResult = await
_amazonCloudWatch.PutAnomalyDetectorAsync(
```

```
            new PutAnomalyDetectorRequest()
            {
                SingleMetricAnomalyDetector = anomalyDetector
            });

        return putAlarmDetectorResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Describe anomaly detectors for a metric and namespace.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The metric of the anomaly detectors.</param>
    /// <returns>The list of detectors.</returns>
    public async Task<List<AnomalyDetector>> DescribeAnomalyDetectors(string
metricNamespace, string metricName)
    {
        List<AnomalyDetector> detectors = new List<AnomalyDetector>();
        var paginatedDescribeAnomalyDetectors =
_amazonCloudWatch.Paginators.DescribeAnomalyDetectors(
            new DescribeAnomalyDetectorsRequest()
            {
                MetricName = metricName,
                Namespace = metricNamespace
            });

        await foreach (var data in
paginatedDescribeAnomalyDetectors.AnomalyDetectors)
        {
            detectors.Add(data);
        }

        return detectors;
    }

    /// <summary>
    /// Delete a single metric anomaly detector.
    /// </summary>
    /// <param name="anomalyDetector">The anomaly detector to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
    {
```

```
        var deleteAnomalyDetectorResponse = await
 _amazonCloudWatch.DeleteAnomalyDetectorAsync(
            new DeleteAnomalyDetectorRequest()
            {
                SingleMetricAnomalyDetector = anomalyDetector
            });

        return deleteAnomalyDetectorResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete a list of CloudWatch dashboards.
    /// </summary>
    /// <param name="dashboardNames">List of dashboard names to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDashboards(List<string> dashboardNames)
    {
        var deleteDashboardsResponse = await
 _amazonCloudWatch.DeleteDashboardsAsync(
            new DeleteDashboardsRequest()
            {
                DashboardNames = dashboardNames
            });

        return deleteDashboardsResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for .NET .
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData
  - GetMetricStatistics

- GetMetricWidgetImage
- ListMetrics
- PutAnomalyDetector
- PutDashboard
- PutMetricAlarm
- PutMetricData

Java

SDK for Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
import com.fasterxml.jackson.core.JsonFactory;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.AlarmHistoryItem;
import software.amazon.awssdk.services.cloudwatch.model.AlarmType;
import software.amazon.awssdk.services.cloudwatch.model.AnomalyDetector;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ComparisonOperator;
import
 software.amazon.awssdk.services.cloudwatch.model.DashboardValidationMessage;
import software.amazon.awssdk.services.cloudwatch.model.Datapoint;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.DeleteAnomalyDetectorRequest;
import software.amazon.awssdk.services.cloudwatch.model.DeleteDashboardsRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmHistoryRequest;
```

```
import
 software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmHistoryResponse;
import
 software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsForMetricRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsForMetricResponse;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsResponse;
import
 software.amazon.awssdk.services.cloudwatch.model.DescribeAnomalyDetectorsRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.DescribeAnomalyDetectorsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricDataResponse;
import
 software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsResponse;
import
 software.amazon.awssdk.services.cloudwatch.model.GetMetricWidgetImageRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.GetMetricWidgetImageResponse;
import software.amazon.awssdk.services.cloudwatch.model.HistoryItemType;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;
import software.amazon.awssdk.services.cloudwatch.model.MetricAlarm;
import software.amazon.awssdk.services.cloudwatch.model.MetricDataQuery;
import software.amazon.awssdk.services.cloudwatch.model.MetricDataResult;
import software.amazon.awssdk.services.cloudwatch.model.MetricDatum;
import software.amazon.awssdk.services.cloudwatch.model.MetricStat;
import
 software.amazon.awssdk.services.cloudwatch.model.PutAnomalyDetectorRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardResponse;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricAlarmRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.ScanBy;
import
 software.amazon.awssdk.services.cloudwatch.model.SingleMetricAnomalyDetector;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.Statistic;
```

```
import
 software.amazon.awssdk.services.cloudwatch.paginators.ListDashboardsIterable;
import software.amazon.awssdk.services.cloudwatch.paginators.ListMetricsIterable;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * To enable billing metrics and statistics for this example, make sure billing
 * alerts are enabled for your account:
 * https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics
 *
 * This Java code example performs the following tasks:
 *
 * 1. List available namespaces from Amazon CloudWatch.
 * 2. List available metrics within the selected Namespace.
 * 3. Get statistics for the selected metric over the last day.
 * 4. Get CloudWatch estimated billing for the last week.
 * 5. Create a new CloudWatch dashboard with metrics.
 * 6. List dashboards using a paginator.
 * 7. Create a new custom metric by adding data for it.
 * 8. Add the custom metric to the dashboard.
 * 9. Create an alarm for the custom metric.
```

```
 * 10. Describe current alarms.
 * 11. Get current data for the new custom metric.
 * 12. Push data into the custom metric to trigger the alarm.
 * 13. Check the alarm state using the action DescribeAlarmsForMetric.
 * 14. Get alarm history for the new alarm.
 * 15. Add an anomaly detector for the custom metric.
 * 16. Describe current anomaly detectors.
 * 17. Get a metric image for the custom metric.
 * 18. Clean up the Amazon CloudWatch resources.
 */
public class CloudWatchScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws IOException {
        final String usage = """

                Usage:
                  <myDate> <costDateWeek> <dashboardName> <dashboardJson>
 <dashboardAdd> <settings> <metricImage> \s

                Where:
                  myDate - The start date to use to get metric statistics. (For
 example, 2023-01-11T18:35:24.00Z.)\s
                  costDateWeek - The start date to use to get AWS/Billinget
 statistics. (For example, 2023-01-11T18:35:24.00Z.)\s
                  dashboardName - The name of the dashboard to create.\s
                  dashboardJson - The location of a JSON file to use to create a
 dashboard. (See Readme file.)\s
                  dashboardAdd - The location of a JSON file to use to update a
 dashboard. (See Readme file.)\s
                  settings - The location of a JSON file from which various
 values are read. (See Readme file.)\s
                  metricImage - The location of a BMP file that is used to create
 a graph.\s
                """;

        if (args.length != 7) {
            System.out.println(usage);
            System.exit(1);
        }

        Region region = Region.US_EAST_1;
        String myDate = args[0];
```

```
        String costDateWeek = args[1];
        String dashboardName = args[2];
        String dashboardJson = args[3];
        String dashboardAdd = args[4];
        String settings = args[5];
        String metricImage = args[6];

        Double dataPoint = Double.parseDouble("10.0");
        Scanner sc = new Scanner(System.in);
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon CloudWatch example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "1. List at least five available unique namespaces from Amazon
CloudWatch. Select one from the list.");
        ArrayList<String> list = listNameSpaces(cw);
        for (int z = 0; z < 5; z++) {
            int index = z + 1;
            System.out.println("    " + index + ". " + list.get(z));
        }

        String selectedNamespace = "";
        String selectedMetrics = "";
        int num = Integer.parseInt(sc.nextLine());
        if (1 <= num && num <= 5) {
            selectedNamespace = list.get(num - 1);
        } else {
            System.out.println("You did not select a valid option.");
            System.exit(1);
        }
        System.out.println("You selected " + selectedNamespace);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. List available metrics within the selected
namespace and select one from the list.");
        ArrayList<String> metList = listMets(cw, selectedNamespace);
```

```
        for (int z = 0; z < 5; z++) {
            int index = z + 1;
            System.out.println("    " + index + ". " + metList.get(z));
        }
        num = Integer.parseInt(sc.nextLine());
        if (1 <= num && num <= 5) {
            selectedMetrics = metList.get(num - 1);
        } else {
            System.out.println("You did not select a valid option.");
            System.exit(1);
        }
        System.out.println("You selected " + selectedMetrics);
        Dimension myDimension = getSpecificMet(cw, selectedNamespace);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get statistics for the selected metric over the
last day.");
        String metricOption = "";
        ArrayList<String> statTypes = new ArrayList<>();
        statTypes.add("SampleCount");
        statTypes.add("Average");
        statTypes.add("Sum");
        statTypes.add("Minimum");
        statTypes.add("Maximum");

        for (int t = 0; t < 5; t++) {
            System.out.println("    " + (t + 1) + ". " + statTypes.get(t));
        }
        System.out.println("Select a metric statistic by entering a number from
the preceding list:");
        num = Integer.parseInt(sc.nextLine());
        if (1 <= num && num <= 5) {
            metricOption = statTypes.get(num - 1);
        } else {
            System.out.println("You did not select a valid option.");
            System.exit(1);
        }
        System.out.println("You selected " + metricOption);
        getAndDisplayMetricStatistics(cw, selectedNamespace, selectedMetrics,
metricOption, myDate, myDimension);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("4. Get CloudWatch estimated billing for the last
week.");
        getMetricStatistics(cw, costDateWeek);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create a new CloudWatch dashboard with metrics.");
        createDashboardWithMetrics(cw, dashboardName, dashboardJson);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. List dashboards using a paginator.");
        listDashboards(cw);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create a new custom metric by adding data to
it.");
        createNewCustomMetric(cw, dataPoint);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Add an additional metric to the dashboard.");
        addMetricToDashboard(cw, dashboardAdd, dashboardName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Create an alarm for the custom metric.");
        String alarmName = createAlarm(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Describe ten current alarms.");
        describeAlarms(cw);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Get current data for new custom metric.");
        getCustomMetricData(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Push data into the custom metric to trigger the
alarm.");
```

```
        addMetricDataForAlarm(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Check the alarm state using the action
DescribeAlarmsForMetric.");
        checkForMetricAlarm(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Get alarm history for the new alarm.");
        getAlarmHistory(cw, settings, myDate);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Add an anomaly detector for the custom metric.");
        addAnomalyDetector(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Describe current anomaly detectors.");
        describeAnomalyDetectors(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Get a metric image for the custom metric.");
        getAndOpenMetricImage(cw, metricImage);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Clean up the Amazon CloudWatch resources.");
        deleteDashboard(cw, dashboardName);
        deleteCWAlarm(cw, alarmName);
        deleteAnomalyDetector(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Amazon CloudWatch example scenario is
complete.");
        System.out.println(DASHES);
        cw.close();
    }
```

```
   public static void deleteAnomalyDetector(CloudWatchClient cw, String
fileName) {
       try {
           // Read values from the JSON file.
           JsonParser parser = new JsonFactory().createParser(new
File(fileName));
           com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
           String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
           String customMetricName =
rootNode.findValue("customMetricName").asText();

           SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                   .metricName(customMetricName)
                   .namespace(customMetricNamespace)
                   .stat("Maximum")
                   .build();

           DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
                   .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                   .build();

           cw.deleteAnomalyDetector(request);
           System.out.println("Successfully deleted the Anomaly Detector.");

       } catch (CloudWatchException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       } catch (IOException e) {
           e.printStackTrace();
       }
   }

   public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
       try {
           DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
                   .alarmNames(alarmName)
                   .build();

           cw.deleteAlarms(request);
           System.out.println("Successfully deleted alarm " + alarmName);
```

```
        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteDashboard(CloudWatchClient cw, String dashboardName)
{
        try {
            DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
                    .dashboardNames(dashboardName)
                    .build();
            cw.deleteDashboards(dashboardsRequest);
            System.out.println(dashboardName + " was successfully deleted.");

        } catch (CloudWatchException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getAndOpenMetricImage(CloudWatchClient cw, String
fileName) {
        System.out.println("Getting Image data for custom metric.");
        try {
            String myJSON = "{\n" +
                    "  \"title\": \"Example Metric Graph\",\n" +
                    "  \"view\": \"timeSeries\",\n" +
                    "  \"stacked \": false,\n" +
                    "  \"period\": 10,\n" +
                    "  \"width\": 1400,\n" +
                    "  \"height\": 600,\n" +
                    "  \"metrics\": [\n" +
                    "    [\n" +
                    "      \"AWS/Billing\",\n" +
                    "      \"EstimatedCharges\",\n" +
                    "      \"Currency\",\n" +
                    "      \"USD\"\n" +
                    "    ]\n" +
                    "  ]\n" +
                    "}";
```

```
            GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
                    .metricWidget(myJSON)
                    .build();

            GetMetricWidgetImageResponse response =
cw.getMetricWidgetImage(imageRequest);
            SdkBytes sdkBytes = response.metricWidgetImage();
            byte[] bytes = sdkBytes.asByteArray();
            File outputFile = new File(fileName);
            try (FileOutputStream outputStream = new
FileOutputStream(outputFile)) {
                outputStream.write(bytes);
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void describeAnomalyDetectors(CloudWatchClient cw, String
fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();
            DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
                    .maxResults(10)
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

            DescribeAnomalyDetectorsResponse response =
cw.describeAnomalyDetectors(detectorsRequest);
            List<AnomalyDetector> anomalyDetectorList =
response.anomalyDetectors();
```

```
            for (AnomalyDetector detector : anomalyDetectorList) {
                System.out.println("Metric name: " +
detector.singleMetricAnomalyDetector().metricName());
                System.out.println("State: " + detector.stateValue());
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .stat("Maximum")
                    .build();

            PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
                    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                    .build();

            cw.putAnomalyDetector(anomalyDetectorRequest);
            System.out.println("Added anomaly detector for metric " +
customMetricName + ".");

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

```
   }

   public static void getAlarmHistory(CloudWatchClient cw, String fileName,
String date) {
       try {
           // Read values from the JSON file.
           JsonParser parser = new JsonFactory().createParser(new
File(fileName));
           com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
           String alarmName = rootNode.findValue("exampleAlarmName").asText();

           Instant start = Instant.parse(date);
           Instant endDate = Instant.now();
           DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
                   .startDate(start)
                   .endDate(endDate)
                   .alarmName(alarmName)
                   .historyItemType(HistoryItemType.ACTION)
                   .build();

           DescribeAlarmHistoryResponse response =
cw.describeAlarmHistory(historyRequest);
           List<AlarmHistoryItem> historyItems = response.alarmHistoryItems();
           if (historyItems.isEmpty()) {
               System.out.println("No alarm history data found for " + alarmName
+ ".");
           } else {
               for (AlarmHistoryItem item : historyItems) {
                   System.out.println("History summary: " +
item.historySummary());
                   System.out.println("Time stamp: " + item.timestamp());
               }
           }

       } catch (CloudWatchException | IOException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }

   public static void checkForMetricAlarm(CloudWatchClient cw, String fileName)
{
```

```
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();
            boolean hasAlarm = false;
            int retries = 10;

            DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

            while (!hasAlarm && retries > 0) {
                DescribeAlarmsForMetricResponse response =
cw.describeAlarmsForMetric(metricRequest);
                hasAlarm = response.hasMetricAlarms();
                retries--;
                Thread.sleep(20000);
                System.out.println(".");
            }
            if (!hasAlarm)
                System.out.println("No Alarm state found for " + customMetricName
+ " after 10 retries.");
            else
                System.out.println("Alarm state found for " + customMetricName +
".");

        } catch (CloudWatchException | IOException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void addMetricDataForAlarm(CloudWatchClient cw, String
fileName) {
        try {
            // Read values from the JSON file.
```

```
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            // Set an Instant object.
            String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
            Instant instant = Instant.parse(time);

            MetricDatum datum = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1001.00)
                    .timestamp(instant)
                    .build();

            MetricDatum datum2 = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1002.00)
                    .timestamp(instant)
                    .build();

            List<MetricDatum> metricDataList = new ArrayList<>();
            metricDataList.add(datum);
            metricDataList.add(datum2);

            PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace(customMetricNamespace)
                    .metricData(metricDataList)
                    .build();

            cw.putMetricData(request);
            System.out.println("Added metric values for for metric " +
customMetricName);

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
```

```
        }
    }

    public static void getCustomMetricData(CloudWatchClient cw, String fileName)
{
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            // Set the date.
            Instant nowDate = Instant.now();

            long hours = 1;
            long minutes = 30;
            Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
                    ChronoUnit.MINUTES);

            Metric met = Metric.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

            MetricStat metStat = MetricStat.builder()
                    .stat("Maximum")
                    .period(1)
                    .metric(met)
                    .build();

            MetricDataQuery dataQUery = MetricDataQuery.builder()
                    .metricStat(metStat)
                    .id("foo2")
                    .returnData(true)
                    .build();

            List<MetricDataQuery> dq = new ArrayList<>();
            dq.add(dataQUery);
```

```
            GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
                    .maxDatapoints(10)
                    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
                    .startTime(nowDate)
                    .endTime(date2)
                    .metricDataQueries(dq)
                    .build();

            GetMetricDataResponse response = cw.getMetricData(getMetReq);
            List<MetricDataResult> data = response.metricDataResults();
            for (MetricDataResult item : data) {
                System.out.println("The label is " + item.label());
                System.out.println("The status code is " +
item.statusCode().toString());
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void describeAlarms(CloudWatchClient cw) {
        try {
            List<AlarmType> typeList = new ArrayList<>();
            typeList.add(AlarmType.METRIC_ALARM);

            DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
                    .alarmTypes(typeList)
                    .maxRecords(10)
                    .build();

            DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
            List<MetricAlarm> alarmList = response.metricAlarms();
            for (MetricAlarm alarm : alarmList) {
                System.out.println("Alarm name: " + alarm.alarmName());
                System.out.println("Alarm description: " +
alarm.alarmDescription());
            }
        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
   public static String createAlarm(CloudWatchClient cw, String fileName) {
       try {
           // Read values from the JSON file.
           JsonParser parser = new JsonFactory().createParser(new
File(fileName));
           com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
           String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
           String customMetricName =
rootNode.findValue("customMetricName").asText();
           String alarmName = rootNode.findValue("exampleAlarmName").asText();
           String emailTopic = rootNode.findValue("emailTopic").asText();
           String accountId = rootNode.findValue("accountId").asText();
           String region = rootNode.findValue("region").asText();

           // Create a List for alarm actions.
           List<String> alarmActions = new ArrayList<>();
           alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);
           PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
                   .alarmActions(alarmActions)
                   .alarmDescription("Example metric alarm")
                   .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
                   .threshold(100.00)
                   .metricName(customMetricName)
                   .namespace(customMetricNamespace)
                   .evaluationPeriods(1)
                   .period(10)
                   .statistic("Maximum")
                   .datapointsToAlarm(1)
                   .treatMissingData("ignore")
                   .build();

           cw.putMetricAlarm(alarmRequest);
           System.out.println(alarmName + " was successfully created!");
           return alarmName;

       } catch (CloudWatchException | IOException e) {
           System.err.println(e.getMessage());
           System.exit(1);
```

```
        }
        return "";
    }

    public static void addMetricToDashboard(CloudWatchClient cw, String fileName,
String dashboardName) {
        try {
            PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
                    .dashboardName(dashboardName)
                    .dashboardBody(readFileAsString(fileName))
                    .build();

            cw.putDashboard(dashboardRequest);
            System.out.println(dashboardName + " was successfully updated.");

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void createNewCustomMetric(CloudWatchClient cw, Double
dataPoint) {
        try {
            Dimension dimension = Dimension.builder()
                    .name("UNIQUE_PAGES")
                    .value("URLS")
                    .build();

            // Set an Instant object.
            String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
            Instant instant = Instant.parse(time);

            MetricDatum datum = MetricDatum.builder()
                    .metricName("PAGES_VISITED")
                    .unit(StandardUnit.NONE)
                    .value(dataPoint)
                    .timestamp(instant)
                    .dimensions(dimension)
                    .build();

            PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace("SITE/TRAFFIC")
```

```
                    .metricData(datum)
                    .build();

            cw.putMetricData(request);
            System.out.println("Added metric values for for metric
PAGES_VISITED");

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listDashboards(CloudWatchClient cw) {
        try {
            ListDashboardsIterable listRes = cw.listDashboardsPaginator();
            listRes.stream()
                    .flatMap(r -> r.dashboardEntries().stream())
                    .forEach(entry -> {
                        System.out.println("Dashboard name is: " +
entry.dashboardName());
                        System.out.println("Dashboard ARN is: " +
entry.dashboardArn());
                    });

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createDashboardWithMetrics(CloudWatchClient cw, String
dashboardName, String fileName) {
        try {
            PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
                    .dashboardName(dashboardName)
                    .dashboardBody(readFileAsString(fileName))
                    .build();

            PutDashboardResponse response = cw.putDashboard(dashboardRequest);
            System.out.println(dashboardName + " was successfully created.");
            List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
            if (messages.isEmpty()) {
```

```
                System.out.println("There are no messages in the new Dashboard");
            } else {
                for (DashboardValidationMessage message : messages) {
                    System.out.println("Message is: " + message.message());
                }
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String readFileAsString(String file) throws IOException {
        return new String(Files.readAllBytes(Paths.get(file)));
    }

    public static void getMetricStatistics(CloudWatchClient cw, String
costDateWeek) {
        try {
            Instant start = Instant.parse(costDateWeek);
            Instant endDate = Instant.now();
            Dimension dimension = Dimension.builder()
                    .name("Currency")
                    .value("USD")
                    .build();

            List<Dimension> dimensionList = new ArrayList<>();
            dimensionList.add(dimension);
            GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
                    .metricName("EstimatedCharges")
                    .namespace("AWS/Billing")
                    .dimensions(dimensionList)
                    .statistics(Statistic.MAXIMUM)
                    .startTime(start)
                    .endTime(endDate)
                    .period(86400)
                    .build();

            GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
            List<Datapoint> data = response.datapoints();
            if (!data.isEmpty()) {
```

```
                for (Datapoint datapoint : data) {
                    System.out
                            .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
                }
            } else {
                System.out.println("The returned data list is empty");
            }

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getAndDisplayMetricStatistics(CloudWatchClient cw, String
nameSpace, String metVal,
            String metricOption, String date, Dimension myDimension) {
        try {
            Instant start = Instant.parse(date);
            Instant endDate = Instant.now();

            GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
                    .endTime(endDate)
                    .startTime(start)
                    .dimensions(myDimension)
                    .metricName(metVal)
                    .namespace(nameSpace)
                    .period(86400)
                    .statistics(Statistic.fromValue(metricOption))
                    .build();

            GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
            List<Datapoint> data = response.datapoints();
            if (!data.isEmpty()) {
                for (Datapoint datapoint : data) {
                    System.out
                            .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
                }
            } else {
                System.out.println("The returned data list is empty");
```

```
            }

        } catch (CloudWatchException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static Dimension getSpecificMet(CloudWatchClient cw, String namespace)
{
        try {
            ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

            ListMetricsResponse response = cw.listMetrics(request);
            List<Metric> myList = response.metrics();
            Metric metric = myList.get(0);
            return metric.dimensions().get(0);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static ArrayList<String> listMets(CloudWatchClient cw, String
namespace) {
        try {
            ArrayList<String> metList = new ArrayList<>();
            ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

            ListMetricsIterable listRes = cw.listMetricsPaginator(request);
            listRes.stream()
                    .flatMap(r -> r.metrics().stream())
                    .forEach(metrics -> metList.add(metrics.metricName()));

            return metList;

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        return null;
    }

    public static ArrayList<String> listNameSpaces(CloudWatchClient cw) {
        try {
            ArrayList<String> nameSpaceList = new ArrayList<>();
            ListMetricsRequest request = ListMetricsRequest.builder()
                    .build();

            ListMetricsIterable listRes = cw.listMetricsPaginator(request);
            listRes.stream()
                    .flatMap(r -> r.metrics().stream())
                    .forEach(metrics -> {
                        String data = metrics.namespace();
                        if (!nameSpaceList.contains(data)) {
                            nameSpaceList.add(data);
                        }
                    });

            return nameSpaceList;
        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Java 2.x .
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData

- GetMetricStatistics
- GetMetricWidgetImage
- ListMetrics
- PutAnomalyDetector
- PutDashboard
- PutMetricAlarm
- PutMetricData

Kotlin

SDK for Kotlin

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 To enable billing metrics and statistics for this example, make sure billing
 alerts are enabled for your account:
 https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics

 This Kotlin code example performs the following tasks:

 1. List available namespaces from Amazon CloudWatch. Select a namespace from the
 list.
 2. List available metrics within the selected namespace.
 3. Get statistics for the selected metric over the last day.
 4. Get CloudWatch estimated billing for the last week.
 5. Create a new CloudWatch dashboard with metrics.
 6. List dashboards using a paginator.
```

```
 7. Create a new custom metric by adding data for it.
 8. Add the custom metric to the dashboard.
 9. Create an alarm for the custom metric.
 10. Describe current alarms.
 11. Get current data for the new custom metric.
 12. Push data into the custom metric to trigger the alarm.
 13. Check the alarm state using the action DescribeAlarmsForMetric.
 14. Get alarm history for the new alarm.
 15. Add an anomaly detector for the custom metric.
 16. Describe current anomaly detectors.
 17. Get a metric image for the custom metric.
 18. Clean up the Amazon CloudWatch resources.
 */

val DASHES: String? = String(CharArray(80)).replace("\u0000", "-")
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <myDate> <costDateWeek> <dashboardName> <dashboardJson>
 <dashboardAdd> <settings> <metricImage>

        Where:
            myDate - The start date to use to get metric statistics. (For
 example, 2023-01-11T18:35:24.00Z.)
            costDateWeek - The start date to use to get AWS Billing and Cost
 Management statistics. (For example, 2023-01-11T18:35:24.00Z.)
            dashboardName - The name of the dashboard to create.
            dashboardJson - The location of a JSON file to use to create a
 dashboard. (See Readme file.)
            dashboardAdd - The location of a JSON file to use to update a
 dashboard. (See Readme file.)
            settings - The location of a JSON file from which various values are
 read. (See Readme file.)
            metricImage - The location of a BMP file that is used to create a
 graph.
    """

    if (args.size != 7) {
        println(usage)
        System.exit(1)
    }

    val myDate = args[0]
    val costDateWeek = args[1]
```

```
    val dashboardName = args[2]
    val dashboardJson = args[3]
    val dashboardAdd = args[4]
    val settings = args[5]
    var metricImage = args[6]
    val dataPoint = "10.0".toDouble()
    val inOb = Scanner(System.`in`)

    println(DASHES)
    println("Welcome to the Amazon CloudWatch example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. List at least five available unique namespaces from Amazon
CloudWatch. Select a CloudWatch namespace from the list.")
    val list: ArrayList<String> = listNameSpaces()
    for (z in 0..4) {
        println("    ${z + 1}. ${list[z]}")
    }

    var selectedNamespace: String
    var selectedMetrics = ""
    var num = inOb.nextLine().toInt()
    println("You selected $num")

    if (1 <= num && num <= 5) {
        selectedNamespace = list[num - 1]
    } else {
        println("You did not select a valid option.")
        exitProcess(1)
    }
    println("You selected $selectedNamespace")
    println(DASHES)

    println(DASHES)
    println("2. List available metrics within the selected namespace and select
one from the list.")
    val metList = listMets(selectedNamespace)
    for (z in 0..4) {
        println("    ${ z + 1}. ${metList?.get(z)}")
    }
    num = inOb.nextLine().toInt()
    if (1 <= num && num <= 5) {
        selectedMetrics = metList!![num - 1]
```

```
    } else {
        println("You did not select a valid option.")
        System.exit(1)
    }
    println("You selected $selectedMetrics")
    val myDimension = getSpecificMet(selectedNamespace)
    if (myDimension == null) {
        println("Error - Dimension is null")
        exitProcess(1)
    }
    println(DASHES)

    println(DASHES)
    println("3. Get statistics for the selected metric over the last day.")
    val metricOption: String
    val statTypes = ArrayList<String>()
    statTypes.add("SampleCount")
    statTypes.add("Average")
    statTypes.add("Sum")
    statTypes.add("Minimum")
    statTypes.add("Maximum")

    for (t in 0..4) {
        println("    ${t + 1}. ${statTypes[t]}")
    }
    println("Select a metric statistic by entering a number from the preceding
list:")
    num = inOb.nextLine().toInt()
    if (1 <= num && num <= 5) {
        metricOption = statTypes[num - 1]
    } else {
        println("You did not select a valid option.")
        exitProcess(1)
    }
    println("You selected $metricOption")
    getAndDisplayMetricStatistics(selectedNamespace, selectedMetrics,
metricOption, myDate, myDimension)
    println(DASHES)

    println(DASHES)
    println("4. Get CloudWatch estimated billing for the last week.")
    getMetricStatistics(costDateWeek)
    println(DASHES)
```

```
    println(DASHES)
    println("5. Create a new CloudWatch dashboard with metrics.")
    createDashboardWithMetrics(dashboardName, dashboardJson)
    println(DASHES)

    println(DASHES)
    println("6. List dashboards using a paginator.")
    listDashboards()
    println(DASHES)

    println(DASHES)
    println("7. Create a new custom metric by adding data to it.")
    createNewCustomMetric(dataPoint)
    println(DASHES)

    println(DASHES)
    println("8. Add an additional metric to the dashboard.")
    addMetricToDashboard(dashboardAdd, dashboardName)
    println(DASHES)

    println(DASHES)
    println("9. Create an alarm for the custom metric.")
    val alarmName: String = createAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("10. Describe 10 current alarms.")
    describeAlarms()
    println(DASHES)

    println(DASHES)
    println("11. Get current data for the new custom metric.")
    getCustomMetricData(settings)
    println(DASHES)

    println(DASHES)
    println("12. Push data into the custom metric to trigger the alarm.")
    addMetricDataForAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("13. Check the alarm state using the action
DescribeAlarmsForMetric.")
    checkForMetricAlarm(settings)
```

```
    println(DASHES)

    println(DASHES)
    println("14. Get alarm history for the new alarm.")
    getAlarmHistory(settings, myDate)
    println(DASHES)

    println(DASHES)
    println("15. Add an anomaly detector for the custom metric.")
    addAnomalyDetector(settings)
    println(DASHES)

    println(DASHES)
    println("16. Describe current anomaly detectors.")
    describeAnomalyDetectors(settings)
    println(DASHES)

    println(DASHES)
    println("17. Get a metric image for the custom metric.")
    getAndOpenMetricImage(metricImage)
    println(DASHES)

    println(DASHES)
    println("18. Clean up the Amazon CloudWatch resources.")
    deleteDashboard(dashboardName)
    deleteAlarm(alarmName)
    deleteAnomalyDetector(settings)
    println(DASHES)

    println(DASHES)
    println("The Amazon CloudWatch example scenario is complete.")
    println(DASHES)
}

suspend fun deleteAnomalyDetector(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal = SingleMetricAnomalyDetector {
        metricName = customMetricName
```

```
        namespace = customMetricNamespace
        stat = "Maximum"
    }

    val request = DeleteAnomalyDetectorRequest {
        singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAnomalyDetector(request)
        println("Successfully deleted the Anomaly Detector.")
    }
}

suspend fun deleteAlarm(alarmNameVal: String) {
    val request = DeleteAlarmsRequest {
        alarmNames = listOf(alarmNameVal)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAlarms(request)
        println("Successfully deleted alarm $alarmNameVal")
    }
}

suspend fun deleteDashboard(dashboardName: String) {
    val dashboardsRequest = DeleteDashboardsRequest {
        dashboardNames = listOf(dashboardName)
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteDashboards(dashboardsRequest)
        println("$dashboardName was successfully deleted.")
    }
}

suspend fun getAndOpenMetricImage(fileName: String) {
    println("Getting Image data for custom metric.")
    val myJSON = """{
        "title": "Example Metric Graph",
        "view": "timeSeries",
        "stacked ": false,
        "period": 10,
        "width": 1400,
        "height": 600,
```

```
        "metrics": [
            [
            "AWS/Billing",
            "EstimatedCharges",
            "Currency",
            "USD"
            ]
        ]
        }"""

    val imageRequest = GetMetricWidgetImageRequest {
        metricWidget = myJSON
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricWidgetImage(imageRequest)
        val bytes = response.metricWidgetImage
        if (bytes != null) {
            File(fileName).writeBytes(bytes)
        }
    }
    println("You have successfully written data to $fileName")
}

suspend fun describeAnomalyDetectors(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val detectorsRequest = DescribeAnomalyDetectorsRequest {
        maxResults = 10
        metricName = customMetricName
        namespace = customMetricNamespace
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAnomalyDetectors(detectorsRequest)
        response.anomalyDetectors?.forEach { detector ->
            println("Metric name:
 ${detector.singleMetricAnomalyDetector?.metricName}")
            println("State: ${detector.stateValue}")
        }
```

```
    }
}

suspend fun addAnomalyDetector(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal = SingleMetricAnomalyDetector {
        metricName = customMetricName
        namespace = customMetricNamespace
        stat = "Maximum"
    }

    val anomalyDetectorRequest = PutAnomalyDetectorRequest {
        singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putAnomalyDetector(anomalyDetectorRequest)
        println("Added anomaly detector for metric $customMetricName.")
    }
}

suspend fun getAlarmHistory(fileName: String, date: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val start = Instant.parse(date)
    val endDateVal = Instant.now()

    val historyRequest = DescribeAlarmHistoryRequest {
        startDate = aws.smithy.kotlin.runtime.time.Instant(start)
        endDate = aws.smithy.kotlin.runtime.time.Instant(endDateVal)
        alarmName = alarmNameVal
        historyItemType = HistoryItemType.Action
    }

    CloudWatchClient { credentialsProvider = EnvironmentCredentialsProvider();
 region = "us-east-1" }.use { cwClient ->
```

```
        val response = cwClient.describeAlarmHistory(historyRequest)
        val historyItems = response.alarmHistoryItems
        if (historyItems != null) {
            if (historyItems.isEmpty()) {
                println("No alarm history data found for $alarmNameVal.")
            } else {
                for (item in historyItems) {
                    println("History summary ${item.historySummary}")
                    println("Time stamp: ${item.timestamp}")
                }
            }
        }
    }
}

suspend fun checkForMetricAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    var hasAlarm = false
    var retries = 10

    val metricRequest = DescribeAlarmsForMetricRequest {
        metricName = customMetricName
        namespace = customMetricNamespace
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        while (!hasAlarm && retries > 0) {
            val response = cwClient.describeAlarmsForMetric(metricRequest)
            if (response.metricAlarms?.count()!! > 0) {
                hasAlarm = true
            }
            retries--
            delay(20000)
            println(".")
        }
        if (!hasAlarm) println("No Alarm state found for $customMetricName after
 10 retries.") else println("Alarm state found for $customMetricName.")
    }
}
```

```
suspend fun addMetricDataForAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum = MetricDatum {
        metricName = customMetricName
        unit = StandardUnit.None
        value = 1001.00
        timestamp = aws.smithy.kotlin.runtime.time.Instant(instant)
    }

    val datum2 = MetricDatum {
        metricName = customMetricName
        unit = StandardUnit.None
        value = 1002.00
        timestamp = aws.smithy.kotlin.runtime.time.Instant(instant)
    }

    val metricDataList = ArrayList<MetricDatum>()
    metricDataList.add(datum)
    metricDataList.add(datum2)

    val request = PutMetricDataRequest {
        namespace = customMetricNamespace
        metricData = metricDataList
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric $customMetricName")
    }
}

suspend fun getCustomMetricData(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
```

```
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set the date.
    val nowDate = Instant.now()
    val hours: Long = 1
    val minutes: Long = 30
    val date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(
        minutes,
        ChronoUnit.MINUTES
    )

    val met = Metric {
        metricName = customMetricName
        namespace = customMetricNamespace
    }

    val metStat = MetricStat {
        stat = "Maximum"
        period = 1
        metric = met
    }

    val dataQUery = MetricDataQuery {
        metricStat = metStat
        id = "foo2"
        returnData = true
    }

    val dq = ArrayList<MetricDataQuery>()
    dq.add(dataQUery)
    val getMetReq = GetMetricDataRequest {
        maxDatapoints = 10
        scanBy = ScanBy.TimestampDescending
        startTime = aws.smithy.kotlin.runtime.time.Instant(nowDate)
        endTime = aws.smithy.kotlin.runtime.time.Instant(date2)
        metricDataQueries = dq
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricData(getMetReq)
        response.metricDataResults?.forEach { item ->
```

```
            println("The label is ${item.label}")
            println("The status code is ${item.statusCode}")
        }
    }
}

suspend fun describeAlarms() {
    val typeList = ArrayList<AlarmType>()
    typeList.add(AlarmType.MetricAlarm)
    val alarmsRequest = DescribeAlarmsRequest {
        alarmTypes = typeList
        maxRecords = 10
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarms(alarmsRequest)
        response.metricAlarms?.forEach { alarm ->
            println("Alarm name: ${alarm.alarmName}")
            println("Alarm description: ${alarm.alarmDescription}")
        }
    }
}

suspend fun createAlarm(fileName: String): String {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode: JsonNode = ObjectMapper().readTree(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val emailTopic = rootNode.findValue("emailTopic").asText()
    val accountId = rootNode.findValue("accountId").asText()
    val region2 = rootNode.findValue("region").asText()

    // Create a List for alarm actions.
    val alarmActionObs: MutableList<String> = ArrayList()
    alarmActionObs.add("arn:aws:sns:$region2:$accountId:$emailTopic")
    val alarmRequest = PutMetricAlarmRequest {
        alarmActions = alarmActionObs
        alarmDescription = "Example metric alarm"
        alarmName = alarmNameVal
        comparisonOperator = ComparisonOperator.GreaterThanOrEqualToThreshold
        threshold = 100.00
```

```
        metricName = customMetricName
        namespace = customMetricNamespace
        evaluationPeriods = 1
        period = 10
        statistic = Statistic.Maximum
        datapointsToAlarm = 1
        treatMissingData = "ignore"
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(alarmRequest)
        println("$alarmNameVal was successfully created!")
        return alarmNameVal
    }
}

suspend fun addMetricToDashboard(fileNameVal: String, dashboardNameVal: String) {
    val dashboardRequest = PutDashboardRequest {
        dashboardName = dashboardNameVal
        dashboardBody = readFileAsString(fileNameVal)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully updated.")
    }
}

suspend fun createNewCustomMetric(dataPoint: Double) {
    val dimension = Dimension {
        name = "UNIQUE_PAGES"
        value = "URLS"
    }

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum = MetricDatum {
        metricName = "PAGES_VISITED"
        unit = StandardUnit.None
        value = dataPoint
        timestamp = aws.smithy.kotlin.runtime.time.Instant(instant)
        dimensions = listOf(dimension)
```

```
    }

    val request = PutMetricDataRequest {
        namespace = "SITE/TRAFFIC"
        metricData = listOf(datum)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric PAGES_VISITED")
    }
}

suspend fun listDashboards() {
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.listDashboardsPaginated({})
            .transform { it.dashboardEntries?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.dashboardName}")
                println("Dashboard ARN is ${obj.dashboardArn}")
            }
    }
}

suspend fun createDashboardWithMetrics(dashboardNameVal: String, fileNameVal:
 String) {
    val dashboardRequest = PutDashboardRequest {
        dashboardName = dashboardNameVal
        dashboardBody = readFileAsString(fileNameVal)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully created.")
        val messages = response.dashboardValidationMessages
        if (messages != null) {
            if (messages.isEmpty()) {
                println("There are no messages in the new Dashboard")
            } else {
                for (message in messages) {
                    println("Message is: ${message.message}")
                }
            }
        }
```

```
    }
}

fun readFileAsString(file: String): String {
    return String(Files.readAllBytes(Paths.get(file)))
}

suspend fun getMetricStatistics(costDateWeek: String?) {
    val start = Instant.parse(costDateWeek)
    val endDate = Instant.now()
    val dimension = Dimension {
        name = "Currency"
        value = "USD"
    }

    val dimensionList: MutableList<Dimension> = ArrayList()
    dimensionList.add(dimension)

    val statisticsRequest = GetMetricStatisticsRequest {
        metricName = "EstimatedCharges"
        namespace = "AWS/Billing"
        dimensions = dimensionList
        statistics = listOf(Statistic.Maximum)
        startTime = aws.smithy.kotlin.runtime.time.Instant(start)
        endTime = aws.smithy.kotlin.runtime.time.Instant(endDate)
        period = 86400
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data: List<Datapoint>? = response.datapoints
        if (data != null) {
            if (!data.isEmpty()) {
                for (datapoint in data) {
                    println("Timestamp:  ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}
```

```
suspend fun getAndDisplayMetricStatistics(nameSpaceVal: String, metVal: String,
 metricOption: String, date: String, myDimension: Dimension) {
    val start = Instant.parse(date)
    val endDate = Instant.now()
    val statisticsRequest = GetMetricStatisticsRequest {
        endTime = aws.smithy.kotlin.runtime.time.Instant(endDate)
        startTime = aws.smithy.kotlin.runtime.time.Instant(start)
        dimensions = listOf(myDimension)
        metricName = metVal
        namespace = nameSpaceVal
        period = 86400
        statistics = listOf(Statistic.fromValue(metricOption))
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data = response.datapoints
        if (data != null) {
            if (data.isNotEmpty()) {
                for (datapoint in data) {
                    println("Timestamp: ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}

suspend fun listMets(namespaceVal: String?): ArrayList<String>? {
    val metList = ArrayList<String>()
    val request = ListMetricsRequest {
        namespace = namespaceVal
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val reponse = cwClient.listMetrics(request)
        reponse.metrics?.forEach { metrics ->
            val data = metrics.metricName
            if (!metList.contains(data)) {
                metList.add(data!!)
            }
        }
    }
```

```
    return metList
}

suspend fun getSpecificMet(namespaceVal: String?): Dimension? {
    val request = ListMetricsRequest {
        namespace = namespaceVal
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.listMetrics(request)
        val myList = response.metrics
        if (myList != null) {
            return myList[0].dimensions?.get(0)
        }
    }
    return null
}

suspend fun listNameSpaces(): ArrayList<String> {
    val nameSpaceList = ArrayList<String>()
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.listMetrics(ListMetricsRequest {})
        response.metrics?.forEach { metrics ->
            val data = metrics.namespace
            if (!nameSpaceList.contains(data)) {
                nameSpaceList.add(data!!)
            }
        }
    }
    return nameSpaceList
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK For Kotlin.
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors

- GetMetricData
- GetMetricStatistics
- GetMetricWidgetImage
- ListMetrics
- PutAnomalyDetector
- PutDashboard
- PutMetricAlarm
- PutMetricData

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan CloudWatch dengan AWS SDK. Topik ini juga mencakup informasi tentang cara memulai dan detail versi-versi SDK sebelumnya.

Mengelola CloudWatch metrik dan alarm menggunakan SDK AWS

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Buat alarm untuk menonton CloudWatch metrik.
- Memasukkan data ke dalam metrik dan picu alarm.
- Ambil data dari alarm.
- Menghapus alarm.

Python

SDK untuk Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Buat kelas yang membungkus CloudWatch operasi.

```
from datetime import datetime, timedelta
import logging
```

```
from pprint import pprint
import random
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def put_metric_data_set(self, namespace, name, timestamp, unit, data_set):
        """
        Sends a set of data to CloudWatch for a metric. All of the data in the
 set
        have the same timestamp and unit.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param timestamp: The UTC timestamp for the metric.
        :param unit: The unit of the metric.
        :param data_set: The set of data to send. This set is a dictionary that
                         contains a list of values and a list of corresponding
 counts.
                         The value and count lists must be the same length.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[
                    {
                        "MetricName": name,
                        "Timestamp": timestamp,
                        "Values": data_set["values"],
                        "Counts": data_set["counts"],
```

```
                        "Unit": unit,
                    }
                ],
            )
            logger.info("Put data set for metric %s.%s.", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data set for metric %s.%s.",
namespace, name)
            raise


    def create_metric_alarm(
        self,
        metric_namespace,
        metric_name,
        alarm_name,
        stat_type,
        period,
        eval_periods,
        threshold,
        comparison_op,
    ):
        """
        Creates an alarm that watches a metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :param alarm_name: The name of the alarm.
        :param stat_type: The type of statistic the alarm watches.
        :param period: The period in which metric data are grouped to calculate
                       statistics.
        :param eval_periods: The number of periods that the metric must be over
the
                             alarm threshold before the alarm is set into an
alarmed
                             state.
        :param threshold: The threshold value to compare against the metric
statistic.
        :param comparison_op: The comparison operation used to compare the
threshold
                              against the metric.
        :return: The newly created alarm.
        """
        try:
```

```
            metric = self.cloudwatch_resource.Metric(metric_namespace,
metric_name)
            alarm = metric.put_alarm(
                AlarmName=alarm_name,
                Statistic=stat_type,
                Period=period,
                EvaluationPeriods=eval_periods,
                Threshold=threshold,
                ComparisonOperator=comparison_op,
            )
            logger.info(
                "Added alarm %s to track metric %s.%s.",
                alarm_name,
                metric_namespace,
                metric_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't add alarm %s to metric %s.%s",
                alarm_name,
                metric_namespace,
                metric_name,
            )
            raise
        else:
            return alarm


    def put_metric_data(self, namespace, name, value, unit):
        """
        Sends a single data value to CloudWatch for a metric. This metric is
given
        a timestamp of the current UTC time.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param value: The value of the metric.
        :param unit: The unit of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[{"MetricName": name, "Value": value, "Unit": unit}],
```

```
            )
            logger.info("Put data for metric %s.%s", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data for metric %s.%s", namespace,
name)
            raise


    def get_metric_statistics(self, namespace, name, start, end, period,
stat_types):
        """
        Gets statistics for a metric within a specified time span. Metrics are
grouped
        into the specified period.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param start: The UTC start time of the time span to retrieve.
        :param end: The UTC end time of the time span to retrieve.
        :param period: The period, in seconds, in which to group metrics. The
period
                       must match the granularity of the metric, which depends on
                       the metric's age. For example, metrics that are older than
                       three hours have a one-minute granularity, so the period
must
                       be at least 60 and must be a multiple of 60.
        :param stat_types: The type of statistics to retrieve, such as average
value
                           or maximum value.
        :return: The retrieved statistics for the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            stats = metric.get_statistics(
                StartTime=start, EndTime=end, Period=period,
Statistics=stat_types
            )
            logger.info(
                "Got %s statistics for %s.", len(stats["Datapoints"]),
stats["Label"]
            )
        except ClientError:
            logger.exception("Couldn't get statistics for %s.%s.", namespace,
name)
```

```
            raise
        else:
            return stats


    def get_metric_alarms(self, metric_namespace, metric_name):
        """
        Gets the alarms that are currently watching the specified metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :returns: An iterator that yields the alarms.
        """
        metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
        alarm_iter = metric.alarms.all()
        logger.info("Got alarms for metric %s.%s.", metric_namespace,
metric_name)
        return alarm_iter


    def delete_metric_alarms(self, metric_namespace, metric_name):
        """
        Deletes all of the alarms that are currently watching the specified
metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(metric_namespace,
metric_name)
            metric.alarms.delete()
            logger.info(
                "Deleted alarms for metric %s.%s.", metric_namespace, metric_name
            )
        except ClientError:
            logger.exception(
                "Couldn't delete alarms for metric %s.%s.",
                metric_namespace,
                metric_name,
            )
            raise
```

Gunakan kelas pembungkus untuk memasukkan data ke dalam metrik, memicu alarm yang mengawasi metrik, dan ambil data dari alarm.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon CloudWatch metrics and alarms demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    cw_wrapper = CloudWatchWrapper(boto3.resource("cloudwatch"))

    minutes = 20
    metric_namespace = "doc-example-metric"
    metric_name = "page_views"
    start = datetime.utcnow() - timedelta(minutes=minutes)
    print(
        f"Putting data into metric {metric_namespace}.{metric_name} spanning the
 "
        f"last {minutes} minutes."
    )
    for offset in range(0, minutes):
        stamp = start + timedelta(minutes=offset)
        cw_wrapper.put_metric_data_set(
            metric_namespace,
            metric_name,
            stamp,
            "Count",
            {
                "values": [
                    random.randint(bound, bound * 2)
                    for bound in range(offset + 1, offset + 11)
                ],
                "counts": [random.randint(1, offset + 1) for _ in range(10)],
            },
        )

    alarm_name = "high_page_views"
    period = 60
    eval_periods = 2
    print(f"Creating alarm {alarm_name} for metric {metric_name}.")
```

```
    alarm = cw_wrapper.create_metric_alarm(
        metric_namespace,
        metric_name,
        alarm_name,
        "Maximum",
        period,
        eval_periods,
        100,
        "GreaterThanThreshold",
    )
    print(f"Alarm ARN is {alarm.alarm_arn}.")
    print(f"Current alarm state is: {alarm.state_value}.")

    print(
        f"Sending data to trigger the alarm. This requires data over the
threshold "
        f"for {eval_periods} periods of {period} seconds each."
    )
    while alarm.state_value == "INSUFFICIENT_DATA":
        print("Sending data for the metric.")
        cw_wrapper.put_metric_data(
            metric_namespace, metric_name, random.randint(100, 200), "Count"
        )
        alarm.load()
        print(f"Current alarm state is: {alarm.state_value}.")
        if alarm.state_value == "INSUFFICIENT_DATA":
            print(f"Waiting for {period} seconds...")
            time.sleep(period)
        else:
            print("Wait for a minute for eventual consistency of metric data.")
            time.sleep(period)
            if alarm.state_value == "OK":
                alarm.load()
                print(f"Current alarm state is: {alarm.state_value}.")

    print(
        f"Getting data for metric {metric_namespace}.{metric_name} during
timespan "
        f"of {start} to {datetime.utcnow()} (times are UTC)."
    )
    stats = cw_wrapper.get_metric_statistics(
        metric_namespace,
        metric_name,
        start,
```

```
        datetime.utcnow(),
        60,
        ["Average", "Minimum", "Maximum"],
    )
    print(
        f"Got {len(stats['Datapoints'])} data points for metric "
        f"{metric_namespace}.{metric_name}."
    )
    pprint(sorted(stats["Datapoints"], key=lambda x: x["Timestamp"]))

    print(f"Getting alarms for metric {metric_name}.")
    alarms = cw_wrapper.get_metric_alarms(metric_namespace, metric_name)
    for alarm in alarms:
        print(f"Alarm {alarm.name} is currently in state {alarm.state_value}.")

    print(f"Deleting alarms for metric {metric_name}.")
    cw_wrapper.delete_metric_alarms(metric_namespace, metric_name)

    print("Thanks for watching!")
    print("-" * 88)
```

- Untuk mengetahui hal detail mengenai API, silakan lihat topik berikut ini di SDK AWS untuk Referensi API Python (Boto3).
  - DeleteAlarms
  - DescribeAlarmsForMetric
  - DisableAlarmActions
  - EnableAlarmActions
  - GetMetricStatistics
  - ListMetrics
  - PutMetricAlarm
  - PutMetricData

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan CloudWatch dengan AWS SDK. Topik ini juga mencakup informasi tentang cara memulai dan detail versi-versi SDK sebelumnya.

Keamanan di Amazon CloudWatch

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

Keamanan menjadi tanggung jawab bersama antara 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 menjalankan layanan–layanan AWS di dalam AWS Cloud. AWS juga memberikan Anda layanan yang dapat digunakan dengan aman. Auditor pihak ketiga melakukan pengujian dan verifikasi secara berkala terhadap efektivitas keamanan kami sebagai bagian dari Program Kepatuhan AWS . Untuk mempelajari tentang program kepatuhan yang berlaku untuk CloudWatch, silakan lihat Layanan AWS dalam Lingkup Program Kepatuhan .
- Keamanan di cloud – Tanggung jawab Anda ditentukan menurut layanan AWS yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, yang mencakup kepekaan data Anda, persyaratan perusahaan, serta peraturan perundangan yang berlaku

Dokumentasi ini akan membantu Anda dalam memahami cara menerapkan model tanggung jawab bersama saat Anda menggunakan Amazon CloudWatch. Dokumentasi tersebut juga menunjukkan kepada Anda cara mengonfigurasi Amazon CloudWatch untuk memenuhi tujuan-tujuan keamanan dan kepatuhan Anda. Anda juga akan mempelajari cara menggunakan layanan AWS lain yang dapat membantu Anda memantau dan mengamankan sumber daya CloudWatch Anda.

Daftar Isi

Perlindungan data di Amazon CloudWatch

Model tanggung jawab bersama AWS berlaku terhadap perlindungan data dalam Amazon CloudWatch. Sebagaimana diuraikan dalam model ini, AWS bertanggung jawab untuk memberikan perlindungan terhadap infrastruktur global yang menjalankan semua AWS Cloud. Anda harus bertanggung jawab untuk memelihara kendali terhadap konten yang di-hosting pada infrastruktur ini. Anda juga bertanggung jawab atas konfigurasi keamanan dan tugas manajemen untuk berbagai layanan Layanan AWS yang Anda gunakan. Untuk informasi selengkapnya tentang privasi data, silakan lihat Pertanyaan Umum Privasi Data. Untuk informasi tentang perlindungan data di Eropa, silakan lihat postingan blog Model Tanggung Jawab Bersama AWS dan GDPR di Blog Keamanan AWS.

Untuk tujuan perlindungan data, sebaiknya Anda melindungi kredensial Akun AWS dan menyiapkan akun pengguna individu dengan AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara seperti itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugas mereka. Kami juga merekomendasikan agar Anda mengamankan data Anda dengan cara-cara berikut ini:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk melakukan komunikasi dengan sumber daya AWS. Kami mensyaratkan TLS 1.2 dan merekomendasikan TLS 1.3.
- Siapkan API dan log aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi enkripsi AWS, bersama dengan semua kontrol keamanan default dalam layanan Layanan AWS.
- Gunakan layanan keamanan terkelola lanjutan seperti Amazon Macie, yang membantu menemukan dan mengamankan data pribadi yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-2 ketika mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Untuk informasi selengkapnya tentang titik akhir FIPS yang tersedia, silakan lihat Standar Pemrosesan Informasi Federal (FIPS) 140-2.

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tag atau bidang isian bebas seperti bidang Nama. Ini termasuk ketika Anda menggunakan CloudWatch atau Layanan AWS lainnya dengan menggunakan konsol, API, AWS CLI, atau SDK AWS. Data apa pun yang Anda masukkan ke dalam tag atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, sebaiknya Anda

tidak menyertakan informasi kredensial di URL untuk memvalidasi permintaan Anda ke server tersebut.

Enkripsi dalam transit

CloudWatch menggunakan enkripsi end-to-end data bergerak.

Manajemen identitas dan akses untuk Amazon CloudWatch

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diberi wewenang (memiliki izin) untuk menggunakan sumber daya. CloudWatch IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

- CloudWatch Referensi izin Amazon

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan. CloudWatch

Pengguna layanan — Jika Anda menggunakan CloudWatch layanan untuk melakukan pekerjaan Anda, administrator Anda memberi Anda kredensi dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak CloudWatch fitur untuk melakukan pekerjaan Anda, Anda mungkin memerlukan izin tambahan. Memahami cara akses dikelola dapat membantu Anda meminta izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur di CloudWatch, lihat Memecahkan masalah CloudWatch identitas dan akses Amazon.

Administrator layanan — Jika Anda bertanggung jawab atas CloudWatch sumber daya di perusahaan Anda, Anda mungkin memiliki akses penuh ke CloudWatch. Tugas Anda adalah menentukan CloudWatch fitur dan sumber daya mana yang harus diakses pengguna layanan Anda. Kemudian, Anda harus mengirimkan permintaan kepada administrator IAM Anda untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep Basic IAM. Untuk mempelajari lebih lanjut tentang bagaimana perusahaan Anda dapat menggunakan IAM CloudWatch, lihatBagaimana Amazon CloudWatch 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 CloudWatch. Untuk melihat contoh kebijakan CloudWatch berbasis identitas yang dapat Anda gunakan di IAM, lihat. Contoh kebijakan berbasis identitas untuk Amazon CloudWatch

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 kredensil 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 gabungan, 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 kredensil Anda. Jika Anda tidak menggunakan AWS alat, Anda harus menandatangani permintaan sendiri. Untuk informasi selengkapnya tentang penggunaan metode yang disarankan untuk menandatangani permintaan sendiri, lihat Menandatangani permintaan AWS API di 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 lebih lanjut, lihat Autentikasi multi-faktor dalam Panduan Pengguna AWS IAM Identity Center dan Menggunakan autentikasi multi-faktor (MFA) di AWS 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 Anda. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar tugas lengkap yang mengharuskan Anda masuk sebagai pengguna root, lihat Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

Identitas terfederasi

Sebagai praktik terbaik, mewajibkan pengguna manusia, termasuk pengguna yang memerlukan akses administrator, untuk menggunakan federasi dengan penyedia identitas untuk mengakses Layanan AWS dengan menggunakan kredensi sementara.

Identitas federasi adalah pengguna dari direktori pengguna perusahaan Anda, penyedia identitas web, direktori Pusat Identitas AWS Directory Service, atau pengguna mana pun yang mengakses Layanan AWS dengan menggunakan kredensil yang disediakan melalui sumber identitas. Ketika identitas federasi mengakses Akun AWS, mereka mengambil peran, dan peran memberikan kredensi sementara.

Untuk pengelolaan akses terpusat, sebaiknya Anda menggunakan AWS IAM Identity Center. Anda dapat membuat pengguna dan grup di Pusat Identitas IAM, atau Anda dapat menghubungkan dan menyinkronkan ke sekumpulan pengguna dan grup di sumber identitas Anda sendiri untuk digunakan di semua aplikasi Akun AWS dan aplikasi Anda. Untuk informasi tentang Pusat Identitas IAM, lihat Apa yang dimaksud Pusat Identitas IAM? dalam Panduan Pengguna AWS IAM Identity Center .

Pengguna dan grup IAM

Pengguna IAM adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, sebaiknya andalkan kredensial temporer, dan bukan membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan khusus yang memerlukan kredensial jangka panjang dengan pengguna IAM, sebaiknya rotasikan 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 kumpulan pengguna IAM. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin untuk beberapa pengguna sekaligus. Grup membuat izin lebih mudah dikelola untuk sekelompok besar pengguna. Misalnya, Anda dapat memiliki grup yang bernama 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 tersebut dimaksudkan untuk dapat diambil oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari selengkapnya, silakan lihat Kapan harus membuat pengguna IAM (bukan peran) 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. Anda dapat mengambil peran IAM untuk sementara AWS Management Console dengan beralih peran. Anda dapat mengambil peran dengan memanggil operasi AWS CLI atau AWS API atau dengan menggunakan URL kustom. Untuk informasi selengkapnya tentang metode untuk menggunakan peran, lihat Menggunakan peran IAM dalam Panduan Pengguna IAM.

Peran IAM dengan kredensial sementara berguna dalam situasi berikut:

- Akses pengguna gabungan – Untuk menetapkan izin ke sebuah identitas gabungan, Anda dapat membuat peran dan menentukan izin untuk peran tersebut. Saat identitas terfederasi diautentikasi, identitas tersebut dikaitkan dengan peran dan diberikan izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat Membuat peran untuk Penyedia Identitas pihak ketiga dalam Panduan Pengguna IAM. Jika Anda menggunakan Pusat Identitas IAM, Anda mengonfigurasi sekumpulan izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM mengaitkan izin yang ditetapkan ke peran dalam IAM. Untuk informasi tentang rangkaian izin, lihat Rangkaian 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 (pengguna utama tepercaya) dengan akun berbeda untuk mengakses sumber daya yang ada 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 kebijakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Bagaimana peran IAM berbeda dari kebijakan berbasis sumber daya dalam Panduan Pengguna IAM.
- Akses lintas layanan — Beberapa Layanan AWS menggunakan fitur lain Layanan AWS. Contoh, ketika Anda melakukan panggilan dalam layanan, umumnya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Suatu layanan mungkin melakukan hal tersebut menggunakan izin pengguna utama panggilan, 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 tindakan yang kemudian memulai 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 Meneruskan sesi akses.
  - Peran IAM – Peran layanan adalah peran IAM yang diambil layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, memodifikasi, dan menghapus

peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

- Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke. Layanan AWS Layanan tersebut dapat mengambil peran untuk melakukan sebuah 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 instans EC2 dan membuat atau permintaan API. AWS CLI AWS Cara ini lebih dianjurkan daripada menyimpan kunci akses dalam instans EC2. Untuk menetapkan AWS peran ke instans EC2 dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instance yang dilampirkan ke instance. Profil instans berisi peran dan memungkinkan program yang berjalan di instans EC2 mendapatkan kredensial sementara. Untuk informasi selengkapnya, lihat Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan di instans Amazon EC2 dalam Panduan Pengguna IAM.

Untuk mempelajari apakah kita harus menggunakan peran IAM atau pengguna IAM, lihat Kapan harus membuat peran IAM (bukan pengguna) dalam 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 Ikhtisar kebijakan JSON dalam Panduan Pengguna IAM.

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

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

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk operasi. Sebagai contoh, anggap saja 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 pengguna dan peran, di sumber daya mana, dan dengan ketentuan apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan terkelola. 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 terkelola atau kebijakan inline, lihat Memilih antara kebijakan terkelola 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 yang dilampiri kebijakan tersebut, kebijakan ini menentukan jenis tindakan yang dapat dilakukan oleh pengguna utama tertentu di sumber daya tersebut dan apa ketentuannya. Anda harus menentukan pengguna utama 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 (ACL)

Daftar kontrol akses (ACL) mengendalikan pengguna utama mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL sama dengan kebijakan berbasis sumber daya, meskipun tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung ACL. Untuk mempelajari ACL selengkapnya, silakan lihat Gambaran umum daftar kontrol akses (ACL) di Panduan Developer Layanan Penyimpanan Ringkas Amazon.

Tipe kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Tipe-tipe kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda berdasarkan tipe kebijakan yang lebih umum.

- Batasan izin – Batasan izin adalah fitur lanjutan di mana Anda menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM (pengguna 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 secara eksplisit terhadap salah satu kebijakan ini akan mengesampingkan izin tersebut. Untuk informasi selengkapnya tentang batasan izin, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCP) — SCP 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 organisasi, Anda dapat menerapkan kebijakan kontrol layanan (SCP) ke sebagian atau semua akun Anda. SCP membatasi izin untuk entitas di akun anggota, termasuk masing-masing. Pengguna root akun AWS Untuk informasi selengkapnya tentang Organisasi dan SCP, lihat Cara kerja SCP dalam Panduan Pengguna AWS Organizations .
- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda teruskan sebagai parameter saat Anda membuat sesi sementara secara terprogram untuk peran atau pengguna gabungan. 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 di salah satu kebijakan ini akan membatalkan izin tersebut. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Jika beberapa jenis kebijakan diberlakukan untuk satu 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 Amazon CloudWatch bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses CloudWatch, pelajari fitur IAM yang tersedia untuk digunakan. CloudWatch

Fitur IAM yang dapat Anda gunakan dengan Amazon CloudWatch

Fitur IAM	CloudWatch dukungan
Kebijakan berbasis identitas	Ya
Kebijakan berbasis sumber daya	Tidak
Tindakan kebijakan	Ya
Sumber daya kebijakan	Ya
kunci-kunci persyaratan kebijakan (spesifik layanan)	Ya
ACL	Tidak
ABAC (tanda dalam kebijakan)	Parsial
Kredensial sementara	Ya
Izin pengguna utama	Ya
Peran layanan	Ya
Peran terkait layanan	Tidak

Untuk mendapatkan tampilan tingkat tinggi tentang cara CloudWatch dan AWS layanan lain bekerja dengan sebagian besar fitur IAM, lihat AWS layanan yang bekerja dengan IAM di Panduan Pengguna IAM.

Kebijakan berbasis identitas untuk CloudWatch

Mendukung kebijakan berbasis identitas	Ya

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 pengguna dan peran, di sumber daya mana, dan dengan ketentuan apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan tindakan dan sumber daya yang diizinkan atau ditolak, serta ketentuan terkait jenis tindakan yang diizinkan atau ditolak. Anda tidak dapat menentukan pengguna utama dalam kebijakan berbasis identitas karena kebijakan ini berlaku untuk pengguna atau peran yang dilampiri kebijakan. Untuk mempelajari semua elemen yang dapat digunakan dalam kebijakan JSON, lihat Referensi elemen kebijakan JSON IAM dalam Panduan Pengguna IAM.

Contoh kebijakan berbasis identitas untuk CloudWatch

Untuk melihat contoh kebijakan CloudWatch berbasis identitas, lihat. Contoh kebijakan berbasis identitas untuk Amazon CloudWatch

Kebijakan berbasis sumber daya dalam CloudWatch

Mendukung kebijakan berbasis sumber daya	Tidak

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 yang dilampiri kebijakan tersebut, kebijakan ini menentukan jenis tindakan yang dapat dilakukan oleh pengguna utama tertentu di sumber daya tersebut dan apa ketentuannya. Anda harus menentukan pengguna utama dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Untuk mengaktifkan akses lintas akun, Anda dapat menentukan seluruh akun atau entitas IAM di akun lain sebagai pengguna utama dalam kebijakan berbasis sumber daya. Menambahkan pengguna utama lintas akun ke kebijakan berbasis sumber daya bagian dari membangun hubungan kepercayaan. Ketika prinsipal dan sumber daya berbeda Akun AWS, administrator IAM di akun tepercaya juga harus memberikan izin entitas utama (pengguna atau peran) untuk mengakses sumber daya. Izin diberikan dengan melampirkan kebijakan berbasis identitas ke entitas tersebut.

Namun, jika kebijakan berbasis sumber daya memberikan akses kepada pengguna utama dalam akun yang sama, kebijakan berbasis identitas lainnya tidak diperlukan. Untuk informasi selengkapnya, lihat Perbedaan peran IAM dengan kebijakan berbasis sumber daya di Panduan Pengguna IAM.

Tindakan kebijakan untuk CloudWatch

Mendukung tindakan kebijakan	Ya

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

Menyertakan tindakan dalam suatu kebijakan untuk memberikan izin melakukan operasi terkait.

Untuk melihat daftar CloudWatch tindakan, lihat Tindakan yang ditentukan oleh Amazon CloudWatch di Referensi Otorisasi Layanan.

Tindakan kebijakan CloudWatch menggunakan awalan berikut sebelum tindakan:

```
cloudwatch
```

Untuk menetapkan secara spesifik beberapa tindakan dalam satu pernyataan, pisahkan tindakan-tindakan tersebut dengan koma.

```
"Action": [
      "cloudwatch:action1",
      "cloudwatch:action2"
         ]
```

Untuk melihat contoh kebijakan CloudWatch berbasis identitas, lihat. Contoh kebijakan berbasis identitas untuk Amazon CloudWatch

Sumber daya kebijakan untuk CloudWatch

Mendukung sumber daya kebijakan	Ya

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

Elemen kebijakan JSON `Resource` menentukan objek atau beberapa 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 di tingkat sumber daya, misalnya operasi pencantuman, gunakan wildcard (*) untuk mengindikasikan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

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

Untuk melihat daftar jenis CloudWatch sumber daya dan ARNnya, lihat Sumber daya yang ditentukan oleh Amazon CloudWatch di Referensi Otorisasi Layanan. Untuk mempelajari tindakan mana yang dapat Anda tentukan ARN dari setiap sumber daya, lihat Tindakan yang ditentukan oleh Amazon. CloudWatch

Untuk melihat contoh kebijakan CloudWatch berbasis identitas, lihat. Contoh kebijakan berbasis identitas untuk Amazon CloudWatch

Kunci kondisi kebijakan untuk CloudWatch

Mendukung kunci kondisi kebijakan spesifik layanan	Ya

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

Elemen `Condition` (atau blok `Condition`) memungkinkan Anda menentukan kondisi di mana suatu pernyataan akan diterapkan. Elemen `Condition` bersifat opsional. Anda dapat membuat ekspresi kondisional yang menggunakan operator kondisi, misalnya sama dengan atau kurang dari, untuk mencocokkan kondisi dalam kebijakan dengan nilai-nilai yang diminta.

Jika Anda menentukan beberapa elemen `Condition` dalam satu pernyataan, atau beberapa kunci dalam satu elemen `Condition`, AWS akan mengevaluasinya dengan menggunakan operasi `AND` logis. Jika Anda menentukan beberapa nilai untuk satu kunci kondisi, AWS mengevaluasi kondisi menggunakan `OR` operasi logis. Semua kondisi harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan kondisi. Sebagai contoh, Anda dapat memberikan izin kepada pengguna IAM untuk mengakses sumber daya hanya jika izin tersebut mempunyai tanda yang sesuai dengan nama pengguna IAM mereka. Untuk informasi selengkapnya, silakan lihat Elemen kebijakan IAM: variabel dan tanda di Panduan Pengguna IAM.

AWS mendukung kunci kondisi global dan kunci kondisi khusus layanan. Untuk melihat semua kunci kondisi AWS global, lihat kunci konteks kondisi AWS global di Panduan Pengguna IAM.

Untuk melihat daftar kunci CloudWatch kondisi, lihat Kunci kondisi untuk Amazon CloudWatch di Referensi Otorisasi Layanan. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan kunci kondisi, lihat Tindakan yang ditentukan oleh Amazon CloudWatch.

Untuk melihat contoh kebijakan CloudWatch berbasis identitas, lihat. Contoh kebijakan berbasis identitas untuk Amazon CloudWatch

ACL di CloudWatch

Mendukung ACL	Tidak

Daftar kontrol akses (ACL) mengontrol pengguna utama (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL sama dengan kebijakan berbasis sumber daya, meskipun tidak menggunakan format dokumen kebijakan JSON.

ABAC dengan CloudWatch

Mendukung ABAC (tanda dalam kebijakan)	Parsial

Kontrol akses berbasis atribut (ABAC) adalah strategi otorisasi yang menentukan izin berdasarkan atribut. Dalam AWS, atribut ini disebut tag. Anda dapat melampirkan tag ke entitas IAM (pengguna atau peran) dan ke banyak AWS sumber daya. Pemberian tanda ke entitas dan sumber daya adalah langkah pertama dari ABAC. Kemudian rancanglah kebijakan ABAC untuk mengizinkan operasi-operasi ketika tanda milik pengguna utama cocok dengan tanda yang ada di sumber daya yang ingin diakses.

ABAC sangat berguna di lingkungan yang berkembang dengan cepat dan berguna dalam situasi di mana pengelolaan kebijakan menjadi rumit.

Untuk mengendalikan akses berdasarkan tag, berikan informasi tentang tanda di elemen syarat dari sebuh kebijakan dengan menggunakan kunci-kunci persyaratan `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`.

Jika sebuah layanan mendukung ketiga kunci kondisi untuk setiap jenis sumber daya, nilainya adalah Ya untuk layanan tersebut. Jika suatu layanan mendukung ketiga kunci kondisi hanya untuk beberapa jenis sumber daya, nilainya adalah Parsial.

Untuk informasi selengkapnya tentang ABAC, lihat Apa itu ABAC? di Panduan Pengguna IAM. Untuk melihat tutorial terkait langkah-langkah penyiapan ABAC, lihat Menggunakan kontrol akses berbasis atribut (ABAC) di Panduan Pengguna IAM.

Menggunakan kredensi sementara dengan CloudWatch

Mendukung kredensial sementara	Ya

Beberapa Layanan AWS tidak berfungsi saat Anda masuk menggunakan kredensi sementara. Untuk informasi tambahan, termasuk yang Layanan AWS bekerja dengan kredensi sementara, lihat Layanan AWS yang bekerja dengan IAM di Panduan Pengguna IAM.

Anda menggunakan kredensi sementara jika Anda masuk AWS Management Console menggunakan metode apa pun kecuali nama pengguna dan kata sandi. Misalnya, ketika Anda mengakses AWS menggunakan tautan masuk tunggal (SSO) perusahaan Anda, proses tersebut secara otomatis

membuat kredensi sementara. Anda juga akan membuat kredensial sementara secara otomatis saat masuk ke konsol sebagai pengguna dan kemudian beralih peran. Untuk informasi selengkapnya tentang cara beralih peran, lihat Beralih peran (konsol) di Panduan Pengguna IAM.

Anda dapat membuat kredensyal sementara secara manual menggunakan API AWS CLI atau AWS . Anda kemudian dapat menggunakan kredensi sementara tersebut untuk mengakses. AWS AWS merekomendasikan agar Anda secara dinamis menghasilkan kredensi sementara alih-alih menggunakan kunci akses jangka panjang. Untuk informasi selengkapnya, lihat Kredensial keamanan sementara di IAM.

Izin utama lintas layanan untuk CloudWatch

Mendukung sesi akses maju (FAS)	Ya

Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan tindakan yang kemudian memulai 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 Meneruskan sesi akses.

Peran layanan untuk CloudWatch

Mendukung peran layanan	Ya

Peran layanan adalah peran IAM yang diambil oleh layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

> **Warning**
>
> Mengubah izin untuk peran layanan dapat merusak CloudWatch fungsionalitas. Edit peran layanan hanya jika CloudWatch memberikan panduan untuk melakukannya.

Contoh kebijakan berbasis identitas untuk Amazon CloudWatch

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi CloudWatch sumber daya. Mereka juga tidak dapat melakukan tugas dengan menggunakan AWS Management Console, AWS Command Line Interface (AWS CLI), atau AWS API. Untuk memberikan izin kepada pengguna untuk melakukan tindakan pada sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat menjalankan peran.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM menggunakan contoh dokumen kebijakan JSON ini, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Untuk detail tentang tindakan dan jenis sumber daya yang ditentukan oleh CloudWatch, termasuk format ARN untuk setiap jenis sumber daya, lihat Kunci tindakan, sumber daya, dan kondisi untuk Amazon CloudWatch di Referensi Otorisasi Layanan.

Topik

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus CloudWatch sumber daya di akun Anda. Tindakan ini dikenai biaya untuk Akun AWS Anda. 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 pekerjaan di 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 melakukan ini dengan menentukan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi

tertentu, juga dikenal sebagai izin hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan IAM untuk menerapkan izin, lihat Kebijakan dan izin di IAM di Panduan Pengguna IAM.

- Gunakan kondisi dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Misalnya, Anda dapat menulis syarat kebijakan untuk menentukan bahwa semua pengajuan 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: Syarat di Panduan Pengguna IAM.
- Menggunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda guna memastikan izin yang aman dan berfungsi – 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 Analizer Akses IAM di 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 mewajibkan MFA saat operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda. Untuk informasi selengkapnya, lihat Mengonfigurasi akses API yang dilindungi MFA di Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat Praktik terbaik keamanan di IAM di Panduan Pengguna IAM.

Menggunakan konsol CloudWatch

Untuk mengakses CloudWatch konsol Amazon, Anda harus memiliki set izin minimum. Izin ini harus memungkinkan Anda untuk membuat daftar dan melihat detail tentang CloudWatch sumber daya di Anda 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.

Anda tidak perlu mengizinkan izin konsol minimum untuk pengguna yang melakukan panggilan hanya ke AWS CLI atau AWS API. Sebaliknya, izinkan akses hanya ke tindakan yang cocok dengan operasi API yang coba dilakukan.

Untuk memastikan bahwa pengguna dan peran masih dapat menggunakan CloudWatch konsol, lampirkan juga kebijakan CloudWatch `ConsoleAccess` atau `ReadOnly` AWS terkelola ke entitas. Untuk informasi selengkapnya, lihat Menambahkan izin ke pengguna di Panduan Pengguna IAM.

Izin diperlukan untuk konsol CloudWatch

Set lengkap izin yang diperlukan untuk bekerja dengan CloudWatch konsol tercantum di bawah ini. Izin ini menyediakan akses tulis dan baca lengkap ke CloudWatch konsol.

- aplikasi-autoscaling: DescribeScalingPolicies
- penskalaan otomatis: DescribeAutoScalingGroups
- penskalaan otomatis: DescribePolicies
- cloudtrail: DescribeTrails
- jam tangan awan: DeleteAlarms
- jam tangan awan: DescribeAlarmHistory
- jam tangan awan: DescribeAlarms
- jam tangan awan: GetMetricData
- jam tangan awan: GetMetricStatistics
- jam tangan awan: ListMetrics
- jam tangan awan: PutMetricAlarm
- jam tangan awan: PutMetricData
- EC2: DescribeInstances
- EC2: DescribeTags
- EC2: DescribeVolumes
- es: DescribeElasticsearchDomain
- es: ListDomainNames
- peristiwa: DeleteRule
- peristiwa: DescribeRule
- peristiwa: DisableRule
- peristiwa: EnableRule
- peristiwa: ListRules

- peristiwa: PutRule
- saya: AttachRolePolicy
- saya: CreateRole
- saya: GetPolicy
- saya: GetPolicyVersion
- saya: GetRole
- saya: ListAttachedRolePolicies
- saya: ListRoles
- kinesis: DescribeStream
- kinesis: ListStreams
- lambda: AddPermission
- lambda: CreateFunction
- lambda: GetFunctionConfiguration
- lambda: ListAliases
- lambda: ListFunctions
- lambda: ListVersionsByFunction
- lambda: RemovePermission
- log: CancelExportTask
- log: CreateExportTask
- log: CreateLogGroup
- log: CreateLogStream
- log: DeleteLogGroup
- log: DeleteLogStream
- log: DeleteMetricFilter
- log: DeleteRetentionPolicy
- log: DeleteSubscriptionFilter
- log: DescribeExportTasks
- log: DescribeLogGroups

- log: DescribeLogStreams
- log: DescribeMetricFilters
- log: DescribeQueries
- log: DescribeSubscriptionFilters
- log: FilterLogEvents
- log: GetLogGroupFields
- log: GetLogRecord
- log: GetLogEvents
- log: GetQueryResults
- log: PutMetricFilter
- log: PutRetentionPolicy
- log: PutSubscriptionFilter
- log: StartQuery
- log: StopQuery
- log: TestMetricFilter
- s3: CreateBucket
- s3: ListBucket
- SNS: CreateTopic
- SNS: GetTopicAttributes
- SNS: ListSubscriptions
- SNS: ListTopics
- SNS: SetTopicAttributes
- sns:Subscribe
- sns:Unsubscribe
- persegi: GetQueueAttributes
- persegi: GetQueueUrl
- persegi: ListQueues
- persegi: SetQueueAttributes
- swf: CreateAction

- swf: DescribeAction
- swf: ListActionTemplates
- swf: RegisterAction
- swf: RegisterDomain
- swf: UpdateAction

Selain itu, untuk melihat X-Ray Trace Map, Anda memerlukan `AWSXrayReadOnlyAccess`

Memecahkan masalah CloudWatch identitas dan akses Amazon

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan CloudWatch dan IAM.

Topik

- Saya tidak berwenang untuk melakukan tindakan di CloudWatch
- Saya tidak berwenang untuk melakukan iam: PassRole
- Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses CloudWatch sumber daya saya

Saya tidak berwenang untuk melakukan tindakan di CloudWatch

Jika Anda menerima pesan kesalahan bahwa Anda tidak memiliki otorisasi untuk melakukan tindakan, kebijakan Anda harus diperbarui agar Anda dapat melakukan tindakan tersebut.

Contoh kesalahan berikut terjadi ketika pengguna IAM `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya *`my-example-widget`* rekaan, tetapi tidak memiliki izin `cloudwatch:`*`GetWidget`* rekaan.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 cloudwatch:GetWidget on resource: my-example-widget
```

Dalam hal ini, kebijakan untuk pengguna `mateojackson` harus diperbarui untuk mengizinkan akses ke sumber daya *`my-example-widget`* dengan menggunakan tindakan `cloudwatch:`*`GetWidget`*.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan yang tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran CloudWatch.

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 ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol tersebut untuk melakukan tindakan di CloudWatch. 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 ingin mengizinkan orang di luar saya Akun AWS untuk mengakses CloudWatch sumber daya saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau pengguna 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 (ACL), Anda dapat menggunakan kebijakan tersebut untuk memberi pengguna akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa hal berikut:

- Untuk mempelajari apakah CloudWatch mendukung fitur-fitur ini, lihatBagaimana Amazon CloudWatch 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 Memberikan akses kepada pengguna eksternal yang sah (federasi identitas) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara penggunaan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Perbedaan antara peran IAM dan kebijakan berbasis sumber daya di Panduan Pengguna IAM.

CloudWatch pembaruan izin dasbor

Pada tanggal 1 Mei 2018, AWS mengubah izin yang diperlukan untuk mengakses CloudWatch dasbor. Akses dasbor di CloudWatch konsol kini memerlukan izin yang diperkenalkan pada tahun 2017 untuk mendukung operasi API dasbor:

- jam tangan awan: GetDashboard
- jam tangan awan: ListDashboards
- jam tangan awan: PutDashboard
- jam tangan awan: DeleteDashboards

Untuk mengakses CloudWatch dasbor, Anda memerlukan salah satu dari yang berikut:

- AdministratorAccess Kebijakan.
- CloudWatchFullAccess Kebijakan.
- Kebijakan kustom yang mencakup satu atau beberapa izin spesifik tersebut:
  - `cloudwatch:GetDashboard` dan `cloudwatch:ListDashboards` untuk dapat menampilkan dasbor
  - `cloudwatch:PutDashboard` untuk dapat membuat atau melakukan modifikasi pada dasbor
  - `cloudwatch:DeleteDashboards` untuk dapat menghapus dasbor

Untuk informasi selengkapnya tentang penggunaan kebijakan untuk mengubah izin bagi pengguna IAM, silakan lihat Mengubah Izin untuk pengguna IAM.

Untuk informasi selengkapnya tentang CloudWatch izin, lihat CloudWatch Referensi izin Amazon.

Untuk informasi selengkapnya tentang operasi API dasbor, lihat PutDashboarddi Referensi CloudWatch API Amazon.

AWS kebijakan terkelola (standar) untuk CloudWatch

AWS mengatasi banyak kasus penggunaan umum dengan menyediakan kebijakan IAM mandiri yang dibuat dan dikelola oleh. AWS Kebijakan AWS terkelola ini memberikan izin yang diperlukan untuk kasus penggunaan umum sehingga Anda dapat menghindari keharusan menyelidiki izin apa yang diperlukan. Untuk informasi selengkapnya, lihat Kebijakan yang dikelola AWS dalam Panduan Pengguna IAM.

Kebijakan AWS terkelola berikut, yang dapat Anda lampirkan ke pengguna di akun Anda, khusus untuk CloudWatch.

Topik

- CloudWatchFullAccessV2
- CloudWatchFullAccess
- CloudWatchReadOnlyAccess
- CloudWatchActionsEC2Akses
- CloudWatchAutomaticDashboardsAccess
- CloudWatchAgentServerPolicy
- CloudWatchAgentAdminPolicy
- AWS kebijakan terkelola (standar) untuk observabilitas CloudWatch lintas akun
- AWS kebijakan terkelola (standar) untuk CloudWatch Synthetics
- AWS kebijakan terkelola (standar) untuk Amazon CloudWatch RUM
- AWS kebijakan terkelola (telah ditentukan) untuk CloudWatch Terbukti
- AWS kebijakan terkelola untuk Manajer Insiden AWS Systems Manager

CloudWatchFullAccessV2

AWS baru-baru ini menambahkan kebijakan IAM yang dikelola CloudWatchFullAccessV2. Kebijakan ini memberikan akses penuh ke CloudWatch tindakan dan sumber daya dan juga lebih tepat mencakup izin yang diberikan untuk layanan lain seperti Amazon SNS dan. Amazon EC2 Auto Scaling Kami menyarankan Anda mulai menggunakan kebijakan ini daripada menggunakan CloudWatchFullAccess. AWS berencana untuk mencela CloudWatchFullAccessdalam waktu dekat.

Ini mencakup beberapa `autoscaling:Describe` izin sehingga pengguna dengan kebijakan ini dapat melihat tindakan Auto Scaling yang terkait dengan CloudWatch alarm. Ini mencakup beberapa `sns` izin sehingga pengguna dengan kebijakan ini dapat mengambil topik buat Amazon SNS dan mengaitkannya dengan alarm. CloudWatch Ini mencakup izin IAM sehingga pengguna dengan kebijakan ini dapat melihat informasi tentang peran terkait layanan yang terkait dengannya. CloudWatch Ini mencakup `oam:ListSinks` dan `oam:ListAttachedLinks` izin sehingga pengguna dengan kebijakan ini dapat menggunakan konsol untuk melihat data yang dibagikan dari akun sumber dalam pengamatan CloudWatch lintas akun.

Ini termasuk`rum`,`synthetics`, dan `xray` izin sehingga pengguna dapat memiliki akses penuh ke CloudWatch Synthetics AWS X-Ray,, CloudWatch dan RUM, yang semuanya berada di bawah CloudWatch layanan.

Isi CloudWatchFullAccessV2 adalah sebagai berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CloudWatchFullAccessPermissions",
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:DescribeScalingPolicies",
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribePolicies",
                "cloudwatch:*",
                "logs:*",
                "sns:CreateTopic",
                "sns:ListSubscriptions",
                "sns:ListSubscriptionsByTopic",
                "sns:ListTopics",
                "sns:Subscribe",
                "iam:GetPolicy",
                "iam:GetPolicyVersion",
                "iam:GetRole",
                "oam:ListSinks",
                "rum:*",
                "synthetics:*",
                "xray:*"
            ],
            "Resource": "*"
        },
```

```
        {
            "Sid": "CloudWatchApplicationSignalsServiceLinkedRolePermissions",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/application-
signals.cloudwatch.amazonaws.com/AWSServiceRoleForCloudWatchApplicationSignals",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "application-
signals.cloudwatch.amazonaws.com"
                }
            }
        },
        {
            "Sid": "EventsServicePermissions",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/events.amazonaws.com/
AWSServiceRoleForCloudWatchEvents*",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "events.amazonaws.com"
                }
            }
        },
         {
            "Sid": "OAMReadPermissions",
            "Effect": "Allow",
            "Action": [
                "oam:ListAttachedLinks"
            ],
            "Resource": "arn:aws:oam:*:*:sink/*"
        }
    ]
}
```

CloudWatchFullAccess

CloudWatchFullAccessKebijakan ini berada di jalur menuju penghentian. Kami menyarankan Anda berhenti menggunakannya, dan gunakan CloudWatchFullAccessV2 sebagai gantinya.

Isi dari CloudWatchFullAccessadalah sebagai berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "autoscaling:Describe*",
                "cloudwatch:*",
                "logs:*",
                "sns:*",
                "iam:GetPolicy",
                "iam:GetPolicyVersion",
                "iam:GetRole",
                "oam:ListSinks"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/events.amazonaws.com/
AWSServiceRoleForCloudWatchEvents*",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "events.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:ListAttachedLinks"
            ],
            "Resource": "arn:aws:oam:*:*:sink/*"
        }
    ]
}
```

CloudWatchReadOnlyAccess

CloudWatchReadOnlyAccessKebijakan ini memberikan akses hanya-baca ke. CloudWatch

Kebijakan ini mencakup beberapa `logs:` izin, sehingga pengguna dengan kebijakan ini dapat menggunakan konsol untuk melihat informasi CloudWatch Log dan kueri Wawasan CloudWatch Log. Ini termasuk`autoscaling:Describe*`, sehingga pengguna dengan kebijakan ini dapat melihat tindakan Auto Scaling yang terkait dengan CloudWatch alarm. Kebijakan ini mencakup `application-autoscaling:DescribeScalingPolicies`, sehingga pengguna dengan kebijakan ini dapat mengakses informasi tentang kebijakan penskalaan otomatis Application Auto Scaling. Ini termasuk `sns:Get*` dan`sns:List*`, sehingga pengguna dengan kebijakan ini dapat mengambil informasi tentang topik Amazon SNS yang menerima pemberitahuan CloudWatch tentang alarm. Ini mencakup `oam:ListSinks` dan `oam:ListAttachedLinks` izin, sehingga pengguna dengan kebijakan ini dapat menggunakan konsol untuk melihat data yang dibagikan dari akun sumber dalam pengamatan CloudWatch lintas akun.

Ini mencakup`rum`,`synthetics`, dan `xray` izin sehingga pengguna dapat memiliki akses hanya-baca ke CloudWatch Synthetics,, dan CloudWatch RUM AWS X-Ray, yang semuanya berada di bawah layanan. CloudWatch

Berikut ini adalah isi dari CloudWatchReadOnlyAccesskebijakan tersebut.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CloudWatchReadOnlyAccessPermissions",
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:DescribeScalingPolicies",
                "autoscaling:Describe*",
                "cloudwatch:BatchGet*",
                "cloudwatch:Describe*",
                "cloudwatch:GenerateQuery",
                "cloudwatch:Get*",
                "cloudwatch:List*",
                "logs:Get*",
                "logs:List*",
                "logs:StartQuery",
                "logs:StopQuery",
                "logs:Describe*",
                "logs:TestMetricFilter",
                "logs:FilterLogEvents",
                "logs:StartLiveTail",
                "logs:StopLiveTail",
```

```
                "oam:ListSinks",
                "sns:Get*",
                "sns:List*",
                "rum:BatchGet*",
                "rum:Get*",
                "rum:List*",
                "synthetics:Describe*",
                "synthetics:Get*",
                "synthetics:List*",
                "xray:BatchGet*",
                "xray:Get*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "OAMReadPermissions",
            "Effect": "Allow",
            "Action": [
                "oam:ListAttachedLinks"
            ],
            "Resource": "arn:aws:oam:*:*:sink/*"
        }
    ]
}
```

CloudWatchActionsEC2Akses

Kebijakan CloudWatchActionsEC2Access memberikan akses hanya-baca ke CloudWatch alarm dan metrik selain metadata Amazon EC2. Kebijakan ini juga memberi akses untuk Menghentikan, Memutus, dan Reboot tindakan API pada instans EC2.

Berikut ini adalah isi dari kebijakan CloudWatchActionsEC2Access.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:Describe*",
        "ec2:Describe*",
        "ec2:RebootInstances",
        "ec2:StopInstances",
```

```
        "ec2:TerminateInstances"
      ],
      "Resource": "*"
    }
  ]
}
```

CloudWatchAutomaticDashboardsAccess

Kebijakan CloudWatch- CrossAccountAccess dikelola digunakan oleh peran CloudWatch-CrossAccountSharingRole IAM. Peran dan kebijakan ini memungkinkan pengguna dasbor lintas akun untuk melihat dasbor otomatis di setiap akun yang merupakan dasbor berbagi.

Berikut ini adalah isi dari CloudWatchAutomaticDashboardsAccess:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "autoscaling:DescribeAutoScalingGroups",
        "cloudfront:GetDistribution",
        "cloudfront:ListDistributions",
        "dynamodb:DescribeTable",
        "dynamodb:ListTables",
        "ec2:DescribeInstances",
        "ec2:DescribeVolumes",
        "ecs:DescribeClusters",
        "ecs:DescribeContainerInstances",
        "ecs:ListClusters",
        "ecs:ListContainerInstances",
        "ecs:ListServices",
        "elasticache:DescribeCacheClusters",
        "elasticbeanstalk:DescribeEnvironments",
        "elasticfilesystem:DescribeFileSystems",
        "elasticloadbalancing:DescribeLoadBalancers",
        "kinesis:DescribeStream",
        "kinesis:ListStreams",
        "lambda:GetFunction",
        "lambda:ListFunctions",
        "rds:DescribeDBClusters",
        "rds:DescribeDBInstances",
        "resource-groups:ListGroupResources",
```

```
        "resource-groups:ListGroups",
        "route53:GetHealthCheck",
        "route53:ListHealthChecks",
        "s3:ListAllMyBuckets",
        "s3:ListBucket",
        "sns:ListTopics",
        "sqs:GetQueueAttributes",
        "sqs:GetQueueUrl",
        "sqs:ListQueues",
        "synthetics:DescribeCanariesLastRun",
        "tag:GetResources"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": [
        "apigateway:GET"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:apigateway:*::/restapis*"
      ]
    }
  ]
```

CloudWatchAgentServerPolicy

CloudWatchAgentServerPolicyKebijakan ini dapat digunakan dalam peran IAM yang dilampirkan ke instans Amazon EC2 untuk memungkinkan CloudWatch agen membaca informasi dari instans dan menuliskannya. CloudWatch Isinya adalah sebagai berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CWACloudWatchServerPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData",
                "ec2:DescribeVolumes",
                "ec2:DescribeTags",
                "logs:PutLogEvents",
```

```
                "logs:PutRetentionPolicy",
                "logs:DescribeLogStreams",
                "logs:DescribeLogGroups",
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets",
                "xray:GetSamplingStatisticSummaries"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CWASSMServerPermissions",
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameter"
            ],
            "Resource": "arn:aws:ssm:*:*:parameter/AmazonCloudWatch-*"
        }
    ]
}
```

CloudWatchAgentAdminPolicy

CloudWatchAgentAdminPolicyKebijakan ini dapat digunakan dalam peran IAM yang dilampirkan ke instans Amazon EC2. Kebijakan ini memungkinkan CloudWatch agen untuk membaca informasi dari instance dan menuliskannya CloudWatch, dan juga untuk menulis informasi ke Parameter Store. Isinya adalah sebagai berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CWACloudWatchPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData",
                "ec2:DescribeTags",
                "logs:PutLogEvents",
                "logs:PutRetentionPolicy",
                "logs:DescribeLogStreams",
```

```
                "logs:DescribeLogGroups",
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets",
                "xray:GetSamplingStatisticSummaries"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CWASSMPermissions",
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameter",
                "ssm:PutParameter"
            ],
            "Resource": "arn:aws:ssm:*:*:parameter/AmazonCloudWatch-*"
        }
    ]
}
```

> **Note**
>
> Anda dapat meninjau kebijakan-kebijakan izin ini dengan masuk ke konsol IAM dan mencari kebijakan-kebijakan tertentu di sana.

Anda juga dapat membuat kebijakan IAM khusus Anda sendiri untuk memberikan izin untuk CloudWatch tindakan dan sumber daya. Anda dapat melampirkan kebijakan-kebijakan kustom ini ke pengguna IAM atau grup yang memerlukan izin-izin tersebut.

AWS kebijakan terkelola (standar) untuk observabilitas CloudWatch lintas akun

Kebijakan di bagian ini memberikan izin yang terkait dengan pengamatan CloudWatch lintas akun. Untuk informasi selengkapnya, lihat CloudWatch observabilitas lintas akun.

CloudWatchCrossAccountSharingConfiguration

CloudWatchCrossAccountSharingConfigurationKebijakan ini memberikan akses untuk membuat, mengelola, dan melihat tautan Pengelola Akses Observabilitas untuk berbagi CloudWatch sumber

daya antar akun. Untuk informasi selengkapnya, lihat CloudWatch observabilitas lintas akun. Isinya sebagai berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch: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/*"
            ]
        }
    ]
}
```

OAM FullAccess

FullAccessKebijakan OAM memberikan akses untuk membuat, mengelola, dan melihat sink dan tautan Manajer Akses Observabilitas, yang digunakan untuk pengamatan lintas akun. CloudWatch

FullAccessKebijakan OAM dengan sendirinya tidak mengizinkan Anda untuk berbagi data observabilitas di seluruh tautan. Untuk membuat tautan untuk berbagi CloudWatch metrik, Anda juga memerlukan salah satu CloudWatchFullAccessatau CloudWatchCrossAccountSharingConfiguration. Untuk membuat tautan untuk berbagi grup CloudWatch log Log, Anda juga memerlukan salah satu CloudWatchLogsFullAccessatau CloudWatchLogsCrossAccountSharingConfiguration. Untuk membuat tautan untuk berbagi jejak X-Ray, Anda juga memerlukan salah satu AWSXRayFullAccessatau AWSXRayCrossAccountSharingConfiguration.

Untuk informasi selengkapnya, lihat CloudWatch observabilitas lintas akun. Isinya sebagai berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "oam:*"
            ],
            "Resource": "*"
        }
    ]
}
```

OAM ReadOnlyAccess

ReadOnlyAccessKebijakan OAM memberikan akses hanya-baca ke sumber daya Manajer Akses Observabilitas, yang digunakan untuk observabilitas lintas akun. CloudWatch Untuk informasi selengkapnya, lihat CloudWatch observabilitas lintas akun. Isinya sebagai berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "oam:Get*",
                "oam:List*"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

AWS kebijakan terkelola (standar) untuk CloudWatch Synthetics

Kebijakan CloudWatchSyntheticsFullAccessdan CloudWatchSyntheticsReadOnlyAccess AWS terkelola tersedia untuk Anda tetapkan kepada pengguna yang akan mengelola atau menggunakan CloudWatch Synthetics. Kebijakan-kebijakan tambahan berikut juga relevan:

- AmazonS3 ReadOnlyAccess dan CloudWatchReadOnlyAccess— Ini diperlukan untuk dapat membaca semua data Synthetics di konsol. CloudWatch
- AWSLambdaReadOnlyAccess – Untuk dapat melihat kode sumber daya yang digunakan oleh kenari.
- CloudWatchSyntheticsFullAccessmemungkinkan Anda membuat kenari, Selain itu, untuk membuat dan menghapus kenari yang memiliki peran IAM baru yang dibuat untuknya, Anda juga memerlukan pernyataan kebijakan sebaris berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateRole",
                "iam:DeleteRole",
                "iam:CreatePolicy",
                "iam:DeletePolicy",
                "iam:AttachRolePolicy",
                "iam:DetachRolePolicy",
            ],
            "Resource": [
                "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*",
                "arn:aws:iam::*:policy/service-role/CloudWatchSyntheticsPolicy*"
            ]
        }
    ]
}
```

> **Important**
>
> Memberipenggunaiam:CreateRole,iam:DeleteRole, iam:CreatePolicy, iam:DeletePolicy, iam:AttachRolePolicy, dan iam:DetachRolePolicy izin memberi pengguna akses administratif penuh untuk membuat, melampirkan, dan menghapus peran dan kebijakan yang dimiliki ARN yang cocok arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole* dan arn:aws:iam::*:policy/service-role/CloudWatchSyntheticsPolicy*. Sebagai contoh, pengguna dengan izin ini dapat membuat kebijakan yang memiliki izin penuh untuk semua sumber daya, dan melampirkan kebijakan tersebut ke peran apa pun yang cocok dengan pola ARN tersebut. Berhati-hatilah dengan orang yang Anda berikan izin ini.

Untuk informasi tentang cara melampirkan kebijakan dan memberikan izin kepada pengguna, silakan lihat Mengubah Izin untuk Pengguna IAM dan Cara menyematkan kebijakan yang selaras bagi pengguna atau peran.

CloudWatchSyntheticsFullAccess

Berikut ini adalah isi dari CloudWatchSyntheticsFullAccesskebijakan tersebut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "synthetics:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:PutEncryptionConfiguration"
            ],
            "Resource": [
                "arn:aws:s3:::cw-syn-results-*"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:ListRoles",
                "s3:ListAllMyBuckets",
                "xray:GetTraceSummaries",
                "xray:BatchGetTraces",
                "apigateway:GET"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::cw-syn-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::aws-synthetics-library-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*"
            ],
            "Condition": {
```

```
                "StringEquals": {
                    "iam:PassedToService": [
                        "lambda.amazonaws.com",
                        "synthetics.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:ListAttachedRolePolicies"
            ],
            "Resource": [
                "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:GetMetricStatistics"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricAlarm",
                "cloudwatch:DeleteAlarms"
            ],
            "Resource": [
                "arn:aws:cloudwatch:*:*:alarm:Synthetics-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:DescribeAlarms"
            ],
            "Resource": [
                "arn:aws:cloudwatch:*:*:alarm:*"
            ]
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:CreateFunction",
                "lambda:AddPermission",
                "lambda:PublishVersion",
                "lambda:UpdateFunctionCode",
                "lambda:UpdateFunctionConfiguration",
                "lambda:GetFunctionConfiguration",
                "lambda:DeleteFunction"
            ],
            "Resource": [
                "arn:aws:lambda:*:*:function:cwsyn-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:GetLayerVersion",
                "lambda:PublishLayerVersion",
                "lambda:DeleteLayerVersion"
            ],
            "Resource": [
                "arn:aws:lambda:*:*:layer:cwsyn-*",
                "arn:aws:lambda:*:*:layer:Synthetics:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "sns:ListTopics"
            ],
```

```
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "sns:CreateTopic",
                "sns:Subscribe",
                "sns:ListSubscriptionsByTopic"
            ],
            "Resource": [
                "arn:*:sns:*:*:Synthetics-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:ListAliases"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:*:*:key/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:*:*:key/*",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": [
                        "s3.*.amazonaws.com"
                    ]
                }
            }
        }
    ]
```

```
}
```

CloudWatchSyntheticsReadOnlyAccess

Berikut ini adalah isi dari CloudWatchSyntheticsReadOnlyAccesskebijakan tersebut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "synthetics:Describe*",
                "synthetics:Get*",
                "synthetics:List*",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS kebijakan terkelola (standar) untuk Amazon CloudWatch RUM

Kebijakan AmazonCloudWatch ReadOnlyAccess AWS terkelola AmazonCloudWatchRUM FullAccess dan RUM tersedia untuk Anda tetapkan kepada pengguna yang akan mengelola atau menggunakan CloudWatch RUM.

AmazonCloudWatchRUM FullAccess

Berikut ini adalah isi dari FullAccess kebijakan AmazonCloudWatchRUM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rum:*"
            ],
            "Resource": "*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/rum.amazonaws.com/
AWSServiceRoleForRealUserMonitoring"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/RUM-Monitor*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "cognito-identity.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:DescribeAlarms"
            ],
            "Resource": "arn:aws:cloudwatch:*:*:alarm:*"
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "cognito-identity:CreateIdentityPool",
                "cognito-identity:ListIdentityPools",
                "cognito-identity:DescribeIdentityPool",
                "cognito-identity:GetIdentityPoolRoles",
                "cognito-identity:SetIdentityPoolRoles"
            ],
            "Resource": "arn:aws:cognito-identity:*:*:identitypool/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:DeleteLogGroup",
                "logs:PutRetentionPolicy",
                "logs:CreateLogStream"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:*RUMService*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries",
                "logs:DescribeResourcePolicies"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": "arn:aws:logs:*:*:log-group::log-stream:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "synthetics:describeCanaries",
                "synthetics:describeCanariesLastRun"
            ],
```

```
            "Resource": "arn:aws:synthetics:*:*:canary:*"
        }
    ]
}
```

AmazonCloudWatchRUM ReadOnlyAccess

Berikut ini adalah isi dari ReadOnlyAccess kebijakan AmazonCloudWatchRUM.

```
{

 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "rum:GetAppMonitor",
    "rum:GetAppMonitorData",
    "rum:ListAppMonitors",
    "rum:ListRumMetricsDestinations",
    "rum:BatchGetRumMetricDefinitions"
   ],
   "Resource": "*"
  }
 ]
}
```

AmazonCloudWatchRUM ServiceRolePolicy

Anda tidak dapat melampirkan AmazonCloudWatchRUM ServiceRolePolicy ke entitas IAM Anda. Kebijakan ini dilampirkan pada peran terkait layanan yang memungkinkan CloudWatch RUM mempublikasikan data pemantauan ke layanan terkait AWS lainnya. Untuk informasi selengkapnya tentang peran terkait layanan, silakan lihat Menggunakan peran terkait layanan untuk RUM CloudWatch .

Isi lengkap AmazonCloudWatchRUM ServiceRolePolicy adalah sebagai berikut.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
```

```
    "xray:PutTraceSegments"
   ],
   "Resource": [
    "*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": "cloudwatch:PutMetricData",
   "Resource": "*",
   "Condition": {
    "StringLike": {
     "cloudwatch:namespace": [
      "RUM/CustomMetrics/*",
      "AWS/RUM"
     ]
    }
   }
  }
 ]
}
```

AWS kebijakan terkelola (telah ditentukan) untuk CloudWatch Terbukti

Kebijakan CloudWatchEvidentlyFullAccessdan CloudWatchEvidentlyReadOnlyAccess AWS terkelola tersedia untuk Anda tetapkan kepada pengguna yang akan mengelola atau menggunakan CloudWatch Evidently.

CloudWatchEvidentlyFullAccess

Berikut ini adalah isi dari CloudWatchEvidentlyFullAccesskebijakan tersebut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "evidently:*"
            ],
            "Resource": "*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/service-role/CloudWatchRUMEvidentlyRole-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:DescribeAlarmHistory",
                "cloudwatch:DescribeAlarmsForMetric",
                "cloudwatch:ListTagsForResource"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:DescribeAlarms",
                "cloudwatch:TagResource",
                "cloudwatch:UnTagResource"
            ],
            "Resource": [
                "arn:aws:cloudwatch:*:*:alarm:*"
            ]
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudtrail:LookupEvents"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricAlarm"
            ],
            "Resource": [
                "arn:aws:cloudwatch:*:*:alarm:Evidently-Alarm-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "sns:ListTopics"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "sns:CreateTopic",
                "sns:Subscribe",
                "sns:ListSubscriptionsByTopic"
            ],
            "Resource": [
                "arn:*:sns:*:*:Evidently-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "*"
```

```
            ]
        }
    ]
}
```

CloudWatchEvidentlyReadOnlyAccess

Berikut ini adalah isi dari CloudWatchEvidentlyReadOnlyAccesskebijakan tersebut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "evidently:GetExperiment",
                "evidently:GetFeature",
                "evidently:GetLaunch",
                "evidently:GetProject",
                "evidently:GetSegment",
                "evidently:ListExperiments",
                "evidently:ListFeatures",
                "evidently:ListLaunches",
                "evidently:ListProjects",
                "evidently:ListSegments",
                "evidently:ListSegmentReferencs"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS kebijakan terkelola untuk Manajer Insiden AWS Systems Manager

AWSCloudWatchAlarms_ActionSSMIncidentsServiceRolePolicyKebijakan ini dilampirkan pada peran terkait layanan yang memungkinkan CloudWatch untuk memulai insiden di Manajer Insiden AWS Systems Manager atas nama Anda. Untuk informasi selengkapnya, lihat Izin peran terkait layanan untuk alarm tindakan CloudWatch Systems Manager Incident Manager.

Kebijakan ini memiliki izin-izin berikut:

- insiden ssm-: StartIncident

Contoh kebijakan yang dikelola pelanggan

Di bagian ini, Anda dapat menemukan contoh kebijakan pengguna yang memberikan izin untuk berbagai CloudWatch tindakan. Kebijakan ini berfungsi saat Anda menggunakan CloudWatch API, AWS SDK, atau. AWS CLI

Contoh-contoh

- Contoh 1: Izinkan pengguna akses penuh CloudWatch
- Contoh 2: Izinkan akses hanya-baca CloudWatch
- Contoh 3: Menghentikan atau Mengakhiri instans Amazon EC2

Contoh 1: Izinkan pengguna akses penuh CloudWatch

Untuk memberi pengguna akses penuh CloudWatch, Anda dapat menggunakan beri mereka kebijakan CloudWatchFullAccessterkelola alih-alih membuat kebijakan yang dikelola pelanggan. Isi dari CloudWatchFullAccesstercantum dalamCloudWatchFullAccess.

Contoh 2: Izinkan akses hanya-baca CloudWatch

Kebijakan berikut memungkinkan pengguna mengakses CloudWatch dan melihat CloudWatch tindakan Auto Scaling Amazon EC2, CloudWatch metrik, data Log, dan data Amazon SNS terkait alarm.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "autoscaling:Describe*",
        "cloudwatch:Describe*",
        "cloudwatch:Get*",
        "cloudwatch:List*",
        "logs:Get*",
        "logs:Describe*",
        "logs:StartQuery",
        "logs:StopQuery",
        "logs:TestMetricFilter",
        "logs:FilterLogEvents",
        "logs:StartLiveTail",
        "logs:StopLiveTail",
```

```
        "sns:Get*",
        "sns:List*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Contoh 3: Menghentikan atau Mengakhiri instans Amazon EC2

Kebijakan berikut memungkinkan tindakan CloudWatch alarm untuk menghentikan atau menghentikan instans EC2. Dalam contoh di bawah ini, GetMetricData ListMetrics, dan DescribeAlarms tindakan adalah opsional. Anda disarankan untuk menyertakan tindakan ini untuk memastikan bahwa Anda telah menghentikan atau mengakhiri instans tersebut dengan benar.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "cloudwatch:PutMetricAlarm",
        "cloudwatch:GetMetricData",
        "cloudwatch:ListMetrics",
        "cloudwatch:DescribeAlarms"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "ec2:DescribeInstanceStatus",
        "ec2:DescribeInstances",
        "ec2:StopInstances",
        "ec2:TerminateInstances"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
```

```
  ]
}
```

CloudWatch pembaruan kebijakan AWS terkelola

Lihat detail tentang pembaruan kebijakan AWS terkelola CloudWatch sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di halaman Riwayat CloudWatch dokumen.

Perubahan	Deskripsi	Tanggal
CloudWatchAgentServerPolicy — Permbaruan ke kebijakan yang sudah ada	CloudWatch menambahkan izin ke CloudWatchAgentServerPolicy. `logs:PutRetentionPolicy` Izin `xray:PutTraceSegments` `xray:PutTelemetryRecords` `xray:GetSamplingRules` ,,`xray:GetSamplingTargets` , `xray:GetSamplingStatisticSummaries` dan ditambahkan sehingga CloudWatch agen dapat mempublikasikan jejak X-Ray dan memodifikasi periode retensi grup log.	Februari 12, 2024
CloudWatchAgentAdminPolicy – Pembaruan pada kebijakan yang sudah ada	CloudWatch menambahkan izin ke CloudWatchAgentAdminPolicy.	Februari 12, 2024

Perubahan	Deskripsi	Tanggal
	`logs:PutRetentionPolicy` Izin `xray:PutTraceSegments` `xray:PutTelemetryRecords` `xray:GetSamplingRules` ,,`xray:GetSamplingTargets` , `xray:GetSamplingStatisticSummaries` dan ditambahkan sehingga CloudWatch agen dapat mempublikasikan jejak X-Ray dan memodifikasi periode retensi grup log.	

Perubahan	Deskripsi	Tanggal
CloudWatchFullAccessV2 - Perbarui ke kebijakan yang ada	CloudWatch menambahkan izin ke CloudWatchFullAccessV2. Izin yang ada untuk tindakan CloudWatch Synthetics, X-Ray, CloudWatch dan RUM serta izin baru CloudWatch untuk Sinyal Aplikasi ditambahkan sehingga pengguna dengan kebijakan ini dapat CloudWatch mengelola Sinyal Aplikasi. Izin untuk membuat peran terkait layanan Sinyal CloudWatch Aplikasi ditambahkan untuk memungkinkan Sinyal CloudWatch Aplikasi menemukan data telemetri dalam log, metrik, jejak, dan tag.	5 Desember 2023

Perubahan	Deskripsi	Tanggal
CloudWatchReadOnlyAccess – Pembaruan pada kebijakan yang sudah ada	CloudWatch menambahkan izin ke CloudWatchReadOnly Access. Izin hanya-baca yang ada untuk tindakan Synthetics CloudWatch , X-Ray, dan CloudWatch RUM serta izin hanya-baca baru untuk Sinyal CloudWatch Aplikasi ditambahkan sehingga pengguna dengan kebijakan ini dapat melakukan triase dan mencerna masalah kesehatan layanan mereka seperti yang dilaporkan oleh Sinyal Aplikasi. CloudWatch `cloudwatch:GenerateQuery` Izin ditambahkan agar pengguna dengan kebijakan ini dapat menghasilkan string kueri CloudWatch Metrics Insights dari prompt bahasa alami.	5 Desember 2023

Perubahan	Deskripsi	Tanggal
CloudWatchApplicationSignalsServiceRolePolicy – Kebijakan baru	CloudWatch menambahkan kebijakan baru CloudWatchApplicationSignalsServiceRolePolicy. Ini CloudWatchApplicationSignalsServiceRolePolicy memberikan izin fitur yang akan datang untuk mengumpulkan data CloudWatch Log, data jejak X-Ray, data CloudWatch metrik, dan data penandaan.	9 November 2023
AWSServiceRoleForCloudWatchMetrics_DbPerfInsightsServiceRolePolicy – Kebijakan baru	CloudWatch menambahkan kebijakan baru AWSServiceRoleForCloudWatchMetrics_DbPerfInsightsServiceRolePolicy. Ini AWSServiceRoleForCloudWatchMetrics_DbPerfInsightsServiceRolePolicymemberikan izin CloudWatch untuk mengambil metrik Performance Insights dari database atas nama Anda.	20 September 2023

Perubahan	Deskripsi	Tanggal
CloudWatchReadOnlyAccess – Pembaruan pada kebijakan yang sudah ada	CloudWatch menambahkan izin untuk CloudWatchReadOnlyAccess. Izin `application-autoscaling:DescribeScalingPolicies` ditambahkan sehingga pengguna yang memiliki kebijakan ini akan dapat mengakses informasi tentang kebijakan penskalaan otomatis Application Auto Scaling.	14 September 2023
CloudWatchFullAccessV2 - Kebijakan baru	CloudWatch menambahkan kebijakan baru CloudWatchFullAccessV2. CloudWatchFullAccessV2 memberikan akses penuh ke CloudWatch tindakan dan sumber daya sambil lebih baik melingkupi izin yang diberikan ke layanan lain seperti Amazon SNS dan. Amazon EC2 Auto Scaling Untuk informasi lebih lanjut, lihat CloudWatchFullAccessV2.	1 Agustus 2023

Perubahan	Deskripsi	Tanggal
AWSServiceRoleForInternetMonitor – Pembaruan pada kebijakan yang sudah ada	Amazon CloudWatch Internet Monitor menambahkan izin baru untuk memantau sumber daya Network Load Balancer. Izin `elasticloadbalancing:DescribeLoadBalancers` dan `ec2:DescribeNetworkInterfaces` diperlukan agar Internet Monitor dapat memantau lalu lintas Penyeimbang Beban Jaringan pelanggan dengan menganalisis log aliran pada sumber daya NLB. Untuk informasi selengkapnya, lihat Menggunakan Amazon CloudWatch Internet Monitor.	15 Juli 2023

Perubahan	Deskripsi	Tanggal
CloudWatchReadOnlyAccess – Pembaruan pada kebijakan yang sudah ada	CloudWatch menambahkan izin ke CloudWatchReadOnly Access. Izin `logs:StartLiveTail` dan `logs:StopLiveTail` izin ditambahkan sehingga pengguna dengan kebijakan ini dapat menggunakan konsol untuk memulai dan menghentikan sesi ekor langsung CloudWatch Log. Untuk informasi selengkap nya, silakan lihat Menggunak an live tail untuk melihat log mendekati waktu nyata.	6 Juni 2023
CloudWatchCrossAcc ountSharingConfiguration – Kebijakan baru	CloudWatch menambahk an kebijakan baru untuk memungkinkan Anda mengelola tautan observabi litas CloudWatch lintas akun yang berbagi CloudWatch metrik. Untuk informasi selengkapnya, lihat CloudWatch observabi litas lintas akun.	27 November 2022

Perubahan	Deskripsi	Tanggal
OAM FullAccess — Kebijakan baru	CloudWatch menambahkan kebijakan baru untuk memungkinkan Anda mengelola tautan dan sink observabilitas CloudWatch lintas akun sepenuhnya. Untuk informasi selengkapnya, lihat CloudWatch observabilitas lintas akun.	27 November 2022
OAM ReadOnlyAccess — Kebijakan baru	CloudWatch menambahkan kebijakan baru untuk memungkinkan Anda melihat informasi tentang tautan dan sink observabilitas CloudWatch lintas akun. Untuk informasi selengkapnya, lihat CloudWatch observabilitas lintas akun.	27 November 2022

Perubahan	Deskripsi	Tanggal
CloudWatchFullAccess – Pembaruan pada kebijakan yang sudah ada	CloudWatch menambahkan izin ke CloudWatchFullAccess. Izin `oam:ListSinks` dan `oam:ListAttachedLinks` izin ditambahkan sehingga pengguna dengan kebijakan ini dapat menggunakan konsol untuk melihat data yang dibagikan dari akun sumber dalam pengamatan CloudWatch lintas akun.	27 November 2022
CloudWatchReadOnlyAccess – Pembaruan pada kebijakan yang sudah ada	CloudWatch menambahkan izin ke CloudWatchReadOnlyAccess. Izin `oam:ListSinks` dan `oam:ListAttachedLinks` izin ditambahkan sehingga pengguna dengan kebijakan ini dapat menggunakan konsol untuk melihat data yang dibagikan dari akun sumber dalam pengamatan CloudWatch lintas akun.	27 November 2022

Perubahan	Deskripsi	Tanggal
AmazonCloudWatchRUMServiceRolePolicy - Perbarui ke kebijakan yang ada	CloudWatch RUM memperbarui kunci kondisi di AmazonCloudWatchRUM ServiceRolePolicy. Kunci `"Condition": { "StringEquals": { "cloudwatch:namespace": "AWS/RUM" } }` kondisi diubah menjadi berikut sehingga CloudWatch RUM dapat mengirim metrik khusus ke ruang nama metrik khusus. `"Condition": { "StringLike": { "cloudwatch:namespace": [ "RUM/CustomMetrics/*", "AWS/RUM" ] } }`	2 Februari 2023

Perubahan	Deskripsi	Tanggal
AmazonCloudWatchRUMReadOnlyAccess - Kebijakan yang diperbarui	CloudWatch menambahkan izin ReadOnlyAccess kebijakan AmazonCloudWatchRUM. `rum:BatchGetRumMetricsDefinitions` Izin `rum:ListRumMetricsDestinations` dan ditambahkan sehingga CloudWatch RUM dapat mengirim metrik yang diperluas ke CloudWatch dan Terbukti.	27 Oktober 2022
AmazonCloudWatchRUMServiceRolePolicy - Perbarui ke kebijakan yang ada	CloudWatch RUM menambahkan izin ke AmazonCloudWatchRUM ServiceRolePolicy. `cloudwatch:PutMetricData` Izin ditambahkan sehingga CloudWatch RUM dapat mengirim metrik yang diperluas ke CloudWatch.	26 Oktober 2022

Perubahan	Deskripsi	Tanggal
CloudWatchEvidentlyReadOnlyAccess – Pembaruan pada kebijakan yang sudah ada	CloudWatch Jelas menambahkan izin ke. CloudWatchEvidentlyReadOnlyAccess Izin `evidently:GetSegment` , `evidently:ListSegments` , dan `evidently:ListSegmentReferences` ditambahkan sehingga pengguna dengan kebijakan ini dapat melihat segmen audiens Evidently yang telah dibuat.	12 Agustus 2022
CloudWatchSyntheticsFullAccess – Pembaruan pada kebijakan yang sudah ada	CloudWatch Synthetics menambahkan izin ke. CloudWatchSyntheticsFullAccess `lambda:DeleteLayerVersion` Izin `lambda:DeleteFunction` dan ditambahkan sehingga CloudWatch Synthetics dapat menghapus sumber daya terkait saat kenari dihapus. `iam:ListAttachedRolePolicies` ditambahkan sehingga pelanggan dapat melihat kebijakan yang melekat pada peran IAM canary.	6 Mei 2022

Perubahan	Deskripsi	Tanggal
AmazonCloudWatchRUMFullAccess — Kebijakan baru	CloudWatch menambahkan kebijakan baru untuk mengaktifkan manajemen penuh CloudWatch RUM. CloudWatch RUM memungkinkan Anda untuk melakukan pemantauan pengguna nyata dari aplikasi web Anda. Untuk informasi selengkapnya, lihat Gunakan CloudWatch RUM.	29 November 2021
AmazonCloudWatchRUMReadOnlyAccess — Kebijakan baru	CloudWatch menambahkan kebijakan baru untuk mengaktifkan akses hanya-baca ke CloudWatch RUM. CloudWatch RUM memungkinkan Anda untuk melakukan pemantauan pengguna nyata dari aplikasi web Anda. Untuk informasi selengkapnya, lihat Gunakan CloudWatch RUM.	29 November 2021

Perubahan	Deskripsi	Tanggal
CloudWatchEvidentlyFullAccess – Kebijakan baru	CloudWatch menambahkan kebijakan baru untuk memungkinkan manajemen penuh CloudWatch Evidently. CloudWatch Jelas memungkinkan Anda untuk melakukan eksperimen A/B dari aplikasi web Anda, dan meluncurkannya secara bertahap. Untuk informasi selengkapnya, lihat Lakukan peluncuran dan eksperimen A/B dengan Evidently CloudWatch .	29 November 2021
CloudWatchEvidentlyReadOnlyAccess – Kebijakan baru	CloudWatch menambahkan kebijakan baru untuk mengaktifkan akses hanya-baca ke CloudWatch Evidently . CloudWatch Jelas memungkinkan Anda untuk melakukan eksperimen A/B dari aplikasi web Anda, dan meluncurkannya secara bertahap. Untuk informasi selengkapnya, lihat Lakukan peluncuran dan eksperimen A/B dengan Evidently CloudWatch .	29 November 2021

Perubahan	Deskripsi	Tanggal
AWSServiceRoleForCloudWatchRUM— Kebijakan terkelola baru	CloudWatch menambahkan kebijakan untuk peran terkait layanan baru untuk memungkinkan CloudWatch RUM memonitor data ke layanan terkait lainnya. AWS	29 November 2021

Perubahan	Deskripsi	Tanggal
CloudWatchSyntheticsFullAccess – Pembaruan pada kebijakan yang sudah ada	CloudWatch Synthetics menambahkan izin ke CloudWatchSyntheticsFullAccess, dan juga mengubah ruang lingkup satu izin. Izin `kms:ListAliases` ditambahkan sehingga pengguna dapat membuat daftar kunci AWS KMS yang tersedia dan dapat digunakan untuk mengenkripsi artefak canary. Izin `kms:DescribeKey` ditambahkan sehingga pengguna dapat melihat detail kunci yang akan digunakan untuk mengenkripsi artefak canary. Dan izin `kms:Decrypt` ditambahkan untuk memungkinkan pengguna mendekripsi artefak canary. Kemampuan dekripsi ini terbatas untuk digunakan pada sumber daya yang ada dalam bucket Amazon S3. Lingkup `Resource` dari izin `s3:GetBucketLocation` diubah dari * menjadi `arn:aws:s3:::*` .	29 September 2021

Perubahan	Deskripsi	Tanggal
CloudWatchSyntheticsFullAccess – Pembaruan pada kebijakan yang sudah ada	CloudWatch Synthetics menambahkan izin untuk. CloudWatchSyntheticsFullAccess Izin `lambda:UpdateFunctionCode` ditambahkan agar pengguna dengan kebijakan ini dapat mengubah versi runtime canary.	20 Juli 2021
AWSCloudWatchAlarms_ActionSSMIncidentsServiceRolePolicy— Kebijakan terkelola baru	CloudWatch menambahkan kebijakan IAM terkelola baru untuk memungkinkan CloudWatch untuk membuat insiden di Manajer AWS Systems Manager Insiden.	10 Mei 2021
CloudWatchAutomaticDashboardsAccess— Perbarui ke kebijakan yang ada	CloudWatch menambahkan izin ke kebijakan CloudWatchAutomaticDashboardsAccessterkelola. `synthetics:DescribeCanariesLastRun` Izin ditambahkan ke kebijakan ini untuk memungkinkan pengguna dasbor lintas akun melihat detail tentang proses kenari CloudWatch Synthetics.	20 April 2021
CloudWatch mulai melacak perubahan	CloudWatch mulai melacak perubahan untuk kebijakan yang AWS dikelola.	14 April 2021

Menggunakan tombol kondisi untuk membatasi akses ke ruang CloudWatch nama

Gunakan kunci kondisi IAM untuk membatasi pengguna untuk menerbitkan metrik hanya di CloudWatch ruang nama yang Anda tentukan.

Mengizinkan penerbitan hanya dalam satu namespace

Kebijakan berikut membatasi pengguna untuk menerbitkan metrik hanya di namespace yang disebut `MyCustomNamespace`.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Resource": "*",
        "Action": "cloudwatch:PutMetricData",
        "Condition": {
            "StringEquals": {
                "cloudwatch:namespace": "MyCustomNamespace"
            }
        }
    }
}
```

Tidak termasuk menerbitkan dari suatu namespace

Kebijakan berikut memungkinkan pengguna menerbitkan metrik di namespace apa pun kecuali `CustomNamespace2`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Resource": "*",
            "Action": "cloudwatch:PutMetricData"
        },
        {
            "Effect": "Deny",
            "Resource": "*",
```

```
            "Action": "cloudwatch:PutMetricData",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": "CustomNamespace2"
                }
            }
        }
    ]
}
```

Menggunakan kunci syarat untuk membatasi akses pengguna Wawasan Kontributor ke grup log

Untuk membuat aturan di Wawasan Kontributor dan melihat hasilnya, pengguna harus memiliki izin `cloudwatch:PutInsightRule`. Secara default, pengguna dengan izin ini dapat membuat aturan Contributor Insights yang mengevaluasi grup log apa pun di CloudWatch Log dan kemudian melihat hasilnya. Hasil dapat memuat data kontributor untuk grup log tersebut.

Anda dapat membuat kebijakan IAM dengan kunci syarat untuk memberikan izin kepada pengguna untuk menulis aturan Wawasan Kontributor pada beberapa grup log sambil mencegah mereka menulis aturan untuk dan melihat data ini dari grup log lainnya.

Untuk informasi selengkapnya tentang elemen `Condition` dalam kebijakan IAM, silakan lihat Elemen kebijakan JSON IAM: Syarat.

Izinkan akses untuk menulis aturan dan melihat hasil hanya untuk grup log tertentu

Kebijakan berikut memungkinkan akses pengguna untuk menulis aturan dan melihat hasil untuk grup log yang diberi nama `AllowedLogGroup` dan semua grup log yang memiliki nama yang dimulai dengan `AllowedWildCard`. Ini tidak memberikan akses untuk menulis aturan atau melihat aturan hasil untuk setiap grup log lainnya.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCertainLogGroups",
            "Effect": "Allow",
            "Action": "cloudwatch:PutInsightRule",
            "Resource": "arn:aws:cloudwatch:*:*:insight-rule/*",
```

```
            "Condition": {
                "ForAllValues:StringEqualsIgnoreCase": {
                    "cloudwatch:requestInsightRuleLogGroups": [
                        "AllowedLogGroup",
                        "AllowedWildcard*"
                    ]
                }
            }
        }
    ]
}
```

Tolak aturan penulisan untuk grup log tertentu, tetapi izinkan penulisan aturan untuk semua grup log lainnya

Kebijakan berikut secara eksplisit menolak akses pengguna untuk menulis aturan dan melihat hasil aturan untuk grup log yang diberi nama `ExplicitlyDeniedLogGroup`, tetapi memungkinkan untuk menulis aturan dan melihat hasil aturan untuk semua grup log lainnya.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowInsightRulesOnLogGroupsByDefault",
            "Effect": "Allow",
            "Action": "cloudwatch:PutInsightRule",
            "Resource": "arn:aws:cloudwatch:*:*:insight-rule/*"

        },
        {
            "Sid": "ExplicitDenySomeLogGroups",
            "Effect": "Deny",
            "Action": "cloudwatch:PutInsightRule",
            "Resource": "arn:aws:cloudwatch:*:*:insight-rule/*",
            "Condition": {
                "ForAllValues:StringEqualsIgnoreCase": {
                    "cloudwatch:requestInsightRuleLogGroups": [
                        "/test/alpine/ExplicitlyDeniedLogGroup"
                    ]
                }
            }
        }
    ]
```

```
}
```

Menggunakan kunci syarat untuk membatasi tindakan-tindakan alarm

Ketika CloudWatch alarm berubah status, mereka dapat melakukan tindakan yang berbeda seperti menghentikan dan menghentikan instans EC2 dan melakukan tindakan Systems Manager. Tindakan ini dapat dimulai ketika alarm berubah ke keadaan apa pun, termasuk ALARM, OK, atau INSUFFICIENT_DATA.

Gunakan kunci syarat `cloudwatch:AlarmActions` untuk memungkinkan pengguna untuk membuat alarm yang hanya dapat melakukan tindakan yang Anda tentukan ketika status alarm berubah. Misalnya, Anda dapat mengizinkan pengguna untuk membuat alarm yang hanya dapat melakukan tindakan yang bukan tindakan EC2.

Izinkan pengguna untuk membuat alarm yang hanya dapat mengirim notifikasi Amazon SNS atau melakukan tindakan Systems Manager

Kebijakan berikut membatasi pengguna untuk membuat alarm yang hanya dapat mengirim notifikasi Amazon SNS dan melakukan tindakan Systems Manager. Pengguna tidak dapat membuat alarm yang melakukan tindakan EC2.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateAlarmsThatCanPerformOnlySNSandSSMActions",
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricAlarm",
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringLike": {
                    "cloudwatch:AlarmActions": [
                        "arn:aws:sns:*",
                        "arn:aws:ssm:*"
                    ]
                }
            }
        }
    ]
}
```

Menggunakan peran terkait layanan untuk CloudWatch

Amazon CloudWatch menggunakan peran AWS Identity and Access Management terkait layanan (IAM). Peran terkait layanan adalah jenis unik peran IAM yang ditautkan langsung ke. CloudWatch Peran terkait layanan telah ditentukan sebelumnya oleh CloudWatch dan mencakup semua izin yang diperlukan layanan untuk memanggil AWS layanan lain atas nama Anda.

Satu peran terkait layanan dalam CloudWatch membuat pengaturan CloudWatch alarm yang dapat menghentikan, menghentikan, atau mem-boot ulang instans Amazon EC2 tanpa mengharuskan Anda menambahkan izin yang diperlukan secara manual. Peran lain terkait layanan memungkinkan akun pemantauan untuk mengakses CloudWatch data dari akun lain yang Anda tentukan, untuk membangun dasbor lintas Wilayah.

CloudWatch mendefinisikan izin peran terkait layanan ini, dan kecuali ditentukan lain, hanya CloudWatch dapat mengambil peran tersebut. Izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, serta bahwa kebijakan izin tidak dapat dilampirkan ke entitas IAM lainnya.

Anda dapat menghapus peran hanya setelah terlebih dahulu menghapus sumber daya terkaitnya. Pembatasan ini melindungi CloudWatch sumber daya Anda karena Anda tidak dapat secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, silakan lihat Layanan AWS yang bisa digunakan dengan IAM dan carilah layanan yang memiliki opsi Ya di kolom Peran Terkait Layanan. Pilih Ya dengan sebuah tautan untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Izin peran terkait layanan untuk CloudWatch alarm tindakan EC2

CloudWatch menggunakan peran terkait layanan bernama AWSServiceRoleForCloudWatchEvents—CloudWatch menggunakan peran terkait layanan ini untuk melakukan tindakan alarm Amazon EC2.

Peran AWSServiceRoleForCloudWatchEvents terkait layanan mempercayai layanan CloudWatch Acara untuk mengambil peran tersebut. CloudWatch Peristiwa memanggil tindakan instance terminate, stop, atau reboot saat dipanggil oleh alarm.

Kebijakan izin peran AWSServiceRoleForCloudWatchEvents terkait layanan memungkinkan CloudWatch Peristiwa menyelesaikan tindakan berikut di instans Amazon EC2:

- `ec2:StopInstances`
- `ec2:TerminateInstances`

- ec2:RecoverInstances
- ec2:DescribeInstanceRecoveryAttribute
- ec2:DescribeInstances
- ec2:DescribeInstanceStatus

Kebijakan izin peran AWSServiceRoleForCloudWatchCrossAccountterkait layanan memungkinkan CloudWatch untuk menyelesaikan tindakan berikut:

- sts:AssumeRole

Izin peran terkait layanan untuk Sinyal Aplikasi CloudWatch

CloudWatch Sinyal Aplikasi menggunakan peran terkait layanan bernama AWSServiceRoleForCloudWatchApplicationSignals— CloudWatch menggunakan peran terkait layanan ini untuk mengumpulkan CloudWatch data Log, data jejak X-Ray, data CloudWatch metrik, dan data penandaan dari aplikasi yang telah Anda aktifkan untuk Sinyal Aplikasi. CloudWatch

Peran AWSServiceRoleForCloudWatchApplicationSignalsterkait layanan mempercayai Sinyal CloudWatch Aplikasi untuk mengambil peran tersebut. Sinyal Aplikasi mengumpulkan data log, jejak, metrik, dan tanda dari akun yang Anda miliki.

AWSServiceRoleForCloudWatchApplicationSignalsMemiliki kebijakan IAM terlampir, dan kebijakan ini dinamai CloudWatchApplicationSignalsServiceRolePolicy. Kebijakan ini memberikan izin kepada Sinyal CloudWatch Aplikasi untuk mengumpulkan data pemantauan dan penandaan dari layanan terkait AWS lainnya. Ini mencakup izin untuk Sinyal Aplikasi untuk menyelesaikan tindakan berikut:

- xray:GetServiceGraph
- logs:StartQuery
- logs:GetQueryResults
- cloudwatch:GetMetricData
- cloudwatch:ListMetrics
- tag:GetResources

Isi lengkapnya CloudWatchApplicationSignalsServiceRolePolicyadalah sebagai berikut:

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "XRayPermission",
            "Effect": "Allow",
            "Action": [
                "xray:GetServiceGraph"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        },
        {
            "Sid": "CWLogsPermission",
            "Effect": "Allow",
            "Action": [
                "logs:StartQuery",
                "logs:GetQueryResults"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:/aws/appsignals/eks:*",
                "arn:aws:logs:*:*:log-group:/aws/appsignals/generic:*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        },
        {
            "Sid": "CWMetricsPermission",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:ListMetrics"
            ],
            "Resource": [
                "*"
            ],
```

```
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        },
        {
            "Sid": "TagsPermission",
            "Effect": "Allow",
            "Action": [
                "tag:GetResources"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        }
    ]
}
```

Izin peran terkait layanan untuk tindakan CloudWatch Systems Manager alarm OpsCenter

CloudWatch menggunakan peran terkait layanan bernama AWSServiceRoleForCloudWatchAlarms_ActionSSM— CloudWatch menggunakan peran terkait layanan ini untuk melakukan OpsCenter tindakan Systems Manager saat CloudWatch alarm masuk ke status ALARM.

Peran AWSServiceRoleForCloudWatchAlarms_ActionSSM terkait layanan mempercayai CloudWatch layanan untuk mengambil peran. CloudWatch alarm memanggil OpsCenter tindakan Systems Manager saat dipanggil oleh alarm.

Kebijakan izin peran AWSServiceRoleForCloudWatchAlarms_ActionSSMterkait layanan memungkinkan Systems Manager menyelesaikan tindakan berikut:

- `ssm:CreateOpsItem`

Izin peran terkait layanan untuk alarm tindakan CloudWatch Systems Manager Incident Manager

CloudWatch menggunakan peran terkait layanan bernama AWSServiceRoleForCloudWatchAlarms_ActionSSMIncidents— CloudWatch menggunakan peran terkait layanan ini untuk memulai insiden Manajer Insiden saat alarm masuk ke status CloudWatch ALARM.

Peran AWSServiceRoleForCloudWatchAlarms_ActionSSMIncidentsterkait layanan mempercayai CloudWatch layanan untuk mengambil peran. CloudWatch alarm memanggil tindakan Systems Manager Incident Manager saat dipanggil oleh alarm.

Kebijakan izin peran AWSServiceRoleForCloudWatchAlarms_ActionSSMIncidentsterkait layanan memungkinkan Systems Manager menyelesaikan tindakan berikut:

- `ssm-incidents:StartIncident`

Izin peran terkait layanan untuk lintas akun Lintas wilayah CloudWatch

CloudWatch menggunakan peran terkait layanan bernama AWSServiceRoleForCloudWatchCrossAccount— CloudWatch menggunakan peran ini untuk mengakses CloudWatch data di AWS akun lain yang Anda tentukan. SLR hanya memberikan izin peran asumsi untuk memungkinkan CloudWatch layanan mengambil peran dalam akun berbagi. Peran berbagi yang menyediakan akses ke data.

Kebijakan izin peran AWSServiceRoleForCloudWatchCrossAccountterkait layanan memungkinkan CloudWatch untuk menyelesaikan tindakan berikut:

- `sts:AssumeRole`

Peran AWSServiceRoleForCloudWatchCrossAccountterkait layanan mempercayai CloudWatch layanan untuk mengambil peran.

Izin peran terkait layanan untuk CloudWatch Performance Insights database

CloudWatch menggunakan peran terkait layanan bernama. AWSServiceRoleForCloudWatchMetrics_DbPerfInsights — CloudWatch menggunakan peran ini untuk mengambil metrik Performance Insights untuk membuat alarm dan snapshotting.

AWSServiceRoleForCloudWatchMetrics_DbPerfInsightsPeran terkait layanan memiliki kebijakan `AWSServiceRoleForCloudWatchMetrics_DbPerfInsightsServiceRolePolicy` IAM terlampir. Isi kebijakan tersebut adalah sebagai berikut:

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "pi:GetResourceMetrics"
   ],
   "Resource": "*",
   "Condition": {
    "StringEquals": {
     "aws:ResourceAccount": "${aws:PrincipalAccount}"
    }
   }
  }
 ]
}
```

Peran AWSServiceRoleForCloudWatchMetrics_DbPerfInsightsterkait layanan mempercayai CloudWatch layanan untuk mengambil peran.

Membuat peran terkait layanan untuk CloudWatch

Anda tidak perlu membuat secara manual peran terkait layanan ini. Pertama kali Anda membuat alarm di AWS Management Console, IAM CLI, atau IAM API CloudWatch, AWSServiceRoleForCloudWatchEvents membuat AWSServiceRoleForCloudWatchAlarms_ActionSSMdan untuk Anda.

Pertama kali Anda mengaktifkan penemuan layanan dan topologi, Application Signals menciptakan AWSServiceRoleForCloudWatchApplicationSignalsuntuk Anda.

Saat Anda pertama kali mengaktifkan akun untuk menjadi akun pemantauan untuk fungsionalitas lintas akun lintas wilayah, CloudWatch buat AWSServiceRoleForCloudWatchCrossAccountuntuk Anda.

Saat pertama kali membuat alarm yang menggunakan fungsi matematika `DB_PERF_INSIGHTS` metrik, CloudWatch buat AWSServiceRoleForCloudWatchMetrics_DbPerfInsightsuntuk Anda.

Untuk informasi selengkapnya, silakan lihat Membuat Peran Terkait Layanan dalam Panduan Pengguna IAM.

Mengedit peran terkait layanan untuk CloudWatch

CloudWatch tidak memungkinkan Anda untuk mengedit AWSServiceRoleForCloudWatchEvents,, AWSServiceRoleForCloudWatchAlarms_ActionSSMAWSServiceRoleForCloudWatchCrossAccount, atau AWSServiceRoleForCloudWatchMetrics_DbPerfInsightsperan. Setelah Anda membuat peran ini, Anda tidak dapat mengubah namanya karena berbagai entitas mungkin mereferensikan peran ini. Namun, Anda dapat mengedit deskripsi peran menggunakan IAM.

Menyunting deskripsi peran terkait layanan (konsol IAM)

Anda dapat menggunakan konsol IAM untuk menyunting deskripsi peran terkait layanan.

Untuk menyunting deskripsi peran terkait layanan (konsol IAM)

1. Di panel navigasi konsol IAM, pilih Peran.
2. Pilih nama peran yang akan diubah.
3. Di sisi kanan jauh dari Deskripsi peran, pilih Sunting.
4. Ketik deskripsi baru di kotak, dan pilih Simpan.

Menyunting deskripsi peran terkait layanan (AWS CLI)

Anda dapat menggunakan perintah IAM dari AWS Command Line Interface untuk mengedit deskripsi peran terkait layanan.

Untuk mengubah deskripsi peran terkait layanan (AWS CLI)

1. (Opsional) Untuk melihat deskripsi peran saat ini, gunakan perintah-perintah berikut:

   ```
   $ aws iam get-role --role-name role-name
   ```

   Gunakan nama peran, bukan ARN, untuk merujuk ke peran dengan perintah. AWS CLI Sebagai contoh, jika peran memiliki ARN berikut: `arn:aws:iam::123456789012:role/myrole`, referensi Anda ke peran sebagai **`myrole`**.
2. Untuk memperbarui deskripsi dari sebuah peran terkait layanan, gunakan perintah berikut:

   ```
   $ aws iam update-role-description --role-name role-name --description description
   ```

Menyunting deskripsi peran terkait layanan (IAM API)

Anda dapat menggunakan IAM API untuk menyunting deskripsi peran terkait layanan.

Untuk mengubah deskripsi peran terkait layanan (API)

1. (Opsional) Untuk melihat penjelasan peran saat ini, gunakan perintah berikut:

   GetRole
2. Untuk memperbarui penjelasan peran, gunakan perintah berikut:

   UpdateRoleDescription

Menghapus peran terkait layanan untuk CloudWatch

Jika Anda tidak lagi memiliki alarm yang secara otomatis menghentikan, menghentikan, atau me-reboot instans EC2, sebaiknya Anda menghapus peran tersebut. AWSServiceRoleForCloudWatchEvents

Jika Anda tidak lagi memiliki alarm yang melakukan OpsCenter tindakan Systems Manager, sebaiknya hapus AWSServiceRoleForCloudWatchAlarms_ActionSSM peran tersebut.

Jika Anda menghapus semua alarm yang menggunakan fungsi matematika `DB_PERF_INSIGHTS` metrik, sebaiknya hapus peran AWSServiceRoleForCloudWatchMetrics_DbPerfInsightsterkait layanan.

Dengan begitu, Anda tidak perlu lagi memantau atau memelihara entitas yang tidak digunakan. Namun, Anda harus membersihkan peran terkait layanan sebelum dapat menghapusnya.

Membersihkan peran terkait layanan

Sebelum Anda dapat menggunakan IAM untuk menghapus peran terkait layanan, Anda harus mengonfirmasi terlebih dahulu bahwa peran tersebut tidak memiliki sesi aktif dan menghapus sumber daya yang digunakan oleh peran tersebut.

Untuk memastikan peran terkait layanan memiliki sesi aktif di konsol IAM

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Peran. Pilih nama (bukan kotak centang) AWSServiceRoleForCloudWatchEvents peran.

3. Di halaman Ringkasan untuk peran yang dipilih, pilih Penasihat Akses dan tinjau aktivitas terbaru untuk peran terkait layanan.

> **Note**
>
> Jika Anda tidak yakin CloudWatch apakah menggunakan AWSServiceRoleForCloudWatchEvents peran tersebut, cobalah untuk menghapus peran tersebut. Jika layanan menggunakan peran tersebut, peran tidak dapat dihapus dan Anda dapat melihat Wilayah tempat peran tersebut digunakan. Jika peran tersebut sedang digunakan, Anda harus menunggu hingga sesi ini berakhir sebelum dapat menghapus peran tersebut. Anda tidak dapat mencabut sesi untuk peran terkait layanan.

Menghapus peran terkait layanan (Konsol IAM)

Anda dapat menggunakan konsol IAM untuk menghapus sebuah peran terkait layanan.

Untuk menghapus peran terkait layanan (konsol)

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi, silakan pilih Peran. Pilih kotak centang di samping nama peran yang ingin Anda hapus, bukan nama atau baris itu sendiri.
3. Untuk Tindakan peran, pilih Hapus peran.
4. Di kotak dialog konfirmasi, tinjau data layanan yang terakhir diakses, yang menunjukkan kapan masing-masing peran yang dipilih terakhir mengakses AWS layanan. Ini membantu Anda mengonfirmasi aktif tidaknya peran tersebut saat ini. Untuk melanjutkan, pilih Ya, Hapus.
5. Perhatikan notifikasi konsol IAM untuk memantau kemajuan penghapusan peran terkait layanan. Karena penghapusan peran terkait layanan IAM bersifat tidak sinkron, tugas penghapusan dapat berhasil atau gagal setelah Anda mengirimkan peran untuk dihapus. Jika tugas tersebut gagal, pilih Lihat detail atau Lihat Sumber Daya dari notifikasi untuk mempelajari alasan penghapusan gagal. Jika penghapusan gagal karena ada sumber daya di layanan yang digunakan oleh peran tersebut, maka alasan kegagalan tersebut mencakup daftar sumber daya.

Menghapus peran terkait layanan (AWS CLI)

Anda dapat menggunakan perintah IAM dari AWS Command Line Interface untuk menghapus peran terkait layanan.

Untuk menghapus peran terkait layanan (AWS CLI)

1. Karena peran terkait layanan tidak dapat dihapus jika sedang digunakan atau memiliki sumber daya terkait, Anda harus mengirimkan permintaan penghapusan. Permintaan tersebut dapat ditolak jika syarat ini tidak terpenuhi. Anda harus menangkap `deletion-task-id` dari tanggapan untuk memeriksa status tugas penghapusan. Ketik perintah berikut untuk mengirimkan permintaan penghapusan peran terkait layanan:

   ```
   $ aws iam delete-service-linked-role --role-name service-linked-role-name
   ```

2. Ketik perintah berikut untuk memeriksa status tugas penghapusan:

   ```
   $ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
   ```

   Status tugas penghapusan adalah `NOT_STARTED`, `IN_PROGRESS`, `SUCCEEDED`, atau `FAILED`. Jika penghapusan gagal, panggilan akan mengembalikan alasan kegagalan panggilan agar Anda dapat memecahkan masalah.

Menghapus peran terkait layanan (IAM API)

Anda dapat menggunakan IAM API untuk menghapus peran terkait layanan.

Untuk menghapus peran terkait layanan (API)

1. Untuk mengirimkan permintaan penghapusan peran terkait layanan, hubungi. DeleteServiceLinkedRole Dalam permintaan, tentukan nama peran yang ingin Anda hapus.

   Karena peran terkait layanan tidak dapat dihapus jika sedang digunakan atau memiliki sumber daya terkait, Anda harus mengirimkan permintaan penghapusan. Permintaan tersebut dapat ditolak jika syarat ini tidak terpenuhi. Anda harus menangkap `DeletionTaskId` dari tanggapan untuk memeriksa status tugas penghapusan.

2. Untuk memeriksa status penghapusan, panggil GetServiceLinkedRoleDeletionStatus. Di permintaan tersebut, tentukan `DeletionTaskId`.

   Kemungkinan status tugas penghapusan dapat berupa `NOT_STARTED`, `IN_PROGRESS`, `SUCCEEDED`, atau `FAILED`. Jika penghapusan gagal, panggilan akan mengembalikan alasan kegagalan panggilan agar Anda dapat memecahkan masalah.

CloudWatch pembaruan untuk peran AWS terkait layanan

Lihat detail tentang pembaruan kebijakan AWS terkelola CloudWatch sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di halaman Riwayat CloudWatch dokumen.

Perubahan	Deskripsi	Tanggal
AWSServiceRoleForCloudWatchApplicationSignals — Peran terkait layanan baru	CloudWatch menambahkan peran terkait layanan baru ini untuk memungkinkan Sinyal CloudWatch Aplikasi mengumpulkan data CloudWatch Log, data jejak X-Ray, data CloudWatch metrik, dan menandai data dari aplikasi yang telah Anda aktifkan untuk Sinyal Aplikasi. CloudWatch	9 November 2023
AWSServiceRoleForCloudWatchMetrics_DbPerfInsights— Peran terkait layanan baru	CloudWatch menambahkan peran terkait layanan baru ini untuk memungkinkan pengambilan metrik Performance Insights CloudWatch untuk mengkhawatirkan dan snapshotting. Kebijakan IAM dilampirkan pada peran ini, dan kebijakan tersebut memberikan izin CloudWatch untuk mengambil metrik Performance Insights atas nama Anda.	13 September 2023

Perubahan	Deskripsi	Tanggal
AWSServiceRoleForCloudWatchAlarms_ActionSSMIncidents— Peran terkait layanan baru	CloudWatch menambahkan peran terkait layanan baru untuk memungkinkan membuat insiden di CloudWatch AWS Systems Manager Manajer Insiden.	26 April 2021
CloudWatch mulai melacak perubahan	CloudWatch mulai melacak perubahan untuk peran terkait layanannya.	26 April 2021

Menggunakan peran terkait layanan untuk RUM CloudWatch

CloudWatch RUM menggunakan peran AWS Identity and Access Management terkait layanan (IAM). Peran terkait layanan adalah tipe peran IAM unik yang langsung terkait ke RUM. Peran terkait layanan telah ditentukan sebelumnya oleh RUM dan mencakup semua izin yang diperlukan layanan untuk memanggil AWS layanan lain atas nama Anda.

RUM menentukan izin peran terkait layanan ini, dan kecuali ditentukan lain, hanya CloudWatch yang dapat mengambil peran tersebut. Izin-izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, serta bahwa kebijakan izin tidak dapat dilampirkan ke entitas IAM lainnya.

Anda dapat menghapus peran tersebut hanya setelah pertama kali menghapus sumber dayanya yang terkait. Pembatasan ini melindungi sumber daya RUM Anda karena Anda tidak dapat secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, silakan lihat layanan AWS yang bisa digunakan dengan IAM dan carilah layanan yang memiliki opsi Ya di kolom Peran terkait layanan. Pilih Ya dengan sebuah tautan untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Izin peran terkait layanan untuk RUM

RUM menggunakan peran terkait layanan bernama AWSServiceRoleForCloudWatchRUM— peran ini memungkinkan RUM mengirim data AWS X-Ray jejak ke akun Anda, untuk monitor aplikasi yang Anda aktifkan penelusuran X-Ray.

Peran AWSServiceRoleForCloudWatchRUMterkait layanan mempercayai layanan X-Ray untuk mengambil peran tersebut. X-Ray mengirimkan data jejak ke akun Anda.

Peran AWSServiceRoleForCloudWatchRUMterkait layanan memiliki kebijakan IAM yang dilampirkan bernama RUM. AmazonCloudWatch ServiceRolePolicy Kebijakan ini memberikan izin kepada CloudWatch RUM untuk mempublikasikan data pemantauan ke AWS layanan terkait lainnya. Ini mencakup izin untuk memungkinkan RUM untuk menyelesaikan tindakan berikut:

- `xray:PutTraceSegments`
- `cloudwatch:PutMetricData`

Isi lengkap AmazonCloudWatchRUM ServiceRolePolicy adalah sebagai berikut.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "xray:PutTraceSegments"
   ],
   "Resource": [
    "*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": "cloudwatch:PutMetricData",
   "Resource": "*",
   "Condition": {
    "StringLike": {
     "cloudwatch:namespace": [
      "RUM/CustomMetrics/*",
      "AWS/RUM"
     ]
    }
   }
  }
 ]
}
```

Membuat sebuah peran terkait layanan untuk RUM

Anda tidak perlu membuat peran terkait layanan untuk CloudWatch RUM secara manual. Pertama kali Anda membuat monitor aplikasi dengan penelusuran X-Ray diaktifkan, atau memperbarui monitor aplikasi untuk menggunakan penelusuran X-Ray, RUM membuat AWSServiceRoleForCloudWatchRUMuntuk Anda.

Untuk informasi selengkapnya, silakan lihat Membuat Peran Terkait Layanan dalam Panduan Pengguna IAM.

Menyunting sebuah peran terkait layanan untuk RUM

CloudWatch RUM tidak memungkinkan Anda untuk mengedit AWSServiceRoleForCloudWatchRUMperan. Setelah Anda membuat peran ini, Anda tidak dapat mengubah namanya karena berbagai entitas mungkin mereferensikan peran ini. Namun, Anda dapat mengedit deskripsi peran menggunakan IAM.

Menyunting deskripsi peran terkait layanan (konsol IAM)

Anda dapat menggunakan konsol IAM untuk menyunting deskripsi peran terkait layanan.

Untuk menyunting deskripsi peran terkait layanan (konsol IAM)

1. Di panel navigasi konsol IAM, pilih Peran.
2. Pilih nama peran yang akan diubah.
3. Di sisi kanan jauh dari Deskripsi peran, pilih Sunting.
4. Ketik deskripsi baru di kotak, dan pilih Simpan.

Menyunting deskripsi peran terkait layanan (AWS CLI)

Anda dapat menggunakan perintah IAM dari AWS Command Line Interface untuk mengedit deskripsi peran terkait layanan.

Untuk mengubah deskripsi peran terkait layanan (AWS CLI)

1. (Opsional) Untuk melihat deskripsi peran saat ini, gunakan perintah-perintah berikut:

```
$ aws iam get-role --role-name role-name
```

Gunakan nama peran, bukan ARN, untuk merujuk ke peran dengan perintah. AWS CLI Sebagai contoh, jika peran memiliki ARN berikut: `arn:aws:iam::123456789012:role/myrole`, referensi Anda ke peran sebagai **`myrole`**.

2. Untuk memperbarui deskripsi dari sebuah peran terkait layanan, gunakan perintah berikut:

```
$ aws iam update-role-description --role-name role-name --description description
```

Menyunting deskripsi peran terkait layanan (IAM API)

Anda dapat menggunakan IAM API untuk menyunting deskripsi peran terkait layanan.

Untuk mengubah deskripsi peran terkait layanan (API)

1. (Opsional) Untuk melihat penjelasan peran saat ini, gunakan perintah berikut:

   GetRole
2. Untuk memperbarui penjelasan peran, gunakan perintah berikut:

   UpdateRoleDescription

Menghapus sebuah peran terkait layanan untuk RUM

Jika Anda tidak lagi mengaktifkan monitor aplikasi dengan X-Ray, kami sarankan Anda menghapus AWSServiceRoleForCloudWatchRUMperan tersebut.

Dengan begitu, Anda tidak perlu lagi memantau atau memelihara entitas yang tidak digunakan. Namun, Anda harus membersihkan peran terkait layanan sebelum dapat menghapusnya.

Membersihkan peran terkait layanan

Sebelum Anda dapat menggunakan IAM untuk menghapus peran terkait layanan, Anda harus mengonfirmasi terlebih dahulu bahwa peran tersebut tidak memiliki sesi aktif dan menghapus sumber daya yang digunakan oleh peran tersebut.

Untuk memastikan peran terkait layanan memiliki sesi aktif di konsol IAM

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.

2. Di panel navigasi, pilih Peran. Pilih nama (bukan kotak centang) dari AWSServiceRoleForCloudWatchRUM peran.
3. Di halaman Ringkasan untuk peran yang dipilih, pilih Penasihat Akses dan tinjau aktivitas terbaru untuk peran terkait layanan.

> **Note**
>
> Jika Anda tidak yakin apakah RUM menggunakan AWSServiceRoleForCloudWatchRUMperan tersebut, cobalah untuk menghapus peran tersebut. Jika layanan menggunakan peran tersebut, peran tidak dapat dihapus dan Anda dapat melihat Wilayah tempat peran tersebut digunakan. Jika peran tersebut sedang digunakan, Anda harus menunggu hingga sesi ini berakhir sebelum dapat menghapus peran tersebut. Anda tidak dapat mencabut sesi untuk peran terkait layanan.

Menghapus peran terkait layanan (Konsol IAM)

Anda dapat menggunakan konsol IAM untuk menghapus sebuah peran terkait layanan.

Untuk menghapus peran terkait layanan (konsol)

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi, silakan pilih Peran. Pilih kotak centang di samping nama peran yang ingin Anda hapus, bukan nama atau baris itu sendiri.
3. Untuk Tindakan peran, pilih Hapus peran.
4. Di kotak dialog konfirmasi, tinjau data layanan yang terakhir diakses, yang menunjukkan kapan masing-masing peran yang dipilih terakhir mengakses AWS layanan. Ini membantu Anda mengonfirmasi aktif tidaknya peran tersebut saat ini. Untuk melanjutkan, pilih Ya, Hapus.
5. Perhatikan notifikasi konsol IAM untuk memantau kemajuan penghapusan peran terkait layanan. Karena penghapusan peran terkait layanan IAM bersifat tidak sinkron, tugas penghapusan dapat berhasil atau gagal setelah Anda mengirimkan peran untuk dihapus. Jika tugas tersebut gagal, pilih Lihat detail atau Lihat Sumber Daya dari notifikasi untuk mempelajari alasan penghapusan gagal. Jika penghapusan gagal karena ada sumber daya di layanan yang digunakan oleh peran tersebut, maka alasan kegagalan tersebut mencakup daftar sumber daya.

Menghapus peran terkait layanan (AWS CLI)

Anda dapat menggunakan perintah IAM dari AWS Command Line Interface untuk menghapus peran terkait layanan.

Untuk menghapus peran terkait layanan (AWS CLI)

1. Karena peran terkait layanan tidak dapat dihapus jika sedang digunakan atau memiliki sumber daya terkait, Anda harus mengirimkan permintaan penghapusan. Permintaan tersebut dapat ditolak jika syarat ini tidak terpenuhi. Anda harus menangkap `deletion-task-id` dari tanggapan untuk memeriksa status tugas penghapusan. Ketik perintah berikut untuk mengirimkan permintaan penghapusan peran terkait layanan:

   ```
   $ aws iam delete-service-linked-role --role-name service-linked-role-name
   ```

2. Ketik perintah berikut untuk memeriksa status tugas penghapusan:

   ```
   $ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
   ```

   Status tugas penghapusan adalah `NOT_STARTED`, `IN_PROGRESS`, `SUCCEEDED`, atau `FAILED`. Jika penghapusan gagal, panggilan akan mengembalikan alasan kegagalan panggilan agar Anda dapat memecahkan masalah.

Menghapus peran terkait layanan (IAM API)

Anda dapat menggunakan IAM API untuk menghapus peran terkait layanan.

Untuk menghapus peran terkait layanan (API)

1. Untuk mengirimkan permintaan penghapusan peran terkait layanan, hubungi. DeleteServiceLinkedRole Dalam permintaan, tentukan nama peran yang ingin Anda hapus.

   Karena peran terkait layanan tidak dapat dihapus jika sedang digunakan atau memiliki sumber daya terkait, Anda harus mengirimkan permintaan penghapusan. Permintaan tersebut dapat ditolak jika syarat ini tidak terpenuhi. Anda harus menangkap `DeletionTaskId` dari tanggapan untuk memeriksa status tugas penghapusan.

2. Untuk memeriksa status penghapusan, panggil GetServiceLinkedRoleDeletionStatus. Di permintaan tersebut, tentukan `DeletionTaskId`.

Kemungkinan status tugas penghapusan dapat berupa `NOT_STARTED`, `IN_PROGRESS`, `SUCCEEDED`, atau `FAILED`. Jika penghapusan gagal, panggilan akan mengembalikan alasan kegagalan panggilan agar Anda dapat memecahkan masalah.

Menggunakan peran tertaut layanan untuk Wawasan Aplikasi CloudWatch

Wawasan Aplikasi CloudWatch menggunakan peran tertaut layanan (IAM) AWS Identity and Access Management. Peran tertaut layanan adalah jenis peran IAM unik yang tertaut langsung ke Wawasan Aplikasi CloudWatch. Peran tertaut layanan ditentukan sebelumnya oleh Wawasan Aplikasi CloudWatch dan mencakup semua izin yang diperlukan layanan untuk memanggil layanan AWS lainnya atas nama Anda.

Peran terkait layanan mempermudah persiapan Wawasan Aplikasi CloudWatch karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. Wawasan Aplikasi CloudWatch menetapkan izin peran tertaut layanan, dan kecuali jika ditentukan lain, hanya Wawasan Aplikasi CloudWatch dapat menjalankan perannya. Izin-izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, serta bahwa kebijakan izin tidak dapat dilampirkan ke entitas IAM lainnya.

Untuk informasi tentang layanan lain yang mendukung peran tertaut layanan, silakan lihat Layanan AWS yang bisa digunakan dengan IAM dan carilah layanan yang memiliki opsi Ya di kolom Peran Tertaut Layanan. Pilih Ya untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Izin peran terkait layanan untuk Wawasan Aplikasi CloudWatch

Wawasan Aplikasi CloudWatch menggunakan peran terkait layanan bernama AWSServiceRoleForApplicationInsights. Wawasan Aplikasi menggunakan peran ini untuk melakukan operasi seperti menganalisis grup sumber daya pelanggan, menciptakan stack CloudFormation untuk menciptakan alarm pada metrik, dan mengonfigurasi Agen CloudWatch pada instans EC2. Peran tertaut layanan memiliki kebijakan IAM yang terlampir padanya yang bernama `CloudwatchApplicationInsightsServiceLinkedRolePolicy`. Untuk pembaruan kebijakan ini, silakan lihat Wawasan Aplikasi diperbarui ke kebijakan terkelola AWS.

Kebijakan izin peran mengizinkan Wawasan Aplikasi CloudWatch menyelesaikan tindakan berikut pada sumber daya yang ditetapkan.

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:DescribeAlarmHistory",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:GetMetricData",
        "cloudwatch:ListMetrics",
        "cloudwatch:PutMetricAlarm",
        "cloudwatch:DeleteAlarms",
        "cloudwatch:PutAnomalyDetector",
        "cloudwatch:DeleteAnomalyDetector",
        "cloudwatch:DescribeAnomalyDetectors"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:FilterLogEvents",
        "logs:GetLogEvents",
        "logs:DescribeLogStreams",
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:DescribeRule"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudFormation:CreateStack",
        "cloudFormation:UpdateStack",
```

```
        "cloudFormation:DeleteStack",
        "cloudFormation:DescribeStackResources"
      ],
      "Resource": [
        "arn:aws:cloudformation:*:*:stack/ApplicationInsights-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudFormation:DescribeStacks",
        "cloudFormation:ListStackResources",
        "cloudFormation:ListStacks"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "tag:GetResources"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "resource-groups:ListGroupResources",
        "resource-groups:GetGroupQuery",
        "resource-groups:GetGroup"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "resource-groups:CreateGroup",
        "resource-groups:DeleteGroup"
      ],
```

```
      "Resource": [
        "arn:aws:resource-groups:*:*:group/ApplicationInsights-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "elasticloadbalancing:DescribeLoadBalancers",
        "elasticloadbalancing:DescribeTargetGroups",
        "elasticloadbalancing:DescribeTargetHealth"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:DescribeAutoScalingGroups"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ssm:PutParameter",
        "ssm:DeleteParameter",
        "ssm:AddTagsToResource",
        "ssm:RemoveTagsFromResource",
        "ssm:GetParameters"
      ],
      "Resource": "arn:aws:ssm:*:*:parameter/AmazonCloudWatch-ApplicationInsights-*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ssm:CreateAssociation",
        "ssm:UpdateAssociation",
        "ssm:DeleteAssociation",
        "ssm:DescribeAssociation"
      ],
      "Resource": [
```

```
        "arn:aws:ec2:*:*:instance/*",
        "arn:aws:ssm:*:*:association/*",
        "arn:aws:ssm:*:*:managed-instance/*",
        "arn:aws:ssm:*:*:document/AWSEC2-
ApplicationInsightsCloudwatchAgentInstallAndConfigure",
        "arn:aws:ssm:*:*:document/AWS-ConfigureAWSPackage",
        "arn:aws:ssm:*:*:document/AmazonCloudWatch-ManageAgent"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetOpsItem",
        "ssm:CreateOpsItem",
        "ssm:DescribeOpsItems",
        "ssm:UpdateOpsItem",
        "ssm:DescribeInstanceInformation"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ssm:AddTagsToResource"
      ],
      "Resource": "arn:aws:ssm:*:*:opsitem/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ssm:ListCommandInvocations",
        "ssm:GetCommandInvocation"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "ssm:SendCommand",
      "Resource": [
        "arn:aws:ec2:*:*:instance/*",
```

```
        "arn:aws:ssm:*:*:document/AWSEC2-CheckPerformanceCounterSets",
        "arn:aws:ssm:*:*:document/AWS-ConfigureAWSPackage",
        "arn:aws:ssm:*:*:document/AWSEC2-DetectWorkload",
        "arn:aws:ssm:*:*:document/AmazonCloudWatch-ManageAgent"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeVolumes",
        "ec2:DescribeVolumeStatus",
        "ec2:DescribeVpcs",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeNatGateways"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "rds:DescribeDBInstances",
        "rds:DescribeDBClusters"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "lambda:ListFunctions",
        "lambda:GetFunctionConfiguration",
        "lambda:ListEventSourceMappings"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "events:PutRule",
        "events:PutTargets",
        "events:RemoveTargets",
        "events:DeleteRule"
      ],
      "Resource": [
        "arn:aws:events:*:*:rule/AmazonCloudWatch-ApplicationInsights-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "xray:GetServiceGraph",
        "xray:GetTraceSummaries",
        "xray:GetTimeSeriesServiceStatistics",
        "xray:GetTraceGraph"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:ListTables",
        "dynamodb:DescribeTable",
        "dynamodb:DescribeContributorInsights",
        "dynamodb:DescribeTimeToLive"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "application-autoscaling:DescribeScalableTargets"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
```

```
      "Action": [
        "s3:ListAllMyBuckets",
        "s3:GetMetricsConfiguration",
        "s3:GetReplicationConfiguration"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "states:ListStateMachines",
        "states:DescribeExecution",
        "states:DescribeStateMachine",
        "states:GetExecutionHistory"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:GET"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ecs:DescribeClusters",
        "ecs:DescribeContainerInstances",
        "ecs:DescribeServices",
        "ecs:DescribeTaskDefinition",
        "ecs:DescribeTasks",
        "ecs:DescribeTaskSets",
        "ecs:ListClusters",
        "ecs:ListContainerInstances",
        "ecs:ListServices",
        "ecs:ListTasks"
      ],
```

```
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ecs:UpdateClusterSettings"
      ],
      "Resource": [
        "arn:aws:ecs:*:*:cluster/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "eks:DescribeCluster",
        "eks:DescribeFargateProfile",
        "eks:DescribeNodegroup",
        "eks:ListClusters",
        "eks:ListFargateProfiles",
        "eks:ListNodegroups",
        "fsx:DescribeFileSystems",
        "fsx:DescribeVolumes"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:GetSubscriptionAttributes",
        "sns:GetTopicAttributes",
        "sns:GetSMSAttributes",
        "sns:ListSubscriptionsByTopic",
        "sns:ListTopics"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
```

```
      "Action": [
        "sqs:ListQueues"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:DeleteSubscriptionFilter"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:PutSubscriptionFilter"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:*",
        "arn:aws:logs:*:*:destination:AmazonCloudWatch-ApplicationInsights-
LogIngestionDestination*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "elasticfilesystem:DescribeFileSystems"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "route53:GetHostedZone",
        "route53:GetHealthCheck",
        "route53:ListHostedZones",
        "route53:ListHealthChecks",
        "route53:ListQueryLoggingConfigs"
      ],
      "Resource": [
```

```
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "route53resolver:ListFirewallRuleGroupAssociations",
        "route53resolver:GetFirewallRuleGroup",
        "route53resolver:ListFirewallRuleGroups",
        "route53resolver:ListResolverEndpoints",
        "route53resolver:GetResolverQueryLogConfig",
        "route53resolver:ListResolverQueryLogConfigs",
        "route53resolver:ListResolverQueryLogConfigAssociations",
        "route53resolver:GetResolverEndpoint",
        "route53resolver:GetFirewallRuleGroupAssociation"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Anda harus mengonfigurasi izin untuk mengizinkan entitas IAM (seperti pengguna, grup, atau peran) untuk membuat, menyunting, atau menghapus peran terkait layanan. Untuk informasi selengkapnya, silakan lihat Izin Peran Tertaut Layanan di Panduan Pengguna IAM.

Membuat peran terkait layanan untuk Wawasan Aplikasi CloudWatch

Anda tidak perlu membuat peran terkait layanan secara manual. Ketika Anda membuat aplikasi Wawasan Aplikasi baru di AWS Management Console, Wawasan Aplikasi CloudWatch akan membuat peran tertaut layanan untuk Anda.

Jika Anda menghapus peran tertaut layanan ini, lalu ingin membuatnya lagi, Anda dapat menggunakan proses yang sama untuk membuat ulang peran tersebut di akun Anda. Ketika Anda membuat aplikasi Wawasan Aplikasi baru, Wawasan Aplikasi CloudWatch akan membuat lagi peran tertaut layanan untuk Anda.

Menyunting peran terkait layanan untuk Wawasan Aplikasi CloudWatch

Wawasan Aplikasi CloudWatch tidak memungkinkan Anda untuk menyunting AWSServiceRoleForApplicationInsights. Setelah Anda membuat peran terkait layanan, Anda tidak

dapat mengubah nama peran karena berbagai entitas mungkin mereferensikan peran tersebut. Namun demikian, Anda dapat menyunting penjelasan peran menggunakan IAM. Untuk informasi selengkapnya, silakan lihat Menyunting peran tertaut layanan dalam Panduan Pengguna IAM.

Menghapus peran terkait layanan untuk Wawasan Aplikasi CloudWatch

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, kami merekomendasikan agar Anda menghapus peran tersebut. Dengan demikian, Anda menghindari memiliki entitas tidak terpakai yang tidak dipantau atau dipelihara secara aktif. Namun demikian, Anda harus menghapus semua aplikasi dalam Wawasan Aplikasi sebelum dapat menghapus peran tersebut secara manual.

> **Note**
>
> Jika layanan Wawasan Aplikasi CloudWatch menggunakan peran tersebut ketika Anda mencoba menghapus sumber daya, maka penghapusan tersebut mungkin gagal. Jika hal itu terjadi, tunggu beberapa menit dan coba mengoperasikannya lagi.

Untuk menghapus sumber daya Wawasan Aplikasi CloudWatch yang digunakan dengan AWSServiceRoleForApplicationInsights

- Hapus semua aplikasi Wawasan Aplikasi CloudWatch Anda. Untuk informasi selengkapnya, silakan lihat "Menghapus Aplikasi Anda" dalam Panduan Pengguna Wawasan Aplikasi CloudWatch.

Untuk menghapus peran tertaut layanan secara manual menggunakan IAM

Gunakan konsol IAM, AWS CLI, atau API AWS untuk menghapus peran tertaut layanan AWSServiceRoleForApplicationInsights. Untuk informasi selengkapnya, silakan lihat Menghapus Peran Tertaut Layanan di Panduan Pengguna IAM.

Wilayah yang Didukung untuk peran tertaut layanan Wawasan Aplikasi CloudWatch

Wawasan Aplikasi CloudWatch mendukung penggunaan peran tertaut layanan di semua Wilayah AWS tempat layanan tersebut tersedia. Untuk informasi selengkapnya, silakan lihat Wilayah dan Titik Akhir Wawasan Aplikasi CloudWatch.

Kebijakan terkelola AWS untuk Wawasan Aplikasi Amazon CloudWatch

Kebijakan terkelola AWS adalah kebijakan mandiri yang dibuat dan oleh dilakukan AWS. Kebijakan terkelola AWS dirancang untuk memberikan izin bagi banyak kasus penggunaan umum sehingga Anda dapat mulai menetapkan izin kepada pengguna, grup, dan peran.

Perlu diingat bahwa kebijakan terkelola AWS mungkin tidak memberikan izin hak akses paling rendah untuk kasus penggunaan khusus Anda karena tersedia untuk digunakan semua pelanggan AWS. Kami menyarankan Anda untuk mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola pelanggan yang khusus untuk kasus penggunaan Anda.

Anda tidak dapat mengubah izin yang ada dalam kebijakan-kebijakan terkelola AWS. Jika AWS memperbarui izin yang ditentukan dalam sebuah kebijakan terkelola AWS, maka pembaruan itu akan mempengaruhi semua identitas pengguna utama (pengguna, grup, dan peran) yang terkait dengan kebijakan tersebut. AWS kemungkinan besar akan memperbarui kebijakan terkelola AWS saat sebuah Layanan AWS baru diluncurkan atau operasi API baru tersedia untuk layanan yang sudah ada.

Untuk informasi selengkapnya, silakan lihat kebijakan terkelola AWS di Panduan Pengguna IAM.

Kebijakan terkelola AWS: CloudWatchApplicationInsightsFullAccess

Anda dapat melampirkan kebijakan `CloudWatchApplicationInsightsFullAccess` ke identitas-identitas IAM Anda.

Kebijakan ini memberikan izin-izin administratif yang memungkinkan akses penuh ke fungsi Wawasan Aplikasi.

Detail izin

Kebijakan ini mencakup izin berikut.

- `applicationinsights` – Memungkinkan akses penuh ke fungsi Wawasan Aplikasi.
- `iam` – Memungkinkan Wawasan Aplikasi membuat peran terkait layanan, AWSServiceRoleForApplicationInsights. Ini diperlukan agar Wawasan Aplikasi dapat melakukan operasi seperti menganalisis grup sumber daya pelanggan, membuat tumpukan CloudFormation untuk membuat alarm pada metrik, dan mengonfigurasi Agen CloudWatch di instans EC2. Untuk informasi selengkapnya, silakan lihat Menggunakan peran tertaut layanan untuk Wawasan Aplikasi CloudWatch.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "applicationinsights:*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeVolumes",
        "rds:DescribeDBInstances",
        "rds:DescribeDBClusters",
        "sqs:ListQueues",
        "elasticloadbalancing:DescribeLoadBalancers",
        "elasticloadbalancing:DescribeTargetGroups",
        "elasticloadbalancing:DescribeTargetHealth",
        "autoscaling:DescribeAutoScalingGroups",
        "lambda:ListFunctions",
        "dynamodb:ListTables",
        "s3:ListAllMyBuckets",
        "sns:ListTopics",
        "states:ListStateMachines",
        "apigateway:GET",
        "ecs:ListClusters",
        "ecs:DescribeTaskDefinition",
        "ecs:ListServices",
        "ecs:ListTasks",
        "eks:ListClusters",
        "eks:ListNodegroups",
        "fsx:DescribeFileSystems",
```

```
        "logs:DescribeLogGroups",
        "elasticfilesystem:DescribeFileSystems"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/aws-service-role/application-insights.amazonaws.com/
AWSServiceRoleForApplicationInsights"
      ],
      "Condition": {
        "StringEquals": {
          "iam:AWSServiceName": "application-insights.amazonaws.com"
        }
      }
    }
  ]
}
```

Kebijakan terkelola AWS: CloudWatchApplicationInsightsReadOnlyAccess

Anda dapat melampirkan kebijakan `CloudWatchApplicationInsightsReadOnlyAccess` ke identitas-identitas IAM Anda.

Kebijakan ini memberikan izin-izin administratif yang memungkinkan akses baca-saja ke semua fungsi Wawasan Aplikasi.

Detail izin

Kebijakan ini mencakup izin berikut.

- `applicationinsights` – Memungkinkan akses baca-saja ke fungsi Wawasan Aplikasi.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "applicationinsights:Describe*",
                "applicationinsights:List*"
            ],
            "Resource": "*"
        }
    ]
}
```

Kebijakan terkelola AWS: CloudwatchApplicationInsightsServiceLinkedRolePolicy

Anda tidak dapat melampirkan CloudwatchApplicationInsightsServiceLinkedRolePolicy ke entitas IAM Anda. Kebijakan ini dilampirkan pada peran terkait layanan yang memungkinkan Wawasan Aplikasi untuk memantau sumber daya pelanggan. Untuk informasi selengkapnya, silakan lihat Menggunakan peran tertaut layanan untuk Wawasan Aplikasi CloudWatch.

Wawasan Aplikasi diperbarui ke kebijakan terkelola AWS

Tampilkan detail tentang pembaruan ke kebijakan terkelola AWS untuk Wawasan Aplikasi sejak layanan ini mulai melacak perubahan-perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan di umpan RSS di halaman Riwayat dokumen Wawasan Aplikasi.

Perubahan	Deskripsi	Tanggal
CloudwatchApplicationInsightsServiceLinkedRolePolicy – Pembaruan ke kebijakan yang sudah ada	Wawasan Aplikasi menambahkan izin baru ke daftar tumpukan CloudFormation. Izin ini diperlukan Wawasan Aplikasi Amazon CloudWatch untuk menganalisis dan	24 April 2023

Perubahan	Deskripsi	Tanggal
	memantau sumber daya AWS yang bersarang di tumpukan CloudFormation.	
CloudwatchApplicationInsightsServiceLinkedRolePolicy – Pembaruan ke kebijakan yang sudah ada	Wawasan Aplikasi menambahkan izin-izin baru untuk mendapatkan daftar sumber daya Amazon VPC dan Route 53. Izin ini diperlukan Wawasan Aplikasi Amazon CloudWatch untuk secara otomatis menyiapkan pemantauan jaringan praktik terbaik dengan Amazon CloudWatch.	23 Januari 2023
CloudwatchApplicationInsightsServiceLinkedRolePolicy – Pembaruan ke kebijakan yang sudah ada	Wawasan Aplikasi menambahkan izin baru untuk mendapatkan hasil invokasi perintah SSM. Izin ini diperlukan Wawasan Aplikasi Amazon CloudWatch untuk secara otomatis mendeteksi dan memantau beban kerja yang berjalan di instans Amazon EC2.	19 Desember 2022

Perubahan	Deskripsi	Tanggal
CloudwatchApplicationInsightsServiceLinkedRolePolicy – Pembaruan ke kebijakan yang sudah ada	Wawasan Aplikasi menambahkan izin-izin baru untuk mendeskripsikan sumber daya Amazon VPC dan Route 53. Izin ini diperlukan Wawasan Aplikasi Amazon CloudWatch untuk membaca konfigurasi sumber daya Amazon VPC dan Route 53 pelanggan, dan membantu pelanggan secara otomatis menyiapkan pemantauan jaringan praktik terbaik dengan Amazon CloudWatch.	19 Desember 2022
CloudwatchApplicationInsightsServiceLinkedRolePolicy – Pembaruan ke kebijakan yang sudah ada	Wawasan Aplikasi sudah menambahkan izin-izin baru untuk mendeskripsikan sumber daya EFS. Izin-izin ini diperlukan Wawasan Aplikasi Amazon CloudWatch untuk membaca konfigurasi sumber daya pelanggan Amazon EFS, dan membantu pelanggan secara otomatis menyiapkan praktik terbaik untuk pemantauan EFS dengan CloudWatch.	3 Oktober 2022

Perubahan	Deskripsi	Tanggal
CloudwatchApplicationInsightsServiceLinkedRolePolicy – Pembaruan ke kebijakan yang sudah ada	Wawasan Aplikasi sudah menambahkan izin-izin baru untuk mendeskripsikan sistem file EFS. Izin ini diperlukan Wawasan Aplikasi Amazon CloudWatch untuk membuat aplikasi berbasis akun dengan menanyakan semua sumber daya yang didukung dalam akun.	3 Oktober 2022
CloudwatchApplicationInsightsServiceLinkedRolePolicy – Pembaruan ke kebijakan yang sudah ada	Wawasan Aplikasi menambahkan izin baru untuk mengambil informasi tentang sumber daya FSx. Izin ini diperlukan Wawasan Aplikasi Amazon CloudWatch untuk memantau beban kerja dengan mengambil informasi yang cukup tentang volume FSx yang mendasarinya.	12 September 2022
Kebijakan terkelola AWS: CloudWatchApplicationInsightsFullAccess – Permbaruan ke kebijakan yang sudah ada	Wawasan Aplikasi menambahkan izin baru untuk menjelaskan grup log. Izin ini diperlukan Wawasan Aplikasi Amazon CloudWatch untuk memastikan bahwa izin yang benar untuk memantau grup log ada di akun saat membuat aplikasi baru.	24 Januari 2022

Perubahan	Deskripsi	Tanggal
CloudwatchApplicationInsightsServiceLinkedRolePolicy – Pembaruan ke kebijakan yang sudah ada	Wawasan Aplikasi menambahkan izin baru untuk membuat dan menghapus Filter Langganan Log CloudWatch. Izin ini diperlukan Wawasan Aplikasi Amazon CloudWatch untuk membuat Filter Langganan guna memfasilitasi pemantauan log sumber daya dalam aplikasi yang dikonfigurasi.	24 Januari 2022
CloudwatchApplicationInsightsServiceLinkedRolePolicy – Pembaruan ke kebijakan yang sudah ada	Wawasan Aplikasi menambahkan izin baru untuk menjelaskan grup target dan kesehatan target untuk Penyeimbang Beban Elastis. Izin ini diperlukan Wawasan Aplikasi Amazon CloudWatch untuk membuat aplikasi berbasis akun dengan menanyakan semua sumber daya yang didukung dalam akun.	4 November 2021

Perubahan	Deskripsi	Tanggal
CloudwatchApplicationInsightsServiceLinkedRolePolicy – Pembaruan ke kebijakan yang sudah ada	Wawasan Aplikasi menambahkan izin baru untuk menjalankan dokumen SSM `AmazonCloudWatch-ManageAgent` di instans Amazon EC2. Izin ini diperlukan Wawasan Aplikasi Amazon CloudWatch untuk membersihkan file konfigurasi agen CloudWatch yang dibuat oleh Wawasan Aplikasi.	30 September 2021

Perubahan	Deskripsi	Tanggal
CloudwatchApplicationInsightsServiceLinkedRolePolicy – Pembaruan ke kebijakan yang sudah ada	Wawasan Aplikasi menambahkan izin baru untuk mendukung pemantauan aplikasi berbasis akun untuk onboard dan memantau semua sumber daya yang didukung di akun Anda. Izin ini diperlukan Wawasan Aplikasi Amazon CloudWatch untuk menanyakan, menandai sumber daya, dan membuat grup untuk sumber daya ini. Wawasan Aplikasi menambahkan izin baru untuk mendukung pemantauan topik SNS. Izin ini diperlukan Wawasan Aplikasi Amazon CloudWatch untuk mengumpulkan metadata dari sumber daya SNS guna mengonfigurasi pemantauan topik SNS.	15 September 2021

Perubahan	Deskripsi	Tanggal
Kebijakan terkelola AWS: CloudWatchApplicationInsightsFullAccess – Permbaruan ke kebijakan yang sudah ada	Wawasan Aplikasi menambahkan izin baru untuk menjelaskan dan mendaftar sumber daya yang didukung. Izin ini diperlukan Wawasan Aplikasi Amazon CloudWatch untuk membuat aplikasi berbasis akun dengan menanyakan semua sumber daya yang didukung dalam akun.	15 September 2021
CloudwatchApplicationInsightsServiceLinkedRolePolicy – Pembaruan ke kebijakan yang sudah ada	Wawasan Aplikasi menambahkan izin baru untuk menjelaskan sumber daya FSX. Izin ini diperlukan Wawasan Aplikasi Amazon CloudWatch untuk membaca konfigurasi sumber daya FSx, dan membantu pelanggan secara otomatis mengatur pemantauan FSx praktik terbaik dengan CloudWatch.	31 Agustus 2021

Perubahan	Deskripsi	Tanggal
CloudwatchApplicationInsightsServiceLinkedRolePolicy – Pembaruan ke kebijakan yang sudah ada	Wawasan Aplikasi menambahkan izin baru untuk menjelaskan dan mendaftar sumber daya layanan ECS dan EKS. Izin ini diperlukan Wawasan Aplikasi Amazon CloudWatch untuk membaca konfigurasi sumber daya kontainer pelanggan, dan membantu pelanggan secara otomatis mengatur pemantauan kontainer pelanggan praktik terbaik dengan CloudWatch.	18 Mei 2021
CloudwatchApplicationInsightsServiceLinkedRolePolicy – Pembaruan ke kebijakan yang sudah ada	Wawasan Aplikasi menambahkan izin baru sehingga memungkinkan OpsCenter memberi tanda OpsItems menggunakan tindakan `ssm:AddTagsToResource` pada sumber daya dengan tipe sumber daya `opsitem`. Izin ini diperlukan OpsCenter. Wawasan Aplikasi Amazon CloudWatch membuat OpsItems sehingga pelanggan dapat menyelesaikan masalah menggunakan SSM OpsCenter AWS.	13 April 2021

Perubahan	Deskripsi	Tanggal
Wawasan Aplikasi mulai melacak perubahan	Wawasan Aplikasi mulai melacak perubahan untuk kebijakan terkelola AWS.	13 April 2021

CloudWatch Referensi izin Amazon

Tabel berikut mencantumkan setiap operasi CloudWatch API dan tindakan terkait yang dapat Anda berikan izin untuk melakukan tindakan. Anda menentukan tindakan dalam bidang `Action` kebijakan, dan Anda menentukan karakter wildcard (*) sebagai nilai sumber daya dalam bidang `Resource` kebijakan.

Anda dapat menggunakan kunci kondisi AWS -wide dalam CloudWatch kebijakan Anda untuk menyatakan kondisi. Untuk daftar lengkap kunci AWS -wide, lihat Kunci Konteks Kondisi AWS Global dan IAM di Panduan Pengguna IAM.

> **Note**
>
> Untuk menentukan tindakan, gunakan awalan `cloudwatch:` diikuti dengan nama operasi API. Misalnya:`cloudwatch:GetMetricData`,`cloudwatch:ListMetrics`, atau `cloudwatch:*` (untuk semua CloudWatch tindakan).

Topik

- CloudWatch Operasi API dan izin yang diperlukan untuk tindakan
- CloudWatch Operasi API Contributor Insights dan izin yang diperlukan untuk tindakan
- CloudWatch Operasi API peristiwa dan izin yang diperlukan untuk tindakan
- CloudWatch Log operasi API dan izin yang diperlukan untuk tindakan
- Operasi API Amazon EC2 dan izin yang diperlukan untuk tindakan
- Operasi-operasi API Amazon EC2 Auto Scaling dan izin-izin yang diperlukan untuk tindakan

CloudWatch Operasi API dan izin yang diperlukan untuk tindakan

CloudWatch Operasi API	Izin yang diperlukan (tindakan API)
DeleteAlarms	`cloudwatch:DeleteAlarms` Diperlukan untuk menghapus alarm.
DeleteDashboards	`cloudwatch:DeleteDashboards` Diperlukan untuk menghapus dasbor.
DeleteMetricStream	`cloudwatch:DeleteMetricStream` Diperlukan untuk menghapus aliran metrik.
DescribeAlarmHistory	`cloudwatch:DescribeAlarmHistory` Diperlukan untuk melihat riwayat alarm. Untuk mengambil informasi tentang alarm gabungan, izin `cloudwatch:DescribeAlarmHistory` Anda harus memiliki ruang lingkup `*`. Anda tidak dapat mengembalikan informasi tentang alarm gabungan jika izin `cloudwatch:DescribeAlarmHistory` Anda memiliki cakupan yang lebih sempit.
DescribeAlarms	`cloudwatch:DescribeAlarms` Diperlukan untuk mengambil informasi mengenai alarm. Untuk mengambil informasi tentang alarm gabungan, izin `cloudwatch:DescribeAlarms` Anda harus memiliki ruang lingkup `*`. Anda tidak dapat mengembalikan informasi tentang alarm gabungan jika izin `cloudwatch:DescribeAlarms` Anda memiliki cakupan yang lebih sempit.

CloudWatch Operasi API	Izin yang diperlukan (tindakan API)
DescribeAlarmsForMetric	`cloudwatch:DescribeAlarmsForMetric` Diperlukan untuk melihat alarm untuk metrik.
DisableAlarmActions	`cloudwatch:DisableAlarmActions` Diperlukan untuk menonaktifkan tindakan alarm.
EnableAlarmActions	`cloudwatch:EnableAlarmActions` Diperlukan untuk mengaktifkan tindakan alarm.
GetDashboard	`cloudwatch:GetDashboard` Diperlukan untuk menampilkan data tentang dasbor yang sudah ada.
GetMetricData	`cloudwatch:GetMetricData` Diperlukan untuk membuat grafik data metrik di CloudWatch konsol, untuk mengambil sejumlah besar data metrik, dan melakukan matematika metrik pada data tersebut.
GetMetricStatistics	`cloudwatch:GetMetricStatistics` Diperlukan untuk melihat grafik di bagian lain CloudWatch konsol dan di widget dasbor.
GetMetricStream	`cloudwatch:GetMetricStream` Diperlukan untuk melihat informasi tentang pengaliran metrik.

CloudWatch Operasi API	Izin yang diperlukan (tindakan API)
GetMetricWidgetImage	`cloudwatch:GetMetricWidgetImage` Diperlukan untuk mengambil grafik snapshot dari satu atau beberapa CloudWatch metrik sebagai gambar bitmap.
ListDashboards	`cloudwatch:ListDashboards` Diperlukan untuk melihat daftar CloudWatch dasbor di akun Anda.
ListMetrics	`cloudwatch:ListMetrics` Diperlukan untuk melihat atau mencari nama metrik di dalam CloudWatch konsol dan di CLI. Diperlukan untuk memilih metrik pada widget dasbor.
ListMetricStreams	`cloudwatch:ListMetricStreams` Diperlukan untuk melihat atau mencari daftar aliran metrik di akun.
PutCompositeAlarm	`cloudwatch:PutCompositeAlarm` Diperlukan untuk membuat alarm gabungan Untuk membuat alarm gabungan, izin `cloudwatch:PutCompositeAlarm` Anda harus memiliki lingkup `*`. Anda tidak dapat mengembalikan informasi tentang alarm gabungan jika izin `cloudwatch:PutCompositeAlarm` Anda memiliki cakupan yang lebih sempit.

CloudWatch Operasi API	Izin yang diperlukan (tindakan API)
PutDashboard	`cloudwatch:PutDashboard` Diperlukan untuk membuat dasbor atau memperbarui dasbor yang sudah ada.
PutMetricAlarm	`cloudwatch:PutMetricAlarm` Diperlukan untuk membuat atau memperbarui alarm.
PutMetricData	`cloudwatch:PutMetricData` Diperlukan untuk membuat metrik.
PutMetricStream	`cloudwatch:PutMetricStream` Diperlukan untuk membuat pengaliran metrik.
SetAlarmState	`cloudwatch:SetAlarmState` Diperlukan untuk mengatur status alarm secara manual.
StartMetricStreams	`cloudwatch:StartMetricStreams` Diperlukan untuk memulai aliran metrik dalam sebuah aliran metrik.
StopMetricStreams	`cloudwatch:StopMetricStreams` Diperlukan untuk menghentikan aliran metrik dalam sebuah aliran metrik untuk sementara waktu.

CloudWatch Operasi API	Izin yang diperlukan (tindakan API)
TagResource	`cloudwatch:TagResource` Diperlukan untuk menambahkan atau memperbarui tag pada CloudWatch sumber daya seperti alarm dan aturan Wawasan Kontributor.
UntagResource	`cloudwatch:UntagResource` Diperlukan untuk menghapus tag dari CloudWatch sumber daya.

CloudWatch Operasi API Contributor Insights dan izin yang diperlukan untuk tindakan

> **Important**
>
> Saat Anda memberikan `cloudwatch:PutInsightRule` izin kepada pengguna, secara default pengguna tersebut dapat membuat aturan yang mengevaluasi grup log apa pun di CloudWatch Log. Anda dapat menambahkan ketentuan kebijakan IAM yang membatasi izin ini agar pengguna dapat menyertakan dan mengecualikan grup log tertentu. Untuk informasi selengkapnya, lihat Menggunakan kunci syarat untuk membatasi akses pengguna Wawasan Kontributor ke grup log.

CloudWatch Operasi API Contributor Insights	Izin yang diperlukan (tindakan API)
DeleteInsightRules	`cloudwatch:DeleteInsightRules` Wajib menghapus aturan Wawasan Kontribut or.
DescribeInsightRules	`cloudwatch:DescribeInsightRules` Diperlukan untuk melihat aturan Wawasan Kontributor di akun Anda.

CloudWatch Operasi API Contributor Insights	Izin yang diperlukan (tindakan API)
EnableInsightRules	`cloudwatch:EnableInsightRules` Diperlukan untuk mengaktifkan aturan Wawasan Kontributor.
GetInsightRuleReport	`cloudwatch:GetInsightRuleReport` Diperlukan untuk mengambil data rangkaian waktu dan statistik lain yang dikumpulkan oleh aturan Wawasan Kontributor.
PutInsightRule	`cloudwatch:PutInsightRule` Diperlukan untuk membuat aturan Wawasan Kontributor. Lihat catatan Penting di awal tabel ini.

CloudWatch Operasi API peristiwa dan izin yang diperlukan untuk tindakan

CloudWatch Operasi API Acara	Izin yang diperlukan (tindakan API)
DeleteRule	`events:DeleteRule` Perlu menghapus aturan.
DescribeRule	`events:DescribeRule` Wajib mencantumkan perincian tentang aturan.
DisableRule	`events:DisableRule` Wajib menonaktifkan aturan.
EnableRule	`events:EnableRule` Diperlukan untuk mengaktifkan aturan.
ListRuleNamesByTarget	`events:ListRuleNamesByTarget`

CloudWatch Operasi API Acara	Izin yang diperlukan (tindakan API)
	Wajib mencantumkan aturan terkait target.
ListRules	`events:ListRules` Wajib mencantumkan semua aturan di akun Anda.
ListTargetsByRule	`events:ListTargetsByRule` Wajib mencantumkan semua target yang terkait dengan aturan.
PutEvents	`events:PutEvents` Diperlukan untuk menambahkan peristiwa khusus yang dapat dicocokkan dengan aturan.
PutRule	`events:PutRule` Diperlukan untuk membuat atau memperbarui aturan.
PutTargets	`events:PutTargets` Diperlukan untuk menambahkan target ke aturan.
RemoveTargets	`events:RemoveTargets` Diperlukan untuk menghapus target dari aturan.
TestEventPattern	`events:TestEventPattern` Diperlukan untuk menguji pola peristiwa terhadap peristiwa tertentu.

CloudWatch Log operasi API dan izin yang diperlukan untuk tindakan

CloudWatch Operasi API log	Izin yang diperlukan (tindakan API)
CancelExportTask	`logs:CancelExportTask` Diperlukan untuk membatalkan tugas ekspor yang tertunda atau berjalan.
CreateExportTask	`logs:CreateExportTask` Diperlukan untuk mengekspor data dari grup log ke buket Amazon S3.
CreateLogGroup	`logs:CreateLogGroup` Diperlukan untuk membuat grup log baru.
CreateLogStream	`logs:CreateLogStream` Diperlukan untuk membuat aliran log baru dalam grup log.
DeleteDestination	`logs:DeleteDestination` Diperlukan untuk menghapus tujuan log dan menonaktifkan penyaring berlangganan apa pun.
DeleteLogGroup	`logs:DeleteLogGroup` Diperlukan untuk menghapus grup log dan semua peristiwa log yang diarsipkan yang terkait.
DeleteLogStream	`logs:DeleteLogStream` Diperlukan untuk menghapus aliran log dan peristiwa log yang diarsipkan yang terkait.
DeleteMetricFilter	`logs:DeleteMetricFilter`

CloudWatch Operasi API log	Izin yang diperlukan (tindakan API)
	Diperlukan untuk menghapus penyaring metrik yang terkait dengan grup log.
DeleteQueryDefinition	`logs:DeleteQueryDefinition` Diperlukan untuk menghapus definisi kueri yang disimpan di Wawasan CloudWatch Log.
DeleteResourcePolicy	`logs:DeleteResourcePolicy` Diperlukan untuk menghapus kebijakan sumber daya CloudWatch Log.
DeleteRetentionPolicy	`logs:DeleteRetentionPolicy` Diperlukan untuk menghapus kebijakan penyimpanan grup log.
DeleteSubscriptionFilter	`logs:DeleteSubscriptionFilter` Diperlukan untuk menghapus penyaring berlangganan yang terkait dengan grup log.
DescribeDestinations	`logs:DescribeDestinations` Diperlukan untuk melihat semua destinasi yang terkait dengan akun.
DescribeExportTasks	`logs:DescribeExportTasks` Diperlukan untuk melihat semua tugas ekspor yang terkait dengan akun.
DescribeLogGroups	`logs:DescribeLogGroups` Diperlukan untuk melihat semua grup log yang terkait dengan akun.

CloudWatch Operasi API log	Izin yang diperlukan (tindakan API)
DescribeLogStreams	`logs:DescribeLogStreams` Diperlukan untuk melihat semua aliran log yang terkait dengan grup log.
DescribeMetricFilters	`logs:DescribeMetricFilters` Diperlukan untuk melihat semua metrik yang terkait dengan grup log.
DescribeQueryDefinitions	`logs:DescribeQueryDefinitions` Diperlukan untuk melihat daftar definisi kueri yang disimpan di Wawasan CloudWatch Log.
DescribeQueries	`logs:DescribeQueries` Diperlukan untuk melihat daftar kueri Wawasan CloudWatch Log yang dijadwalkan, dijalankan, atau baru-baru ini dikeluarkan.
DescribeResourcePolicies	`logs:DescribeResourcePolicies` Diperlukan untuk melihat daftar kebijakan sumber daya CloudWatch Log.
DescribeSubscriptionFilters	`logs:DescribeSubscriptionFilters` Diperlukan untuk melihat semua penyaring berlangganan yang terkait dengan grup log.
FilterLogEvents	`logs:FilterLogEvents` Diperlukan untuk mengurutkan peristiwa log berdasarkan pola penyaringan grup log.

CloudWatch Operasi API log	Izin yang diperlukan (tindakan API)
GetLogEvents	`logs:GetLogEvents` Diperlukan untuk mengambil kejadian log dari aliran log.
GetLogGroupFields	`logs:GetLogGroupFields` Diperlukan untuk mengambil daftar kolom yang disertakan dalam peristiwa log di grup log.
GetLogRecord	`logs:GetLogRecord` Diperlukan untuk mengambil rincian dari satu peristiwa log.
GetQueryResults	`logs:GetQueryResults` Diperlukan untuk mengambil hasil kueri Wawasan CloudWatch Log.
ListTagsLogGroup	`logs:ListTagsLogGroup` Diperlukan untuk membuat daftar tag yang terkait dengan grup log.
PutDestination	`logs:PutDestination` Diperlukan untuk membuat atau memperbarui aliran log tujuan (seperti aliran Kinesis).
PutDestinationPolicy	`logs:PutDestinationPolicy` Diperlukan untuk membuat atau memperbarui kebijakan akses yang terkait dengan tujuan log yang sudah ada.

CloudWatch Operasi API log	Izin yang diperlukan (tindakan API)
PutLogEvents	`logs:PutLogEvents` Diperlukan untuk mengunggah kumpulan peristiwa log ke aliran log.
PutMetricFilter	`logs:PutMetricFilter` Diperlukan untuk membuat atau memperbarui penyaring metrik dan mengaitkannya dengan grup log.
PutQueryDefinition	`logs:PutQueryDefinition` Diperlukan untuk menyimpan kueri di Wawasan CloudWatch Log.
PutResourcePolicy	`logs:PutResourcePolicy` Diperlukan untuk membuat kebijakan sumber daya CloudWatch Log.
PutRetentionPolicy	`logs:PutRetentionPolicy` Diperlukan untuk mengatur jumlah hari untuk menyimpan peristiwa (penyimpanan) log dalam grup log.
PutSubscriptionFilter	`logs:PutSubscriptionFilter` Diperlukan untuk membuat atau memperbarui penyaring berlangganan dan mengaitkannya dengan grup log.
StartQuery	`logs:StartQuery` Diperlukan untuk memulai kueri Wawasan CloudWatch Log.

CloudWatch Operasi API log	Izin yang diperlukan (tindakan API)
StopQuery	`logs:StopQuery` Diperlukan untuk menghentikan kueri Wawasan CloudWatch Log yang sedang berlangsung.
TagLogGroup	`logs:TagLogGroup` Perlu menambahkan atau memperbarui tag grup log.
TestMetricFilter	`logs:TestMetricFilter` Diperlukan untuk menguji pola penyaringan terhadap sampel pesan peristiwa log.

Operasi API Amazon EC2 dan izin yang diperlukan untuk tindakan

Operasi API Amazon EC2	Izin yang diperlukan (tindakan API)
DescribeInstanceStatus	`ec2:DescribeInstanceStatus` Diperlukan untuk melihat rincian status contoh EC2.
DescribeInstances	`ec2:DescribeInstances` Diperlukan untuk melihat rincian contoh EC2.
RebootInstances	`ec2:RebootInstances` Diperlukan untuk menyalakan ulang contoh EC2.
StopInstances	`ec2:StopInstances` Diperlukan untuk menghentikan contoh EC2.
TerminateInstances	`ec2:TerminateInstances`

Operasi API Amazon EC2	Izin yang diperlukan (tindakan API)
	Diperlukan untuk mengakhiri contoh EC2.

Operasi-operasi API Amazon EC2 Auto Scaling dan izin-izin yang diperlukan untuk tindakan

Operasi-operasi API Amazon EC2 Auto Scaling	Izin yang diperlukan (tindakan API)
Penskalaan	`autoscaling:Scaling` Diperlukan untuk menskalakan sebuah grup Auto Scaling.
Pemicu	`autoscaling:Trigger` Diperlukan untuk memicu tindakan sebuah penskalaan otomatis Auto Scaling.

Validasi kepatuhan untuk Amazon CloudWatch

Auditor pihak ketiga menilai keamanan dan kepatuhan Amazon CloudWatch sebagai bagian dari beberapa program kepatuhan AWS. Program ini mencakup SOC, PCI, FedRAMP, HIPAA, dan lainnya.

Untuk daftar layanan AWS dalam cakupan program kepatuhan tertentu, silakan lihat Layanan AWS dalam Cakupan berdasarkan Program Kepatuhan . Untuk informasi umum, silakan lihat Program Kepatuhan AWS .

Anda bisa mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, silakan lihat Mengunduh Laporan di AWS Artifact .

Tanggung jawab kepatuhan Anda saat menggunakan Amazon CloudWatch ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, serta hukum dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- Panduan Quick Start Keamanan dan Kepatuhan – Panduan deployment ini membahas pertimbangan arsitektur dan menyediakan langkah–langkah untuk melakukan deployment terhadap lingkungan dasar di AWS yang menjadi fokus keamanan dan kepatuhan.
- Merancang Laporan Resmi Keamanan dan Kepatuhan HIPAA – Laporan resmi ini menjelaskan bagaimana perusahaan dapat menggunakan AWS untuk membuat aplikasi yang memenuhi syarat HIPAA.
- Sumber Daya Kepatuhan AWS – Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- Mengevaluasi Sumber Daya dengan Aturan di Panduan Developer AWS Config – AWS Config menilai seberapa baik konfigurasi sumber daya Anda dalam mematuhi praktik–praktik internal, pedoman industri, dan regulasi internal.
- AWS Security Hub – Layanan AWS ini menyediakan sebuah pandangan yang komprehensif tentang status keamanan Anda dalam AWS yang membantu Anda memeriksa kepatuhan Anda terhadap standar industri dan praktik terbaik untuk keamanan.

Ketahanan di Amazon CloudWatch

Infrastruktur global AWS dibangun di sekitar Wilayah AWS dan Zona Ketersediaan. Wilayah memberikan beberapa Zona Ketersediaan yang terpisah dan terisolasi secara fisik, yang terkoneksi melalui jaringan latensi rendah, throughput tinggi, dan sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara otomatis melakukan fail over di antara zona tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur pusat data tunggal atau multi tradisional.

Untuk informasi selengkapnya tentang Wilayah AWS dan Zona Ketersediaan, silakan lihat Infrastruktur Global AWS.

Keamanan infrastruktur di Amazon CloudWatch

Sebagai layanan terkelola, Amazon CloudWatch dilindungi oleh keamanan jaringan global AWS. Untuk informasi tentang layanan keamanan AWS dan cara AWS melindungi infrastruktur, silakan lihat Keamanan Cloud AWS. Untuk mendesain lingkungan AWS Anda menggunakan praktik terbaik untuk keamanan infrastruktur, silakan lihat Perlindungan Infrastruktur dalam Kerangka Kerja yang Dirancang dengan Baik AWS Pilar Keamanan.

Anda dapat menggunakan panggilan API yang diterbitkan AWS untuk mengakses CloudWatch melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Pengangkutan (TLS) Kami mensyaratkan TLS 1.2 dan merekomendasikan TLS 1.3.
- Suite cipher dengan perfect forward secrecy (PFS) seperti DHE (Ephemeral Diffie-Hellman) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Sebagian besar sistem-sistem modern seperti Java 7 dan versi yang lebih baru mendukung mode-mode ini.

Selain itu, permintaan harus ditandatangani menggunakan ID kunci akses dan kunci akses rahasia yang dikaitkan dengan pengguna utama IAM. Atau Anda dapat menggunakan AWS Security Token Service (AWS STS) untuk membuat kredensial keamanan sementara untuk menandatangani permintaan.

Isolasi jaringan

Cloud Privat Virtual (VPC) adalah jaringan virtual dalam area Anda yang diisolasi secara logika dalam Amazon Web Services Cloud. Subnet adalah serangkaian alamat IP dalam sebuah VPC. Anda dapat men-deploy berbagai sumber daya AWS di subnet VPC Anda. Misalnya, Anda dapat men-deploy instans Amazon EC2, klaster EMR, dan tabel DynamoDB dalam subnet. Untuk informasi selengkapnya, silakan lihat ACL Jaringan di Panduan Pengguna Amazon VPC.

Guna mengaktifkan CloudWatch untuk berkomunikasi dengan sumber daya di VPC tanpa melalui internet publik, gunakan AWS PrivateLink. Untuk informasi selengkapnya, silakan lihat Menggunakan CloudWatch dan CloudWatch Synthetics dengan titik akhir VPC antarmuka.

Subnet pribadi adalah subnet tanpa rute bawaan ke internet publik. Men-deploy sumber daya AWS di subnet privat tidak mencegah Amazon CloudWatch mengumpulkan metrik bawaan dari sumber daya.

Jika Anda perlu menerbitkan metrik kustom dari sumber daya AWS di subnet privat, Anda dapat melakukannya menggunakan server proksi. Server proksi meneruskan permintaan HTTPS tersebut ke titik akhir API publik untuk CloudWatch.

Hub Keamanan AWS

Pantau penggunaan CloudWatch yang berkaitan dengan praktik terbaik keamanan dengan menggunakan Hub Keamanan AWS. Hub Keamanan menggunakan kontrol keamanan untuk mengevaluasi konfigurasi sumber daya dan standar keamanan guna membantu Anda mematuhi

berbagai kerangka kerja kepatuhan. Untuk informasi selengkapnya tentang menggunakan Hub Keamanan guna mengevaluasi sumber daya CloudWatch, silakan lihat kontrol Amazon CloudWatch di Panduan Pengguna Hub Keamanan AWS.

Menggunakan CloudWatch dan CloudWatch Synthetics dengan titik akhir VPC antarmuka

Jika Anda menggunakan Amazon Virtual Private Cloud (Amazon VPC) untuk menjadi host dari AWS sumber daya Anda, Anda dapat membuat koneksi pribadi antara VPC Anda, CloudWatch, dan CloudWatch Synthetics. Anda dapat menggunakan koneksi ini untuk mengaktifkan CloudWatch dan CloudWatch Synthetics untuk berkomunikasi dengan sumber daya Anda di VPC Anda tanpa melalui internet publik.

Amazon VPC adalah AWS layanan yang dapat Anda gunakan untuk meluncurkan AWS sumber daya dalam jaringan virtual yang Anda tetapkan. Dengan VPC, Anda memiliki kendali terhadap pengaturan jaringan, seperti rentang alamat IP, subnet, tabel rute, dan pintu masuk jaringan. Untuk menghubungkan VPC Anda ke CloudWatch atau CloudWatch Synthetics, Anda menentukan titik akhir VPC antarmuka untuk menghubungkan VPC ke AWS layanan. Titik akhir memberikan konektivitas yang dapat diandalkan dan diskalakan ke CloudWatch atau CloudWatch Synthetics tanpa memerlukan gateway internet, Network Address Translation (NAT), instans, atau koneksi VPN. Untuk informasi selengkapnya, silakan lihat Apa itu Amazon VPC dalam Panduan Pengguna Amazon VPC.

Titik akhir VPC antarmuka didukung oleh AWS PrivateLink sebuah AWS teknologi yang mendukung komunikasi privat antara AWS layanan dengan menggunakan antarmuka jaringan elastis dengan alamat IP privat. Untuk informasi selengkapnya, silakan lihat kiriman blog Baru – AWS PrivateLink untuk Layanan AWS.

Langkah-langkah berikut ditujukan untuk para pengguna Amazon VPC. Untuk informasi selengkapnya, silakan lihat Getting Started di Panduan Pengguna Amazon VPC.

Titik akhir VPC CloudWatch

CloudWatch saat ini mendukung titik akhir VPC di Wilayah AWS berikut:

- AS Timur (Ohio)
- AS Timur (Virginia Utara)
- AS Barat (California Utara)

- AS Barat (Oregon)
- Asia Pasifik (Hong Kong)
- Asia Pasifik (Mumbai)
- Asia Pasifik (Seoul)
- Asia Pasifik (Singapura)
- Asia Pasifik (Sydney)
- Asia Pasifik (Tokyo)
- Kanada (Pusat)
- Eropa (Frankfurt)
- Eropa (Irlandia)
- Eropa (London)
- Eropa (Paris)
- Timur Tengah (UEA)
- Amerika Selatan (Sao Paulo)
- AWS GovCloud (US-East)
- AWS GovCloud (US-West)

Membuat sebuah titik akhir VPC untuk CloudWatch

Untuk mulai menggunakan CloudWatch dengan VPC Anda, buat sebuah antarmuka titik akhir VPC untuk CloudWatch. Nama layanan untuk dipilih adalah `com.amazonaws.region.monitoring`. Untuk informasi selengkapnya, silakan lihat Membuat sebuah Titik Akhir Antarmuka dalam Panduan Pengguna Amazon VPC.

Anda tidak perlu mengubah pengaturan untuk CloudWatch. CloudWatch menghubungi layanan AWS lain menggunakan titik akhir publik atau titik akhir antarmuka VPC pribadi, mana pun yang digunakan. Sebagai contoh, jika Anda membuat titik akhir VPC antarmuka untuk CloudWatch, dan Anda sudah memiliki metrik yang mengalir ke CloudWatch dari sumber daya yang terletak di VPC, metrik ini mulai mengalir melalui titik akhir VPC antarmuka secara bawaan.

Mengontrol akses ke titik akhir VPC CloudWatch

Kebijakan titik akhir VPC 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, silakan lihat Mengendalikan Akses ke Layanan dengan titik akhir VPC dalam Panduan Pengguna Amazon VPC.

Berikut adalah sebuah contoh kebijakan titik akhir untuk CloudWatch. Kebijakan ini memungkinkan pengguna terhubung ke CloudWatch melalui VPC untuk mengirimkan data metrik ke CloudWatch dan mencegahnya dari melakukan tindakan CloudWatch lain.

```
{
  "Statement": [
    {
      "Sid": "PutOnly",
      "Principal": "*",
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Untuk menyunting kebijakan titik akhir VPC untuk CloudWatch

1. Buka konsol Amazon VPC di https://console.aws.amazon.com/vpc/.
2. Pada panel navigasi, silakan pilih Titik akhir.
3. Jika Anda belum membuat titik akhir untuk CloudWatch, pilih Buat Titik Akhir. Pilih com.amazonaws.***wilayah***.monitoring, lalu pilih Buat titik akhir.
4. Pilih titik akhir com.amazonaws.***region***.monitoring, titik akhir, dan kemudian pilih tab Kebijakan.
5. Pilih Edit Kebijakan, lalu lakukan perubahan.

Titik akhir VPC CloudWatch Synthetics

CloudWatch Synthetics saat ini mendukung titik akhir VPC di Wilayah AWS berikut:

- AS Timur (Ohio)
- AS Timur (Virginia Utara)
- AS Barat (California Utara)
- AS Barat (Oregon)
- Asia Pasifik (Hong Kong)
- Asia Pasifik (Mumbai)
- Asia Pasifik (Seoul)
- Asia Pasifik (Singapura)
- Asia Pasifik (Sydney)
- Asia Pasifik (Tokyo)
- Kanada (Pusat)
- Eropa (Frankfurt)
- Eropa (Irlandia)
- Eropa (London)
- Eropa (Paris)
- Amerika Selatan (Sao Paulo)

Membuat titik akhir VPC untuk CloudWatch Synthetics

Untuk mulai menggunakan CloudWatch Synthetics dengan VPC Anda, buat titik akhir VPC antarmuka untuk CloudWatch Synthetics. Nama layanan untuk dipilih adalah `com.amazonaws.region.synthetics`. Untuk informasi selengkapnya, silakan lihat Membuat sebuah Titik Akhir Antarmuka dalam Panduan Pengguna Amazon VPC.

Anda tidak perlu mengubah pengaturan untuk CloudWatch Synthetics. CloudWatch Synthetics berkomunikasi dengan layanan AWS lainnya menggunakan titik akhir publik atau titik akhir VPC antarmuka pribadi, mana saja yang digunakan. Sebagai contoh, jika Anda membuat titik akhir VPC antarmuka untuk CloudWatch Synthetics, dan Anda sudah memiliki titik akhir antarmuka untuk Amazon S3, CloudWatch Synthetics mulai berkomunikasi dengan Amazon S3 melalui titik akhir VPC antarmuka secara default.

Mengontrol akses ke titik akhir VPC CloudWatch Synthetics

Kebijakan titik akhir VPC 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, kami 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 mempengaruhi canary yang dikelola secara pribadi oleh VPC. Mereka tidak diperlukan untuk canary yang berjalan di subnet privat.

Kebijakan titik akhir harus ditulis dalam format JSON.

Untuk informasi selengkapnya, silakan lihat Mengendalikan Akses ke Layanan dengan titik akhir VPC dalam Panduan Pengguna Amazon VPC.

Berikut adalah sebuah contoh kebijakan titik akhir untuk CloudWatch Synthetics. Kebijakan ini memungkinkan pengguna terhubung ke CloudWatch Synthetics melalui VPC untuk melihat informasi tentang canary dan pelaksanaannya, tetapi tidak untuk membuat, memodifikasi, atau menghapus canary.

```
{
    "Statement": [
        {
            "Action": [
                "synthetics:DescribeCanaries",
                "synthetics:GetCanaryRuns"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        }
    ]
}
```

Untuk menyunting kebijakan titik akhir VPC untuk CloudWatch Synthetics

1. Buka konsol Amazon VPC di https://console.aws.amazon.com/vpc/.
2. Pada panel navigasi, silakan pilih Titik akhir.
3. Jika Anda belum membuat titik akhir untuk CloudWatch Synthetics, pilih Buat Titik Akhir. Pilih com.amazonaws.***region***.synthetics, lalu pilih Buat titik akhir.
4. Pilih titik akhir com.amazonaws.***region***.synthetics, dan kemudian pilih tab Kebijakan.
5. Pilih Edit Kebijakan, lalu lakukan perubahan.

Pertimbangan keamanan untuk canary Synthetics

Bagian berikut menjelaskan masalah keamanan yang harus Anda pertimbangkan ketika membuat dan menjalankan canary dalam Synthetics.

Gunakan Koneksi yang aman

Karena kode canary dan hasil dari test run canary dapat memuat informasi sensitif, jangan menghubungkan canary Anda ke titik akhir melalui koneksi yang tidak terenkripsi. Selalu gunakan koneksi yang terenkripsi, seperti yang dimulai dengan `https://`.

Pertimbangan penamaan canary

Amazon Resource Name (ARN) dari sebuah canary disertakan dalam header agen pengguna sebagai bagian dari panggilan keluar yang dibuat dari browser Chromium yang digerakkan oleh Puppeteer yang disertakan sebagai bagian dari pustaka pembungkus CloudWatch Synthetics. Hal ini membantu mengidentifikasi lalu lintas canary CloudWatch Synthetics dan mengaitkannya kembali dengan canary yang melakukan panggilan.

ARN canary mencakup nama canary. Pilih nama canary yang tidak mengungkapkan informasi hak milik.

Selain itu, pastikan untuk menunjuk canary Anda hanya di situs web dan titik akhir yang Anda kendalikan.

Rahasia dan informasi sensitif dalam kode canary

Jika Anda meneruskan kode canary langsung ke canary menggunakan file zip, isi dari skrip tersebut dapat dilihat di log AWS CloudTrail.

Jika Anda memiliki informasi atau rahasia sensitif (seperti kunci akses atau kredensial basis data) dalam skrip canary, kami sangat menyarankan Anda menyimpan skrip sebagai objek berversi di Amazon S3 dan meneruskan lokasi Amazon S3 ke dalam canary, alih-alih meneruskan kode canary dengan file zip.

Jika Anda menggunakan file zip untuk meneruskan skrip canary, kami sangat menyarankan agar Anda tidak menyertakan informasi rahasia atau informasi sensitif di kode sumber canary Anda. Untuk informasi selengkapnya tentang cara menggunakan AWS Secrets Manager untuk membantu menjaga keamanan rahasia Anda, silakan lihat Apa itu AWS Secrets Manager?.

Pertimbangan izin

Kami menyarankan Anda agar membatasi akses ke sumber daya yang dibuat atau digunakan oleh CloudWatch Synthetics. Gunakan izin ketat pada bucket Amazon S3 di mana canary menyimpan hasil test run dan artefak lainnya, seperti log dan tangkapan layar.

Demikian pula, simpan izin ketat pada lokasi tempat kode sumber canary Anda disimpan, sehingga tidak ada pengguna yang secara tidak sengaja atau jahat menghapus lapisan Lambda atau fungsi Lambda yang digunakan untuk canary tersebut.

Untuk membantu memastikan bahwa Anda menjalankan kode canary yang diinginkan, Anda dapat menggunakan versioning objek pada bucket Amazon S3 tempat kode canary disimpan. Kemudian, ketika Anda menentukan kode ini untuk dijalankan sebagai canary, Anda dapat menyertakan objek `versionId` sebagai bagian dari jalur, seperti dalam contoh berikut.

```
https://bucket.s3.amazonaws.com/path/object.zip?versionId=version-id
https://s3.amazonaws.com/bucket/path/object.zip?versionId=version-id
https://bucket.s3-region.amazonaws.com/path/object.zip?versionId=version-id
```

Jejak stack dan pesan pengecualian

Secara bawaan, canary CloudWatch Synthetics menangkap pengecualian apa pun yang dilempar oleh skrip canary Anda, tanpa memperhatikan apakah skrip ini kustom atau berasal dari cetak biru. CloudWatch Synthetics membuat log pesan pengecualian dan jejak stack ke tiga lokasi:

- Kembali ke layanan CloudWatch Synthetics untuk mempercepat proses debug ketika Anda menjelaskan test run
- Ke dalam Log CloudWatch sesuai dengan konfigurasi yang digunakan untuk membuat fungsi Lambda Anda
- Ke dalam file log Sintetis, yang merupakan file teks biasa yang diunggah ke lokasi Amazon S3 yang ditentukan oleh nilai yang ditetapkan untuk `resultsLocation` dari canary tersebut

Jika Anda ingin mengirim dan menyimpan lebih sedikit informasi, Anda dapat memperoleh pengecualian sebelum kembali ke pustaka pembungkus CloudWatch Synthetics.

Anda juga dapat memiliki URL permintaan di kesalahan Anda. CloudWatch Synthetics memindai URL apa pun dalam kesalahan yang dilemparkan oleh skrip Anda dan menyunting parameter URL yang dibatasi darinya berdasarkan konfigurasi restrictedUrlParameters. Jika Anda membuat log

kesalahan dalam skrip Anda, Anda dapat menggunakan getSanitizedErrorPesan untuk menyunting URL sebelum membuat log.

Batasi peran IAM Anda secara sempit

Kami menyarankan agar Anda tidak mengonfigurasi canary untuk mengunjungi titik akhir URL yang berpotensi berbahaya. Menunjukkan canary Anda ke situs web atau titik akhir yang tidak dipercaya atau tidak dikenal dapat membuat kode fungsi Lambda Anda mengekspos skrip pengguna yang berbahaya. Dengan asumsi bahwa situs web berbahaya dapat keluar dari Chromium, situs web tersebut dapat memiliki akses ke kode Lambda Anda dengan cara yang sama jika Anda terhubung dengan menggunakan browser internet.

Jalankan fungsi Lambda Anda dengan peran eksekusi IAM yang memiliki cakupan izin. Dengan cara ini, jika fungsi Lambda Anda dikompromikan oleh skrip berbahaya, tindakan terbatas yang dapat diambil ketika menjalankan akun AWS canary Anda.

Ketika Anda menggunakan konsol CloudWatch untuk membuat canary, itu dibuat dengan peran eksekusi IAM yang dibatasi.

Redaksi data sensitif

CloudWatch Synthetics menangkap URL, kode status, alasan kegagalan (jika ada), dan header serta isi permintaan dan respons. Ini memungkinkan pengguna canary untuk memahami, memantau, dan men-debug canary.

Konfigurasi yang dijelaskan di bagian berikut dapat diatur kapan saja dalam eksekusi canary. Anda juga dapat memilih untuk menerapkan konfigurasi yang berbeda untuk langkah-langkah synthetics yang berbeda.

URL Permintaan

Secara default, CloudWatch Synthetics log meminta URL, kode status, dan alasan status untuk setiap URL di log canary. URL permintaan juga dapat muncul dalam laporan eksekusi canary, file HAR, dan sebagainya. URL permintaan Anda mungkin berisi parameter kueri sensitif, seperti token akses atau kata sandi. Anda dapat menyunting informasi sensitif agar tidak dibuat log oleh CloudWatch Synthetics.

Untuk menyunting informasi sensitif, atur properti konfigurasi restrictedUrlParameters. Untuk informasi selengkapnya, lihat SyntheticsConfiguration kelas. Hal ini menyebabkan CloudWatch

Synthetics menyunting parameter URL, termasuk nilai parameter path dan query, berdasarkan restrictedUrlParameters sebelum membuat log. Jika Anda membuat log URL di skrip Anda, Anda dapat menggunakan getSanitizedUrl(url, StepConfig = nol) untuk menyunting URL sebelum membuat log. Untuk informasi selengkapnya, lihat SyntheticsLogHelper kelas.

Header

Secara default, CloudWatch Synthetics tidak membuat log header permintaan/respons. Untuk canary UI, ini adalah perilaku default untuk canary yang menggunakan versi `syn-nodejs-puppeteer-3.2` runtime dan yang lebih baru.

Jika header Anda tidak berisi informasi sensitif, Anda dapat mengaktifkan header dalam file HAR dan laporan HTTP dengan menyetel includeRequestHeaders dan includeResponseHeaders ke `true`. Anda dapat mengaktifkan semua header tetapi pilihlah untuk membatasi nilai tombol header sensitif. Misalnya, Anda dapat memilih untuk hanya menyunting header `Authorization` dari artefak yang dihasilkan oleh canary.

Permintaan dan isi respons

Secara default, CloudWatch Synthetics tidak membuat log isi permintaan/respons di log atau laporan canary. Informasi ini sangat berguna untuk canary API. Synthetics menangkap semua permintaan HTTP dan dapat menampilkan header, permintaan, dan isi respons. Untuk informasi selengkapnya, lihat executeHttpStep(StepName, requestOptions, [callback], [stepConfig]). Anda dapat memilih untuk mengaktifkan isi permintaan/respons dengan menyetel properti includeRequestBody dan includeResponseBody ke `true`.

Mencatat panggilan CloudWatch API Amazon dengan AWS CloudTrail

Amazon CloudWatch dan CloudWatch Synthetics terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau layanan. AWS CloudTrail menangkap panggilan API yang dilakukan oleh atau atas nama AWS akun Anda. Panggilan yang direkam tersebut mencakup panggilan dari konsol dan panggilan kode ke operasi API.

Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara berkelanjutan ke bucket S3, termasuk acara untuk CloudWatch. Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru di CloudTrail konsol dalam Riwayat acara. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat CloudWatch, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail lainnya.

Untuk mempelajari selengkapnya CloudTrail, termasuk cara mengonfigurasi dan mengaktifkannya, lihat Panduan AWS CloudTrail Pengguna.

Setiap peristiwa atau entri log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan berikut:

- Apakah permintaan itu dibuat dengan kredenal pengguna root atau AWS Identity and Access Management (IAM).
- Apakah permintaan dibuat dengan kredensial keamanan sementara untuk suatu peran atau pengguna gabungan.
- Apakah permintaan itu dibuat oleh AWS layanan lain.

Untuk informasi lain, lihat Elemen userIdentity CloudTrail .

Untuk catatan peristiwa yang sedang berlangsung di AWS akun Anda, termasuk acara untuk CloudWatch dan CloudWatch Synthetics, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket S3. Secara default, saat Anda membuat jejak di konsol, jejak tersebut berlaku untuk semua AWS Wilayah. Jejak mencatat peristiwa dari semua Wilayah di AWS partisi dan mengirimkan file log ke bucket S3 yang Anda tentukan. Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan bertindak atas data peristiwa yang dikumpulkan dalam CloudTrail log. Untuk informasi selengkapnya, lihat berikut:

- Gambaran umum untuk Membuat Jejak
- CloudTrail Layanan dan Integrasi yang Didukung
- Mengonfigurasi Notifikasi Amazon SNS untuk CloudTrail
- Menerima File CloudTrail Log dari Beberapa Wilayah dan Menerima File CloudTrail Log dari Beberapa Akun

> **Note**
>
> Untuk informasi tentang panggilan API CloudWatch Log yang masuk CloudTrail, lihat Informasi CloudWatch log masuk CloudTrail.

Topik

- CloudWatch informasi di CloudTrail
- CloudWatch Monitor Internet di CloudTrail
- CloudWatch Informasi Synthetics di CloudTrail

CloudWatch informasi di CloudTrail

CloudWatch mendukung pencatatan tindakan berikut sebagai peristiwa dalam file CloudTrail log:

- DeleteAlarms
- DeleteAnomalyDetector
- DeleteDashboards
- DescribeAlarmHistory
- DescribeAlarms
- DescribeAlarmsForMetric
- DescribeAnomalyDetectors
- DisableAlarmActions
- EnableAlarmActions
- GetDashboard
- ListDashboards

- PutAnomalyDetector
- PutDashboard
- PutMetricAlarm
- SetAlarmState

Contoh: entri file CloudWatch log

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan `PutMetricAlarm` tindakan.

```
{
    "Records": [{
        "eventVersion": "1.01",
        "userIdentity": {
            "type": "Root",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:root",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID"
        },
        "eventTime": "2014-03-23T21:50:34Z",
        "eventSource": "monitoring.amazonaws.com",
        "eventName": "PutMetricAlarm",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "127.0.0.1",
        "userAgent": "aws-sdk-ruby2/2.0.0.rc4 ruby/1.9.3 x86_64-linux Seahorse/0.1.0",
        "requestParameters": {
            "threshold": 50.0,
            "period": 60,
            "metricName": "CloudTrail Test",
            "evaluationPeriods": 3,
            "comparisonOperator": "GreaterThanThreshold",
            "namespace": "AWS/CloudWatch",
            "alarmName": "CloudTrail Test Alarm",
            "statistic": "Sum"
        },
        "responseElements": null,
        "requestID": "29184022-b2d5-11e3-a63d-9b463e6d0ff0",
        "eventID": "b096d5b7-dcf2-4399-998b-5a53eca76a27"
    },
    ..additional entries
  ]
```

```
  }
```

Entri file log berikut menunjukkan bahwa pengguna disebut `PutRule` tindakan CloudWatch Peristiwa.

```
{
        "eventVersion":"1.03",
        "userIdentity":{
           "type":"Root",
           "principalId":"123456789012",
           "arn":"arn:aws:iam::123456789012:root",
           "accountId":"123456789012",
           "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
           "sessionContext":{
              "attributes":{
                 "mfaAuthenticated":"false",
                 "creationDate":"2015-11-17T23:56:15Z"
              }
           }
        },
        "eventTime":"2015-11-18T00:11:28Z",
        "eventSource":"events.amazonaws.com",
        "eventName":"PutRule",
        "awsRegion":"us-east-1",
        "sourceIPAddress":"AWS Internal",
        "userAgent":"AWS CloudWatch Console",
        "requestParameters":{
           "description":"",
           "name":"cttest2",
           "state":"ENABLED",
           "eventPattern":"{\"source\":[\"aws.ec2\"],\"detail-type\":[\"EC2 Instance
 State-change Notification\"]}",
           "scheduleExpression":""
        },
        "responseElements":{
           "ruleArn":"arn:aws:events:us-east-1:123456789012:rule/cttest2"
        },
        "requestID":"e9caf887-8d88-11e5-a331-3332aa445952",
        "eventID":"49d14f36-6450-44a5-a501-b0fdcdfaeb98",
        "eventType":"AwsApiCall",
        "apiVersion":"2015-10-07",
        "recipientAccountId":"123456789012"
}
```

Entri file log berikut menunjukkan bahwa pengguna bernama `CreateExportTask` tindakan CloudWatch Log.


```
{
        "eventVersion": "1.03",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/someuser",
            "accountId": "123456789012",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "userName": "someuser"
        },
        "eventTime": "2016-02-08T06:35:14Z",
        "eventSource": "logs.amazonaws.com",
        "eventName": "CreateExportTask",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "127.0.0.1",
        "userAgent": "aws-sdk-ruby2/2.0.0.rc4 ruby/1.9.3 x86_64-linux Seahorse/0.1.0",
        "requestParameters": {
            "destination": "yourdestination",
            "logGroupName": "yourloggroup",
            "to": 123456789012,
            "from": 0,
            "taskName": "yourtask"
        },
        "responseElements": {
            "taskId": "15e5e534-9548-44ab-a221-64d9d2b27b9b"
        },
        "requestID": "1cd74c1c-ce2e-12e6-99a9-8dbb26bd06c9",
        "eventID": "fd072859-bd7c-4865-9e76-8e364e89307c",
        "eventType": "AwsApiCall",
        "apiVersion": "20140328",
        "recipientAccountId": "123456789012"
}
```


CloudWatch Monitor Internet di CloudTrail

CloudWatch Internet Monitor mendukung pencatatan tindakan berikut sebagai peristiwa dalam file CloudTrail log.

- CreateMonitor

- DeleteMonitor
- GetHealthEvent
- GetMonitor
- GetQueryResults
- GetQueryStatus
- ListHealthEvents
- ListMonitors
- ListTagsForResource
- StartQuery
- StopQuery
- UpdateMonitor

Contoh: entri file log CloudWatch Internet Monitor

Contoh berikut menunjukkan entri log CloudTrail Internet Monitor yang menunjukkan `ListMonitors` tindakan.


```
{
        "eventVersion": "1.08",
        "userIdentity": {
            "type": "AssumedRole",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::000000000000:assumed-role/role_name",
            "accountId":"123456789012",
            "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
            "sessionContext": {
                "sessionIssuer": {
                "type": "Role",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::000000000000:role/Admin",
                "accountId":"123456789012",
                "userName": "SAMPLE_NAME"
                },
                "webIdFederationData": {},
                "attributes": {
                    "creationDate": "2022-10-11T17:25:41Z",
                    "mfaAuthenticated": "false"
                }
```

```
            }
        },
        "eventTime": "2022-10-11T17:30:18Z",
        "eventSource": "internetmonitor.amazonaws.com",
        "eventName": "ListMonitors",
        "awsRegion": "us-east-2",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)",
        "requestParameters": null,
        "responseElements": null,
        "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
        "readOnly": true,
        "eventType": "AwsApiCall",
        "managementEvent": true,
        "recipientAccountId": "111122223333",
        "eventCategory": "Management"
    }
```

Contoh berikut menunjukkan entri log CloudTrail Internet Monitor yang menunjukkan `CreateMonitor` tindakan.

```
{
        "eventVersion": "1.08",
        "userIdentity": {
            "type": "AssumedRole",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::000000000000:assumed-role/role_name",
            "accountId":"123456789012",
            "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
            "sessionContext": {
                "sessionIssuer": {
                "type": "Role",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::000000000000:role/Admin",
                "accountId":"123456789012",
                "userName": "SAMPLE_NAME"
                },
                "webIdFederationData": {},
                "attributes": {
                    "creationDate": "2022-10-11T17:25:41Z",
                    "mfaAuthenticated": "false"
                }
```

```
            }
        },
        "eventTime": "2022-10-11T17:30:08Z",
        "eventSource": "internetmonitor.amazonaws.com",
        "eventName": "CreateMonitor",
        "awsRegion": "us-east-2",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)",
        "requestParameters": {
            "MonitorName": "TestMonitor",
            "Resources": ["arn:aws:ec2:us-east-2:444455556666:vpc/vpc-febc0b95"],
            "ClientToken": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333"
        },
        "responseElements": {
            "Arn": "arn:aws:internetmonitor:us-east-2:444455556666:monitor/ct-
onboarding-test",
            "Status": "PENDING"
        },
        "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
        "readOnly": false,
        "eventType": "AwsApiCall",
        "managementEvent": true,
        "recipientAccountId": "111122223333",
        "eventCategory": "Management"
    }
```

CloudWatch Informasi Synthetics di CloudTrail

CloudWatch Synthetics mendukung pencatatan tindakan berikut sebagai peristiwa dalam file CloudTrail log:

- CreateCanary
- DeleteCanary
- DescribeCanaries
- DescribeCanariesLastRun
- DescribeRuntimeVersions
- GetCanary
- GetCanaryRuns

- ListTagsForResource
- StartCanary
- StopCanary
- TagResource
- UntagResource
- UpdateCanary

Contoh: Entri CloudWatch file log Synthetics

Contoh berikut menunjukkan entri log CloudTrail Synthetics yang menunjukkan tindakan. `DescribeCanaries`


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:assumed-role/role_name",
        "accountId":"123456789012",
        "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::111222333444:role/Administrator",
                "accountId":"123456789012",
                "userName": "SAMPLE_NAME"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-04-08T21:43:24Z"
            }
        }
    },
    "eventTime": "2020-04-08T23:06:47Z",
    "eventSource": "synthetics.amazonaws.com",
    "eventName": "DescribeCanaries",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
```

```
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.590
 Linux/4.9.184-0.1.ac.235.83.329.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.212-b03
 java/1.8.0_212 vendor/Oracle_Corporation",
    "requestParameters": null,
    "responseElements": null,
    "requestID": "201ed5f3-15db-4f87-94a4-123456789",
    "eventID": "73ddbd81-3dd0-4ada-b246-123456789",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

Contoh berikut menunjukkan entri log CloudTrail Synthetics yang menunjukkan tindakan. `UpdateCanary`

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:assumed-role/role_name",
        "accountId":"123456789012",
        "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
               "type": "Role",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::111222333444:role/Administrator",
       "accountId":"123456789012",
                "userName": "SAMPLE_NAME"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-04-08T21:43:24Z"
            }
        }
    },
    "eventTime": "2020-04-08T23:06:47Z",
    "eventSource": "synthetics.amazonaws.com",
    "eventName": "UpdateCanary",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
```

```
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.590
 Linux/4.9.184-0.1.ac.235.83.329.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.212-b03
 java/1.8.0_212 vendor/Oracle_Corporation",
    "requestParameters": {
        "Schedule": {
            "Expression": "rate(1 minute)"
        },
        "name": "sample_canary_name",
        "Code": {
            "Handler": "myOwnScript.handler",
            "ZipFile": "SAMPLE_ZIP_FILE"
        }
    },
    "responseElements": null,
    "requestID": "fe4759b0-0849-4e0e-be71-1234567890",
    "eventID": "9dc60c83-c3c8-4fa5-bd02-1234567890",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

Contoh berikut menunjukkan entri log CloudTrail Synthetics yang menunjukkan tindakan. `GetCanaryRuns`

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:assumed-role/role_name",
        "accountId":"123456789012",
        "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::111222333444:role/Administrator",
       "accountId":"123456789012",
                "userName": "SAMPLE_NAME"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
```

```
                "creationDate": "2020-04-08T21:43:24Z"
            }
        }
    },
    "eventTime": "2020-04-08T23:06:30Z",
    "eventSource": "synthetics.amazonaws.com",
    "eventName": "GetCanaryRuns",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.590
 Linux/4.9.184-0.1.ac.235.83.329.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.212-b03
 java/1.8.0_212 vendor/Oracle_Corporation",
    "requestParameters": {
        "Filter": "TIME_RANGE",
        "name": "sample_canary_name",
        "FilterValues": [
            "2020-04-08T23:00:00.000Z",
            "2020-04-08T23:10:00.000Z"
        ]
    },
    "responseElements": null,
    "requestID": "2f56318c-cfbd-4b60-9d93-1234567890",
    "eventID": "52723fd9-4a54-478c-ac55-1234567890",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

Pemberian tag pada sumber daya Amazon CloudWatch Anda

Tag adalah label atribut khusus yang Anda atau AWS tetapkan ke sumber daya AWS. Setiap tag memiliki dua bagian:

- Sebuah kunci tag (misalnya, `CostCenter`, `Environment`, atau `Project`). Kunci tag peka huruf besar dan kecil.
- Bidang opsional yang dikenal sebagai nilai tag (misalnya, `111122223333` atau `Production`). Mengabaikan nilai tag sama dengan menggunakan sebuah string kosong. Seperti kunci tag, nilai tag peka huruf besar dan kecil.

Tag membantu Anda melakukan hal berikut:

- Mengidentifikasi dan mengorganisasi sumber daya AWS Anda. Banyak layanan AWS yang mendukung pemberian tag, sehingga Anda dapat menetapkan tag yang sama ke sumber daya dari layanan yang berbeda untuk menunjukkan bahwa sumber daya tersebut terkait. Contohnya, Anda dapat menetapkan tag yang sama ke aturan CloudWatch yang Anda tetapkan ke sebuah instans EC2.

Bagian berikut menyediakan informasi selengkapnya tentang tag untuk CloudWatch.

Sumber daya yang didukung di CloudWatch

Sumber daya berikut dalam pemberian tag dukungan CloudWatch:

- Alarm – Anda dapat memberi tag alarm dengan menggunakan perintah AWS CLI tag–resource dan API TagResource. Anda juga dapat melihat dan mengelola tag alarm menggunakan halaman rincian Alarm di konsol CloudWatch.
- Canary – Anda dapat memberi tag canary menggunakan konsol CloudWatch. Untuk informasi selengkapnya, silakan lihat Membuat canary.
- Aturan Wawasan Kontributor – Anda dapat menandai aturan Wawasan Kontributor ketika membuatnya dengan menggunakan perintah AWS CLI put–insight–rule, dan API PutInsightRule. Anda dapat menambahkan tag ke aturan yang ada dengan menggunakan perintah AWS CLI tag-resource, dan API TagResource.

- Aliran metrik – Anda dapat memberi tag aliran metrik saat membuatnya menggunakan perintah AWS CLI put–metric–stream dan API PutMetricStream. Anda dapat menambahkan tanda ke aliran metrik dengan menggunakan perintah AWS CLI tag-resource, dan API TagResource.

Untuk informasi tentang menambahkan dan mengelola tag, silakan lihat Mengelola tag.

Mengelola tag

Tag terdiri atas properti `Key` dan `Value` pada sumber daya. Anda dapat menggunakan konsol CloudWatch, AWS CLI, atau API CloudWatch untuk menambahkan, mengedit, atau menghapus nilai untuk properti ini. Untuk informasi tentang bekerja dengan tag, silakan lihat hal berikut:

- TagResource, UntagResource, dan ListTagsForResource di Referensi API Amazon CloudWatch
- tag-resource, untag-resource, dan list-tags-for-resource di Referensi CLI Amazon CloudWatch
- Bekerja dengan Editor Tag di Panduan Pengguna Grup Sumber Daya

Kesepakatan penamaan dan penggunaan tag

Kesepakatan penamaan dan penggunaan dasar berikut berlaku untuk menggunakan tag dengan sumber daya CloudWatch:

- Setiap sumber daya dapat memiliki maksimum 50 tag.
- Untuk setiap sumber daya, setiap kunci tag harus unik, dan setiap kunci tag hanya dapat memiliki satu nilai.
- Panjang kunci tag maksimum adalah 128 karakter Unicode dalam UTF-8.
- Panjang nilai tag maksimum adalah 256 karakter Unicode dalam UTF-8.
- Karakter yang diperbolehkan adalah huruf, angka, spasi yang dapat ditampilkan di UTF-8, serta karakter berikut: . : + = @ _ / - (tanda hubung).
- Kunci dan nilai tag peka huruf besar dan kecil. Sebagai praktik terbaik, tentukan strategi untuk menulis tag dalam huruf kapital dan secara konsisten menerapkan strategi tersebut di semua jenis sumber daya. Misalnya, putuskan apakah akan menggunakan `Costcenter`, `costcenter`, atau `CostCenter` dan menggunakan kesepakatan yang sama untuk semua tag. Hindari penggunaan tag yang serupa dengan perlakuan kasus yang tidak konsisten.

- Awalan `aws:` dilarang untuk tag karena diperuntukkan untuk penggunaan AWS. Anda tidak dapat menyunting atau menghapus kunci atau nilai tag dengan awalan ini. Tag dengan awalan ini tidak dihitung terhadap tag Anda per batas sumber daya.

Integrasi Grafana

Anda dapat menggunakan Grafana versi 6.5.0 dan versi yang lebih baru untuk memajukan secara kontekstual melalui konsol CloudWatch dan pencarian daftar metrik dinamis dengan menggunakan karakter pengganti. Hal ini akan dapat membantu Anda memantau metrik untuk sumber daya AWS, seperti instans atau kontainer Amazon Elastic Compute Cloud. Ketika ada instans baru yang dibuat sebagai bagian dari peristiwa Penskalaan Otomatis, instans baru tersebut akan muncul di grafik secara otomatis. Anda tidak perlu melacak ID instans baru tersebut. Dasbor yang telah dibuat sebelumnya akan membantu Anda menyederhanakan pengalaman dalam memulai pemantauan Amazon EC2, Amazon Elastic Block Store, dan sumber daya AWS Lambda.

Anda dapat menggunakan Grafana versi 7.0 dan versi yang lebih baru untuk menjalankan kueri Wawasan Log CloudWatch pada grup log yang ada di Log CloudWatch. Anda dapat membuat visualisasi dari hasil pencarian Anda dalam bentuk grafik batang, baris, dan grafik bertumpuk, serta dalam format tabel. Untuk informasi selengkapnya mengenai Wawasan Log CloudWatch, silakan lihat Menganalisis Data Log dengan Wawasan Log CloudWatch.

Untuk informasi selengkapnya tentang cara memulainya, silakan lihat Menggunakan AWS CloudWatch di Grafana dalam dokumentasi Lab Grafana.

Konsol CloudWatch lintas akun lintas Wilayah

Untuk mendapatkan pengalaman observabilitas dan penemuan lintas akun paling kaya untuk metrik, log, dan jejak Anda, sebaiknya Anda menggunakan observabilitas lintas akun CloudWatch. Untuk informasi selengkapnya, lihat CloudWatch observabilitas lintas akun.

CloudWatch juga menawarkan sebuah dasbor CloudWatch lintas-akun, lintas-Wilayah. Fungsi ini akan memberikan Anda visibilitas lintas akun ke dasbor, alarm, metrik, dan dasbor otomatis. CloudWatch tidak memberikan visibilitas lintas akun untuk log atau untuk jejak.

Jika Anda juga menggunakan observabilitas lintas akun CloudWatch, satu kasus penggunaan untuk dasbor CloudWatch lintas akun ini adalah membiarkan salah satu akun sumber observabilitas lintas akun CloudWatch Anda melihat metrik yang berasal dari akun sumber lain.

Sisa bagian ini akan memberikan penjelasan tentang dasbor lintas akun dan lintas Wilayah. Anda dapat menggunakannya untuk membuat dasbor yang akan meringkas data CloudWatch dari berbagai akun AWS dan berbagai Wilayah AWS ke dalam satu dasbor. Anda juga dapat membuat sebuah alarm di satu akun yang mengawasi sebuah metrik yang terletak di akun yang berbeda.

Banyak organisasi menerapkan sumber daya AWS mereka dalam beberapa akun, untuk menyediakan batas penagihan dan keamanan. Dalam hal ini, kami menyarankan agar Anda untuk menunjuk satu atau beberapa akun sebagai akun pemantauan Anda, dan membuat dasbor lintas akun Anda di akun ini.

Fungsi lintas akun terintegrasi dengan AWS Organizations, hal ini akan membantu Anda dalam membuat dasbor lintas akun secara efisien.

Fungsionalitas Lintas Wilayah

Fungsionalitas Lintas Wilayah sekarang sudah terpasang secara otomatis. Anda tidak perlu melakukan langkah tambahan apa pun untuk dapat menampilkan metrik-metrik dari Wilayah yang berbeda di satu akun pada grafik yang sama atau dasbor yang sama. Fungsionalitas Lintas Wilayah ini tidak didukung untuk alarm, sehingga Anda tidak dapat membuat sebuah alarm di satu Wilayah yang akan mengawasi sebuah metrik di Wilayah lain yang berbeda.

Topik

- Melakukan penonaktifan dan pembersihan setelah menggunakan lintas akun

Mengaktifkan fungsionalitas lintas akun di CloudWatch

Untuk mengatur fungsionalitas lintas akun di konsol CloudWatch, Anda hris menggunakan konsol CloudWatch tersebut untuk menyiapkan akun berbagi dan akun pemantauan Anda.

Menyiapkan akun berbagi

Anda harus mengaktifkan berbagi di masing-masing akun yang akan menyediakan data ke akun pemantauan.

Langkah ini akan memberikan izin baca-saja yang Anda pilih pada langkah 5 untuk semua pengguna yang melihat dasbor lintas akun di akun yang Anda bagikan, jika pengguna memiliki izin yang sesuai dengan akun yang Anda bagikan.

Untuk memungkinkan akun Anda membagikan data CloudWatch dengan akun-akun lain

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Pengaturan.
3. Untuk Bagikan data CloudWatch Anda, pilih Konfigurasikan.
4. Untuk Berbagi, pilih Akun-akun tertentu dan kemudian masukkan ID dari akun yang Anda inginkan untuk berbagi data dengannya.

   Semua akun yang Anda tentukan di sini akan dapat melihat data CloudWatch yang dimiliki akun Anda. Masukkan ID akun hanya dari akun-akun yang Anda kenal dan percaya.
5. Untuk Izin, tentukan cara berbagi data Anda dengan salah satu pilihan berikut:

   - Berikan akses baca-saja ke metrik, dasbor, dan alarm CloudWatch Anda. Pilihan ini akan memungkinkan akun pemantauan untuk membuat dasbor lintas akun yang mempunyai widget yang berisi data CloudWatch dari akun Anda.
   - Sertakan dasbor otomatis CloudWatch. Jika Anda memilih pilihan ini, maka pengguna yang ada di akun pemantauan juga akan dapat melihat informasi di dasbor otomatis akun ini. Untuk informasi selengkapnya, lihat Memulai dengan Amazon CloudWatch.
   - Sertakan akses baca-saja X-Ray untuk X-Ray Trace Map. Jika Anda memilih pilihan ini, maka pengguna yang ada di akun pemantauan juga akan dapat melihat peta jejak X-Ray dan informasi jejak X-Ray di akun ini. Untuk informasi selengkapnya, silakan lihat Menggunakan X-Ray Trace Map.

- Akses baca-saja penuh ke semua yang ada di akun Anda. Pilihan ini akan memungkinkan akun yang Anda gunakan untuk berbagi dapat membuat dasbor lintas akun yang mempunyai widget yang berisi data CloudWatch dari akun Anda. Pilihan ini juga akan memungkinkan akun-akun tersebut untuk melihat lebih dalam ke akun Anda dan melihat data akun Anda di dalam konsol layanan AWS lainnya.

6. Pilih Luncurkan template CloudFormation.

   Di layar konfirmasi, ketik **`Confirm`**, dan pilih Template peluncuran.

7. Pilih kotak centang Saya mengakui... , dan pilih Buat tumpukan.

Berbagi dengan seluruh organisasi

Bila Anda telah menyelesaikan prosedur-prosedur sebelumnya, hal itu akan menciptakan sebuah peran IAM sehingga memungkinkan akun Anda untuk berbagi data dengan satu akun. Anda dapat membuat atau menyunting sebuah peran IAM yang membagikan data Anda dengan semua akun dalam sebuah organisasi. Lakukan ini hanya jika Anda mengenal dan memercayai semua akun yang ada dalam organisasi tersebut.

Langkah ini akan memberikan izin baca-saja yang tercantum di kebijakan yang ditunjukkan di langkah 5 dalam prosedur sebelumnya kepada semua pengguna yang melihat dasbor lintas akun di akun yang Anda bagikan, jika pengguna memiliki izin yang sesuai dengan akun yang Anda bagikan.

Untuk berbagi data akun CloudWatch Anda dengan semua akun yang ada di dalam sebuah organisasi

1. Jika Anda belum melakukannya sama sekali, maka Anda harus menyelesaikan prosedur untuk membagikan data Anda dengan satu akun AWS yang telah dijelaskan sebelumnya.
2. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
3. Pada panel navigasi, silakan pilih Peran.
4. Dalam daftar peran, pilih CloudWatch-CrossAccountSharingRole.
5. Pilih Hubungan kepercayaan, Sunting hubungan kepercayaan.

   Anda akan melihat kebijakan seperti ini:

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:root"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

6. Ubah kebijakan tersebut sehingga menjadi seperti berikut ini, yang mengganti *org-id* dengan ID dari organisasi Anda.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalOrgID": "org-id"
        }
      }

    }
  ]
}
```

7. Pilih Perbarui Kebijakan Kepercayaan.

Menyiapkan sebuah akun pemantauan

Aktifkan setiap akun pemantauan jika Anda ingin menampilkan data lintas akun CloudWatch.

Setelah Anda menyelesaikan prosedur berikut ini, CloudWatch akan membuat sebuah peran tertaut layanan yang digunakan CloudWatch di akun pemantauan untuk mengakses data yang dibagikan dari akun Anda yang lain. Peran tertaut layanan ini disebut

AWSServiceRoleForCloudWatchCrossAccount. Untuk informasi selengkapnya, silakan lihat Menggunakan peran terkait layanan untuk CloudWatch.

Cara memungkinkan akun Anda untuk melihat data lintas akun CloudWatch

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Pada panel navigasi, silakan pilih Pengaturan, dan kemudian, di bagian lintas akun lintas wilayah, silakan pilih Konfigurasikan.
3. Pada bagian Tampilkan lintas akun lintas wilayah, pilih Aktifkan, dan kemudian pilih kotak centang Tampilkan pemilih di konsol untuk mengaktifkan pemilih akun agar ditampilkan di konsol CloudWatch saat Anda membuat grafik dari sebuh metrik atau membuat sebuah alarm.
4. Pada Tampilkan lintas akun lintas wilayah, pilih salah satu pilihan berikut:

   - Masukan Id Akun. Pilihan ini akan meminta Anda untuk memasukkan ID akun secara manual setiap kali Anda ingin beralih akun ketika Anda menampilkan data lintas akun.
   - Pemilih akun Organisasi AWS. Pilihan ini akan menyebabkan akun yang Anda tetapkan ketika menyelesaikan integrasi lintas akun dengan Organisasi dimunculkan. Saat berikutnya Anda menggunakan konsol, CloudWatch akan menampilkan daftar geser-turun dari akun-akun ini yang bisa Anda pilih ketika Anda menampilkan data lintas akun.

     Untuk melakukan hal ini, Anda harus terlebih dahulu menggunakan akun manajemen organisasi Anda untuk mengizinkan CloudWatch melihat daftar akun yang ada dalam organisasi Anda. Untuk informasi selengkapnya, lihat (Opsional) Integrasikan dengan AWS Organizations.
   - Pemilih akun kustom. Pilihan ini akan meminta Anda untuk memasukkan daftar ID akun. Saat berikutnya Anda menggunakan konsol, CloudWatch akan menampilkan daftar geser-turun dari akun-akun ini yang bisa Anda pilih ketika Anda menampilkan data lintas akun.

     Anda juga dapat memasukkan label untuk masing-masing akun ini sehingga dapat membantu dalam mengidentifikasi akun-akun tersebut ketika memilih akun yang akan ditampilkan.

     Pengaturan pemilih akun yang dibuat oleh seorang pengguna di sini hanya akan dipertahankan untuk pengguna tersebut, bukan untuk semua pengguna lain yang ada dalam akun pemantauan.
5. Pilih Aktifkan.

Setelah menyelesaikan pengaturan ini, Anda akan dapat membuat dasbor lintas akun. Untuk informasi selengkapnya, lihat Dasbor lintas akun lintas Wilayah.

(Opsional) Integrasikan dengan AWS Organizations

Jika Anda ingin mengintegrasikan fungsionalitas lintas akun dengan AWS Organizations, maka Anda harus membuat daftar dari semua akun yang ada dalam organisasi yang tersedia untuk akun pemantauan.

Untuk memungkinkan fungsionalitas lintas akun CloudWatch mengakses daftar dari semua akun yang ada dalam organisasi Anda

1. Masuk ke akun manajemen organisasi Anda.
2. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
3. Pada panel navigasi, silakan pilih Pengaturan, lalu pilih Konfigurasikan.
4. Untuk Memberikan izin untuk melihat daftar akun yang ada dalam organisasi, pilih Akun tertentu untuk diminta memasukkan daftar ID akun. Daftar akun-akun yang ada dalam organisasi Anda tersebut hanya akan dibagikan dengan akun-akun yang Anda tentukan di sini.
5. Pilih Bagikan daftar akun organisasi.
6. Pilih Luncurkan template CloudFormation.

   Di layar konfirmasi, ketik **`Confirm`**, dan pilih Template peluncuran.

Menyelesaikan masalah-masalah yang terjadi saat penyiapan lintas akun CloudWatch

Bagian ini memuat tips pemecahan masalah untuk deployment konsol lintas akun di CloudWatch.

Saya mendapatkan kesalahan akses ditolak yang menampilkan data lintas akun

Periksa hal-hal berikut:

- Akun pemantauan Anda harus memiliki sebuah peran yang bernama AWSServiceRoleForCloudWatchCrossAccount. Jika tidak, maka Anda harus membuat peran ini terlebih dahulu. Untuk informasi selengkapnya, lihat Set Up a Monitoring Account.

- Setiap akun berbagi harus memiliki sebuah peran yang bernama CloudWatch-CrossAccountSharingRole. Jika tidak, maka Anda harus membuat peran ini terlebih dahulu. Untuk informasi selengkapnya, lihat Set Up A Sharing Account.
- Peran berbagi tersebut harus mempercayai akun pemantauan.

Cara mengonfirmasi apakah peran Anda telah diatur dengan tepat untuk konsol CloudWatch lintas akun

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi, silakan pilih Peran.
3. Dalam daftar peran, pastikan peran yang diperlukan sudah ada. Di akun berbagi, cari CloudWatch-CrossAccountSharingRole. Di akun pemantauan, cari AWSServiceRoleForCloudWatchCrossAccount.
4. Jika Anda berada di sebuah akun bersama dan CloudWatch-CrossAccountSharingRole sudah ada, pilih CloudWatch-CrossAccountSharingRole.
5. Pilih Hubungan kepercayaan, Sunting hubungan kepercayaan.
6. Konfirmasikan bahwa kebijakan tersebut mencantumkan ID akun dari akun pemantauan, atau ID organisasi dari organisasi yang memuat akun pemantauan tersebut.

Saya tidak melihat menu gerser-turun akun di dalam konsol

Pertama, periksa apakah Anda telah membuat peran IAM yang benar, seperti yang telah dibahas pada bagian pemecahan masalah sebelumnya. Jika pengaturan Anda sudah lakukan dengan benar, pastikan Anda telah memungkinkan akun ini untuk menampilkan data lintas akun, seperti yang dijelaskan di Enable Your Account to View Cross-Account Data.

Melakukan penonaktifan dan pembersihan setelah menggunakan lintas akun

Untuk menonaktifkan fungsionalitas lintas akun untuk CloudWatch, selesaikan langkah-langkah ini.

Langkah 1: Menghapus tumpukan atau peran lintas akun

Metode terbaik untuk melakukannya adalah dengan menghapus tumpukan AWS CloudFormation yang sudah Anda gunakan untuk mengaktifkan fungsionalitas lintas akun.

- Di masing-masing akun berbagi, hapus tumpukan CloudWatch-CrossAccountSharingRole.
- Jika Anda menggunakan AWS Organizations untuk mengaktifkan fungsionalitas lintas akun ini dengan semua akun yang ada dalam organisasi, maka Anda harus menghapus tumpukan CloudWatch-CrossAccountListAccountsRole di akun manajemen organisasi.

Jika Anda tidak menggunakan tumpukan AWS CloudFormation untuk mengaktifkan fungsionalitas lintas akun, selesaikan langkah berikut:

- Di masing-masing akun berbagi, hapus peran IAM CloudWatch-CrossAccountSharingRole.
- Jika Anda menggunakan AWS Organizations untuk mengaktifkan fungsionalitas lintas akun ini dengan semua akun yang ada dalam organisasi, maka Anda harus menghapus peran IAM CloudWatch-CrossAccountSharing-ListAccountsRole di akun manajemen organisasi.

Langkah 2: Menghapus peran tertaut layanan

Dalam akun pemantauan, hapus peran IAM tertaut layanan AWSServiceRoleForCloudWatchCrossAccount.

CloudWatch kuota layanan

CloudWatch memiliki kuota berikut untuk metrik, alarm, permintaan API, dan pemberitahuan email alarm.

> **Note**
>
> Untuk beberapa AWS layanan termasuk CloudWatch, Anda dapat menggunakan metrik CloudWatch penggunaan untuk memvisualisasikan penggunaan layanan Anda saat ini pada CloudWatch grafik dan dasbor. Anda dapat menggunakan fungsi matematika CloudWatch metrik untuk menampilkan kuota layanan untuk sumber daya tersebut pada grafik Anda. Anda juga dapat mengonfigurasi alarm yang memberi tahu Anda ketika penggunaan mendekati kuota layanan. Untuk informasi selengkapnya, lihat Memvisualisasikan kuota layanan dan mengatur alarm Anda.

Sumber daya	Kuota bawaan
Tindakan-tindakan alarm	5/alarm. Kuota ini tidak bisa diubah.
Periode evaluasi Alarm	Nilai maksimal, dihitung dengan mengalikan periode alarm dengan jumlah periode evaluasi yang digunakan , adalah satu hari (86.400 detik). Kuota ini tidak bisa diubah.
Alarm	10/bulan/pelanggan gratis. Alarm tambahan dikenakan biaya. Tidak ada batasan jumlah alarm per akun. Alarm yang didasarkan pada ekspresi matematika metrik dapat memiliki hingga 10 metrik. 200 alarm Wawasan Metrik per Wilayah. Anda dapat meminta penambahan kuota.
Model deteksi anomali	500 per Wilayah, per akun.

Sumber daya	Kuota bawaan
Permintaan API	1.000.000/bulan/pelanggan gratis.
Canary	200 per Wilayah, per akun. Anda dapat meminta penambahan kuota.
Permintaan API Wawasan Kontributor	API berikut memiliki kuota 20 transaksi per detik (TPS) dan per Wilayah. • DescribeInsightRules Kuota tidak dapat diubah. • GetInsightRuleReport Anda dapat meminta penambahan kuota. API berikut memiliki kuota 5 TPS per Wilayah. Kuota ini tidak bisa diubah. • DeleteInsightRules • PutInsightRule API berikut memiliki kuota 1 TPS per Wilayah. Kuota ini tidak bisa diubah. • DisableInsightRules • EnableInsightRules
Aturan Wawasan Kontributor	100 aturan per Wilayah, per akun. Anda dapat meminta penambahan kuota.
Metrik-metrik kustom	Tidak ada kuota.

Sumber daya	Kuota bawaan
Dasbor	Hingga 500 widget per dasbor. Hingga 500 metrik per widget dasbor. Hingga 2500 metrik per dasbor, di semua widget. Kuota ini mencakup semua metrik yang diambil untuk digunakan dalam fungsi matematika metrik, meskipun metrik tersebut tidak ditampilkan pada grafik. Kuota ini tidak dapat diubah.
DescribeAlarms	9 transaksi per detik (TPS) per Wilayah. Jumlah maksimal permintaan operasi yang dapat Anda lakukan per detik tanpa mengalami throttling. Anda dapat meminta penambahan kuota.
DeleteAlarms permintaan DescribeAlarmHistory permintaan DisableAlarmActions permintaan EnableAlarmActions permintaan SetAlarmState permintaan	3 TPS per Wilayah untuk setiap operasi ini. Jumlah maksimal permintaan operasi yang dapat Anda lakukan per detik tanpa mengalami throttling. Kuota-kuota ini tidak dapat diubah.
DescribeAlarmsForMetric permintaan	9 TPS per Wilayah. Jumlah maksimal permintaan operasi yang dapat Anda lakukan per detik tanpa mengalami throttling. Kuota ini tidak dapat diubah.
DeleteDashboards permintaan GetDashboard permintaan ListDashboards permintaan PutDashboard permintaan	10 TPS per Wilayah untuk setiap operasi ini. Jumlah maksimal permintaan operasi yang dapat Anda lakukan per detik tanpa mengalami throttling. Kuota ini tidak dapat diubah.

Sumber daya	Kuota bawaan
PutAnomalyDetector DescribeAnomalyDetectors	10 TPS per Wilayah. Jumlah maksimum permintaan operasi yang dapat Anda lakukan per detik tanpa dibatasi.
DeleteAnomalyDetector	5 TPS per Wilayah. Jumlah maksimal permintaan operasi yang dapat Anda lakukan per detik tanpa mengalami throttling.
Dimensi	30/metrik. Kuota ini tidak bisa diubah.

Sumber daya	Kuota bawaan
GetMetricData	10 TPS per Wilayah untuk operasi yang mencakup kueri Wawasan Metrik. Untuk operasi yang tidak menyertakan kueri Wawasan Metrik, kuota adalah 50 TPS per Wilayah. Ini adalah jumlah maksimal permintaan operasi yang dapat Anda lakukan per detik tanpa dibatasi. Anda dapat meminta penambahan kuota. Untuk operasi `GetMetricData` yang menyertakan kueri Wawasan Metrik, kuota adalah 4.300.000 Datapoint Per Second (DPS) selama 3 jam terakhir. Ini dihitung terhadap jumlah total titik data yang dipindai oleh kueri (yang dapat mencakup tidak lebih dari 10.000 metrik.) 180.000 Titik Data Per Detik (DPS) jika `StartTime` yang digunakan dalam permintaan API kurang dari atau sama dengan tiga jam dari waktu saat ini. 396.000 DPS jika `StartTime` lebih dari tiga jam dari waktu saat ini. Ini adalah jumlah maksimal titik data yang dapat Anda minta per detik menggunakan satu atau beberapa panggilan API tanpa dibatasi. Kuota ini tidak bisa diubah. DPS dihitung didasarkan pada perkiraan titik data, bukan titik data aktual. Perkiraan titik data dihitung menggunakan rentang waktu, periode, dan periode retensi yang diminta. Ini berarti bahwa jika titik data aktual dalam metrik yang diminta jarang atau kosong, throttling tetap terjadi jika perkiraan titik data melebihi kuota. Kuota DPS adalah per-Wilayah.

Sumber daya	Kuota bawaan
GetMetricData	Satu panggilan `GetMetricData` dapat mencakup hal-hal berikut ini: • Sebanyak 500 struktur `MetricDataQuery` . • Sebanyak 100 fungsi `SERVICE_QUOTA()` . • Sebanyak 100 fungsi `SEARCH()`. • Sebanyak 5 fungsi `LAMBDA()`. Kuota-kuota ini tidak dapat diubah.
GetMetricStatistics	400 TPS per Wilayah. Jumlah maksimal permintaan operasi yang dapat Anda lakukan per detik tanpa mengalami throttling. Anda dapat meminta kenaikan kuota.
GetMetricWidgetImage	Hingga 500 metrik per gambar. Kuota ini tidak bisa diubah. 20 TPS per Wilayah. Jumlah maksimal permintaan operasi yang dapat Anda lakukan per detik tanpa mengalami throttling. Anda dapat meminta penambahan kuota.
ListMetrics	25 TPS per Wilayah. Jumlah maksimal permintaan operasi yang dapat Anda lakukan per detik tanpa mengalami throttling. Anda dapat meminta penambahan kuota.
Nilai data metrik	Nilai titik data metrik harus dalam rentang -2^360 hingga 2^360. Nilai khusus (misalnya, NaN, +Angka tak terbatas, -Angka tak terbatas) tidak didukung. Kuota ini tidak bisa diubah.

Sumber daya	Kuota bawaan
MetricDatum item	PutMetricData1000/permintaan. Sebuah MetricDatumobjek dapat berisi nilai tunggal atau StatisticSetobjek yang mewakili banyak nilai. Kuota ini tidak bisa diubah.
Metrik-metrik	10/bulan/pelanggan gratis.
Kueri Wawasan Metrik	Satu kueri dapat memproses tidak lebih dari 10.000 metrik. Ini berarti bahwa jika klausa SELECT, FROM, dan WHERE akan cocok dengan lebih dari 10.000 metrik, hanya 10.000 metrik pertama yang ditemukan yang akan diproses oleh kueri. Satu kueri dapat mengembalikan tidak lebih dari 500 deret waktu. Anda hanya dapat menanyakan tiga jam data terbaru
Tarif permintaan API OAM.	1 TPS per Wilayah untuk PutSinkPolicy. 10 TPS per Wilayah untuk satu sama lain CloudWatch OAM API. Kuota ini mencerminkan jumlah maksimumal permintaan operasi yang dapat Anda lakukan per detik tanpa mengalami throttling. Kuota-kuota ini tidak dapat diubah.
Tautan akun sumber OAM	Setiap akun sumber dapat ditautkan ke sebanyak 5 akun pemantauan Kuota ini tidak dapat diubah.
Sink OAM	1 sink per akun Kuota ini tidak dapat diubah.

Sumber daya	Kuota bawaan
PutCompositeAlarm permintaan	3 TPS per Wilayah. Jumlah maksimal permintaan operasi yang dapat Anda lakukan per detik tanpa mengalami throttling. Anda dapat meminta penambahan kuota.
PutMetricAlarm permintaan	3 TPS per Wilayah. Jumlah maksimal permintaan operasi yang dapat Anda lakukan per detik tanpa mengalami throttling. Anda dapat meminta penambahan kuota.
PutMetricData permintaan	1MB untuk permintaan HTTP POST. PutMetricDatadapat menangani 500 transaksi per detik (TPS), yang merupakan jumlah maksimum permintaan operasi yang dapat Anda lakukan per detik tanpa dibatasi. `PutMetricData` dapat menangani 1.000 metrik per permintaan. Anda dapat meminta penambahan kuota.
Notifikasi email Amazon SNS	1.000/bulan/pelanggan gratis.
Grup Synthetics	20 per akun. Kuota ini tidak dapat diubah.
TagResource	20 TPS per Wilayah. Jumlah maksimal permintaa n operasi yang dapat Anda lakukan per detik tanpa mengalami throttling. Kuota ini tidak dapat diubah.
UntagResource	20 TPS per Wilayah. Jumlah maksimal permintaa n operasi yang dapat Anda lakukan per detik tanpa mengalami throttling. Kuota ini tidak dapat diubah.

Riwayat dokumen

Tabel berikut menjelaskan perubahan penting dalam setiap rilis Panduan CloudWatch Pengguna Amazon, dimulai pada Juni 2018. Untuk notifikasi tentang pembaruan dokumentasi ini, Anda dapat berlangganan ke umpan RSS.

Perubahan	Deskripsi	Tanggal
Amazon CloudWatch Internet Monitor sekarang mendukung pengamatan lintas akun	Anda sekarang dapat menggunakan observabilitas lintas akun Internet Monitor untuk memantau aplikasi Anda yang menjangkau beberapa Akun AWS dalam satu. Wilayah AWS	Maret 29, 2024
CloudWatchAgentServerPolicy dan CloudWatchAgentAdminPolicykebijakan diperbarui	CloudWatch menambahkan izin ke kedua `CloudWatchAgentAdminPolicy` kebijakan `CloudWatchAgentServerPolicy` dan untuk memungkinkan CloudWatch agen mempublikasikan jejak X-Ray dan memodifikasi periode penyimpanan grup log. Di kedua kebijakan, `xray:PutTraceSegments xray:PutTelemetryRecords ,xray:GetSamplingRules ,xray:GetSamplingTargets , xray:GetSamplingStatisticSummaries` dan	Februari 12, 2024

	`logs:PutRetentionPolicy` izin ditambahkan	
Peran terkait layanan baru dan kebijakan IAM untuk Monitor CloudWatch Jaringan	CloudWatch menambahkan peran terkait layanan baru, yang disebut. AWSServiceRoleForNetworkMonitor CloudWatch menambahkan peran terkait layanan baru ini untuk memungkinkan Anda membuat monitor untuk mengambil metrik jaringan antara subnet sumber dan alamat IP tujuan. Kebijakan CloudWatchNetworkMonitorServiceRolePolicyIAM baru dilampirkan pada peran ini, dan kebijakan tersebut memberikan izin CloudWatch untuk mengambil metrik jaringan atas nama Anda.	22 Desember 2023
CloudWatch merilis Monitor CloudWatch Jaringan Amazon	CloudWatch merilis fitur baru, Amazon CloudWatch Network Monitor. Ini adalah layanan pemantauan jaringan aktif baru yang mengidentifikasi apakah ada sebuah masalah jaringan dalam jaringan AWS atau jaringan perusahaan Anda sendiri.	22 Desember 2023

CloudWatchReadOnly Accesskebijakan diperbarui	CloudWatch menambahkan izin hanya-baca yang ada untuk Synthetics CloudWatch , X-Ray, dan CloudWatch RUM serta izin hanya-baca baru untuk Sinyal CloudWatch Aplikasi CloudWatchReadOnly Accessagar pengguna dengan kebijakan ini dapat melakukan triase dan mendiagnosis masalah kesehatan layanan seperti yang dilaporkan oleh Sinyal Aplikasi. CloudWatch `cloudwatch:GenerateQuery` Izin ditambahkan agar pengguna dengan kebijakan ini dapat menghasilkan string kueri CloudWatch Metrics Insights dari prompt bahasa alami.	5 Desember 2023
CloudWatchFullAccessKebijakan V2 diperbarui	CloudWatch menambahkan izin yang ada ke CloudWatchFullAccessV2 untuk CloudWatch Synthetics, X-Ray, CloudWatch dan RUM, dan menambahkan izin baru CloudWatch untuk Sinyal Aplikasi sehingga pengguna dengan kebijakan ini dapat sepenuhnya mengelola Sinyal Aplikasi untuk melakukan triase dan mendiagnosis masalah dengan kesehatan layanan.	5 Desember 2023

Peran terkait layanan baru dan kebijakan IAM baru	CloudWatch menambahkan peran terkait layanan baru, yang disebut. AWSServiceRoleForCloudWatchApplicationSignals CloudWatch menambahkan peran terkait layanan baru ini untuk memungkinkan Sinyal CloudWatch Aplikasi mengumpulkan data CloudWatch Log, data jejak X-Ray, data CloudWatch metrik, dan menandai data dari aplikasi yang telah Anda aktifkan untuk Sinyal Aplikasi. CloudWatch Kebijakan CloudWatchApplicationSignalsServiceRolePolicyIAM yang baru dilampirkan pada peran ini, dan kebijakan tersebut memberikan izin kepada Sinyal CloudWatch Aplikasi untuk mengumpulkan data pemantauan dan penandaan dari layanan terkait lainnya. AWS	30 November 2023

CloudWatch meluncurkan rilis Pratinjau Sinyal Aplikasi	CloudWatch Sinyal Aplikasi ada di Pratinjau. Gunakan Sinyal Aplikasi untuk menginstruksikan aplikasi Anda AWS sehingga Anda dapat memantau kesehatan aplikasi saat ini, membuat tujuan tingkat layanan (SLOs), dan melacak kinerja aplikasi jangka panjang terhadap tujuan bisnis Anda. Untuk informasi selengkapnya, silakan lihat Sinyal Aplikasi.	30 November 2023
CloudWatch menambahkan dukungan untuk menanyakan sumber data lain	Anda dapat menggunakan CloudWatch kueri, memvisual isasikan, dan membuat alarm untuk metrik dari sumber data lain Untuk informasi selengkap nya, lihat Menanyakan metrik dari sumber data lain.	26 November 2023
CloudWatch Metrics Insights mendukung pembuatan kueri bahasa alami	CloudWatch Metrics Insights mendukung kueri bahasa alami untuk menghasilkan dan memperbarui kueri. Untuk informasi selengkapnya, lihat Menggunakan bahasa alami untuk membuat dan memperbarui kueri Wawasan CloudWatch Metrik.	26 November 2023

CloudWatch merilis Container Insights dengan peningkatan observabilitas untuk Amazon EKS	CloudWatch merilis versi baru dari Container Insights. Versi ini mendukung peningkatan observabilitas untuk klaster Amazon EKS dan akan dapat mengumpulkan metrik-metrik yang lebih rinci dari klaster yang menjalankan Amazon EKS. Setelah instalasi selesai, ia secara otomatis akan mengumpulkan telemetri infrastruktur terperinci dan log kontainer untuk klaster-klaster Amazon EKS Anda. Anda kemudian dapat menggunak an dasbor yang dikuratori yang pada saat itu juga dapat digunakan untuk menelusuri telemetri aplikasi dan infrastru ktur secara lebih dalam.	6 November 2023
CloudWatch aliran metrik menambahkan pengaturan mitra cepat	CloudWatch aliran metrik sekarang menyediakan opsi pengaturan mitra cepat, yang dapat Anda gunakan untuk mengatur aliran metrik dengan cepat ke beberapa penyedia pihak ketiga.	17 Oktober 2023

CloudWatch merilis rekomendasi alarm	CloudWatch Synthetics sekarang menyediakan rekomendasi alarm untuk metrik dari layanan lain. AWS Rekomendasi ini dapat membantu Anda dalam mengidentifikasi metrik yang harus Anda tetapkan sebagai peringatan alarm untuk mengikuti praktik terbaik dalam melakukan pemantauan terhadap layanan-layanan ini.	16 Oktober 2023
CloudWatch Synthetics merilis runtime -6.0 syn-nodejs-puppeteer	CloudWatch Synthetics merilis runtime. `syn-nodejs-puppeteer-6.0`	26 September 2023
Menambahkan dukungan Amazon CloudWatch Application Insights untuk aplikasi lintas akun	Anda sekarang dapat berbagi CloudWatch aplikasi Application Insights di seluruh batas akun.	26 September 2023

Peran terkait layanan baru dan kebijakan IAM baru	CloudWatch menambahkan peran terkait layanan baru, yang disebut. AWSServiceRoleForCloudWatchMetrics_DbPerfInsights CloudWatch menambahkan peran terkait layanan baru ini untuk memungkinkan pengambilan metrik Performance Insights CloudWatch untuk mengkhawatirkan, deteksi anomali, dan snapshotting. Kebijakan AWSServiceRoleForCloudWatchMetrics_DbPerfInsightsServiceRolePolicyIAM baru dilampirkan pada peran ini, dan kebijakan tersebut memberikan izin CloudWatch untuk mengambil metrik Performance Insights atas nama Anda.	20 September 2023
Menambahkan fungsi matematika metrik baru	CloudWatch menambahkan fungsi matematika metrik baru`DB_PERF_INSIGHTS` , yang dapat Anda gunakan untuk mengambil metrik Performance Insights AWS dari layanan database untuk mengkhawatirkan, deteksi anomali, dan snapshotting.	20 September 2023

CloudWatchReadOnly Accesskebijakan diperbarui	CloudWatch menambahkan `application-autoscaling:DescribeScalingPolicies` izin CloudWatchReadOnly Accessagar pengguna dengan kebijakan ini dapat mengakses informasi tentang kebijakan Application Auto Scaling.	14 September 2023
CloudWatch agen menambahkan dukungan untuk AL2023	CloudWatch Agen mendukung AL2023.	8 Agustus 2023
Kebijakan IAM terkelola baru, V2 CloudWatchFullAccess	CloudWatch menambahkan kebijakan baru CloudWatchFullAccessV2. Kebijakan ini memberikan akses penuh ke CloudWatch tindakan dan sumber daya sambil dengan lebih baik mencakup izin yang diberikan ke layanan lain seperti Amazon SNS dan. Amazon EC2 Auto Scaling	1 Agustus 2023
Peran terkait layanan yang diperbarui untuk Amazon CloudWatch Internet Monitor - perbarui ke kebijakan yang ada	Menambahkan izin baru, `elasticloadbalancing:DescribeLoadBalancers` dan `ec2:DescribeNetworkInterfaces` , untuk peran terkait layanan untuk Internet Monitor, untuk mendukung pemantauan lalu lintas untuk sumber daya Penyeimbang Beban Jaringan tertentu.	25 Juli 2023

Menambahkan dukungan untuk sumber daya Network Load Balancer di Amazon CloudWatch Internet Monitor	Menambahkan dukungan untuk membuat sebuah monitor di Monitor Internet dengan sumber daya Penyeimbang Beban Jaringan tertentu, untuk memberikan tingkat observabilitas yang lebih terperinci untuk aplikasi Anda.	25 Juli 2023
Fitur variabel dasbor	CloudWatch variabel dasbor yang dirilis, yang dapat Anda gunakan untuk membuat dasbor fleksibel yang dapat dengan cepat menampilk an konten yang berbeda tergantung pada bagaimana Anda mengatur satu bidang input di dalam dasbor. Sebagai contoh, Anda dapat membuat sebuah dasbor yang dapat beralih dengan cepat antara fungsi Lambda yang berbeda atau ID instans Amazon EC2, atau yang dapat beralih ke Wilayah AWS yang berbeda. Untuk informasi selengkapnya, silakan lihat Membuat dasbor fleksibel dengan variabel dasbor.	28 Juni 2023

Internet Monitor sekarang mendukung penyesuaian ambang batas untuk peristiwa-peristiwa kondisi	Internet Monitor menambahkan kemampuan untuk menyesuaikan ambang batas ketika skor performa global atau skor ketersediaan memicu terjadinya sebuah peristiwa kondisi. Untuk informasi selengkapnya, lihat Melacak kinerja dan ketersediaan real-time di Amazon CloudWatch Internet Monitor.	26 Juni 2023
Internet Monitor sekarang mendukung semua Wilayah komersial	Internet Monitor menambahkan tujuh baru Wilayah AWS dan sekarang mendukung semua Wilayah komersial.	19 Juni 2023
Versi ekstensi Wawasan Lambda baru	CloudWatch menambahkan versi 1.0.229.0 dari ekstensi Lambda Insights untuk platform x86-64 dan platform ARM64. Untuk informasi selengkapnya, silakan lihat Versi ekstensi Wawasan Lambda yang tersedia.	12 Juni 2023

CloudWatchReadOnly Accesskebijakan diperbarui	CloudWatch menambahkan izin ke CloudWatchReadOnlyAccess. Izin `logs:StartLiveTail` dan `logs:StopLiveTail` izin ditambahkan sehingga pengguna dengan kebijakan ini dapat menggunakan konsol untuk memulai dan menghentikan sesi ekor langsung CloudWatch Log. Untuk informasi selengkapnya, silakan lihat Menggunakan live tail untuk melihat log mendekati waktu nyata.	6 Juni 2023
CloudWatch RUM menambahkan dukungan untuk metrik khusus	Anda dapat menggunakan monitor aplikasi CloudWatch RUM untuk membuat metrik khusus dan mengirimkannya ke CloudWatch dan CloudWatch Terbukti. Fitur ini mencakup pembaruan ke kebijakan AmazonCloudWatchIAM ServiceRolePolicy terkelola RUM. Dalam kebijakan itu, kunci kondisi diubah sehingga CloudWatch RUM dapat mengirim metrik khusus ke ruang nama metrik khusus.	9 Februari 2023

Kebijakan terkelola baru dan diperbarui untuk CloudWatch	Untuk mendukung pengamatan CloudWatch lintas akun, `CloudWatchReadOnlyAccess` kebijakan `CloudWatchFullAccess` dan kebijakan telah diperbarui, dan kebijakan terkelola baru berikut telah ditambahkan:`CloudWatchCrossAccountSharingConfiguration` ,`OAMFullAccess` , dan. `OAMReadOnlyAccess` Untuk informasi selengkapnya, lihat CloudWatch pembaruan kebijakan AWS terkelola.	7 Februari 2023
CloudWatch Pembaruan kebijakan peran terkait layanan Wawasan Aplikasi — pembaruan ke kebijakan yang ada.	CloudWatch Application Insights memperbarui kebijakan peran AWS terkait layanan yang ada.	19 Desember 2022
Amazon CloudWatch Application Insights mendukung aplikasi kontainer dan layanan mikro dari konsol Container Insights.	Anda dapat menampilkan masalah yang terdeteksi CloudWatch Application Insights untuk Amazon ECS dan Amazon EKS di dasbor Container Insights.	17 November 2021
Pemantauan Wawasan CloudWatch Aplikasi Amazon untuk database SAP HANA.	Anda dapat memantau basis data SAP HANA dengan Wawasan Aplikasi.	15 November 2021
Dukungan Amazon CloudWatch Application Insights untuk memantau semua sumber daya dalam akun.	Anda dapat melakukan onboarding dan memantau semua sumber daya yang ada di sebuah akun.	15 September 2021

Dukungan Amazon CloudWatch Application Insights untuk Amazon FSx.	Anda dapat memantau metrik-metrik yang diambil dari Amazon FSx.	31 Agustus 2021
Metrik SDK tidak lagi didukung.	CloudWatch Metrik SDK tidak lagi didukung.	25 Agustus 2021
Dukungan Amazon CloudWatch Application Insights untuk menyiapkan pemantauan kontainer.	Anda dapat memantau kontainer menggunakan praktik terbaik dengan Amazon CloudWatch Application Insights.	18 Mei 2021
Aliran metrik tersedia secara umum	Anda dapat menggunakan aliran metrik untuk terus mengalirkan CloudWatch metrik ke tujuan pilihan Anda. Untuk informasi selengkapnya, lihat Aliran metrik di Panduan CloudWatch Pengguna Amazon.	31 Maret 2021
Pemantauan Wawasan CloudWatch Aplikasi Amazon untuk database Oracle di Amazon RDS dan Amazon EC2.	Anda dapat memantau metrik dan log yang diambil dari Oracle dengan Amazon CloudWatch Application Insights.	16 Januari 2021

Wawasan Lambda tersedia secara umum	CloudWatch Lambda Insights adalah solusi pemantauan dan pemecahan masalah untuk aplikasi tanpa server yang berjalan. AWS Lambda Untuk informasi selengkap nya, lihat Menggunakan Wawasan Lambda di Panduan Pengguna Amazon CloudWatc h .	3 Desember 2020
Pemantauan Wawasan CloudWatch Aplikasi Amazon untuk metrik eksportir Prometheus JMX.	Anda dapat memantau metrik yang diambil dari eksportir Prometheus JMX dengan Amazon Application Insights. CloudWatch	20 November 2020
CloudWatch Synthetics merilis versi runtime baru	CloudWatch Synthetics telah merilis versi runtime baru. Untuk informasi selengkapnya, lihat Versi Runtime Canary di CloudWatch Panduan Pengguna Amazon.	11 September 2020
Pemantauan Wawasan CloudWatch Aplikasi Amazon untuk Postgre SQL di Amazon RDS dan Amazon EC2.	Anda dapat melakukan pemantauan atas aplikasi-aplikasi yang dibuat dengan PostgreSQL yang berjalan di Amazon RDS atau Amazon EC2.	11 September 2020

CloudWatch mendukung berbagi dasbor	Sekarang Anda dapat berbagi CloudWatch dasbor dengan orang-orang di luar organisasi dan AWS akun Anda. Untuk informasi selengkapnya, lihat Berbagi CloudWatch Dasbor di Panduan CloudWatch Pengguna Amazon.	10 September 2020
Siapkan monitor untuk aplikasi.NET menggunakan SQL Server di backend dengan Application Insights CloudWatch	Anda dapat menggunakan tutorial dokumentasi untuk membantu Anda mengatur monitor untuk aplikasi.NET menggunakan SQL Server di backend dengan CloudWatch Application Insights.	19 Agustus 2020
AWS CloudFormation dukungan untuk CloudWatch aplikasi Amazon Application Insights.	Anda dapat menambahkan pemantauan CloudWatch Application Insights, termasuk metrik utama dan telemetri, ke aplikasi, database, dan server web Anda, langsung dari template. AWS CloudFormation	30 Juli 2020
Pemantauan Wawasan CloudWatch Aplikasi Amazon untuk Aurora untuk kluster database MySQL.	Anda dapat memantau Aurora untuk cluster database MySQL (RDS Aurora) dengan Amazon Application Insights. CloudWatch	2 Juli 2020

CloudWatch Kontributor Wawasan ketersediaan umum	CloudWatch Contributor Insights sekarang tersedia secara umum. Hal ini akan memungkinkan Anda untuk menganalisis data log dan membuat deret waktu yang menampilkan data kontributor. Anda dapat melihat metrik tentang kontributor-kontributor N teratas, total kontributor unik, dan penggunaannya. Untuk informasi selengkapnya, lihat Menggunakan Wawasan Kontributor untuk Menganalisis Data Kardinalitas Tinggi di Panduan Pengguna Amazon. CloudWatch	2 April 2020
CloudWatch Pratinjau publik Synthetics	CloudWatch Synthetics sekarang dalam pratinjau publik. Hal ini akan memungkinkan Anda membuat canary untuk memantau titik akhir dan API. Untuk informasi selengkapnya, lihat Menggunakan Canary di Panduan CloudWatch Pengguna Amazon.	25 November 2019

CloudWatch Pratinjau publik Contributor Insights	CloudWatch Contributor Insights sekarang dalam pratinjau publik. Hal ini akan memungkinkan Anda untuk menganalisis data log dan membuat deret waktu yang menampilkan data kontributor. Anda dapat melihat metrik tentang kontributor-kontributor N teratas, total kontributor unik, dan penggunaannya. Untuk informasi selengkapnya, lihat Menggunakan Wawasan Kontributor untuk Menganalisis Data Kardinalitas Tinggi di Panduan Pengguna Amazon. CloudWatch	25 November 2019
CloudWatch meluncurkan fitur ServiceLens	ServiceLens memudahkan pengamatan layanan dan aplikasi Anda dengan memungkinkan Anda mengintegrasikan jejak, metrik, log, dan alarm ke satu tempat. ServiceLens terintegrasi CloudWatch dengan AWS X-Ray untuk memberikan end-to-end tampilan aplikasi Anda.	21 November 2019

Gunakan CloudWatch untuk secara proaktif mengelola kuota AWS layanan Anda	Anda dapat menggunakan CloudWatch untuk secara proaktif mengelola kuota AWS layanan Anda. CloudWatch Metrik penggunaan memberikan visibilitas ke dalam penggunaan sumber daya dan operasi API akun Anda. Untuk informasi selengkapnya, lihat Integrasi Service Quotas dan Metrik Penggunaan di Panduan Pengguna Amazon CloudWatch .	19 November 2019
CloudWatch mengirim acara saat alarm mengubah status	CloudWatch sekarang mengirim acara ke Amazon EventBridge ketika CloudWatch alarm berubah status. Untuk informasi selengkapnya, lihat Acara Alarm dan EventBridge di Panduan CloudWatch Pengguna Amazon.	8 Oktober 2019

Wawasan Kontainer	CloudWatch Wawasan Kontainer sekarang tersedia secara umum. Aplikasi ini akan memungkinkan Anda untuk mengumpul kan, menggabungkan, serta merangkum metrik dan log dari aplikasi dan layanan mikro terkontainerisasi Anda. Untuk informasi selengkapnya, lihat Menggunakan Wawasan Kontainer di Panduan CloudWatch Pengguna Amazon.	30 Agustus 2019
Pembaruan untuk metrik-metrik pratinjau Wawasan Kontainer di Amazon EKS dan Kubernetes	Pratinjau publik Wawasan Kontainer di Amazon EKS dan Kubernetes telah diperbarui. InstanceId sekarang disertaka n sebagai dimensi untuk instance EC2 cluster. Hal ini akan memungkinkan alarm yang telah dibuat pada metrik-metrik ini untuk memicu tindakan-tindakan EC2 berikut: Berhenti, Akhiri, Mulai Ulang, atau Pulihkan. Selain itu, pod dan metrik layanan sekarang akan dilaporkan oleh namespace Kubernete s untuk menyederhanakan pemantauan dan pengaturan alarm pada metrik berdasark an namespace.	19 Agustus 2019

Pembaruan untuk AWS Systems Manager OpsCenter integrasi	Pembaruan tentang bagaimana CloudWatch Application Insights terintegrasi dengan Systems Manager. OpsCenter	7 Agustus 2019
CloudWatch metrik penggunaan	CloudWatch Metrik penggunaan membantu Anda melacak penggunaan CloudWatch sumber daya Anda dan tetap dalam batas layanan Anda. Untuk informasi selengkapnya, lihat https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/CloudWatch-Usage-Metrics.html.	6 Agustus 2019
CloudWatch Pratinjau publik Wawasan Kontainer	CloudWatch Wawasan Kontainer sekarang dalam pratinjau publik. Aplikasi ini akan memungkinkan Anda untuk mengumpulkan, menggabungkan, serta merangkum metrik dan log dari aplikasi dan layanan mikro terkontainerisasi Anda. Untuk informasi selengkapnya, lihat Menggunakan Wawasan Kontainer di Panduan CloudWatch Pengguna Amazon.	9 Juli 2019

CloudWatch Pratinjau publik Deteksi Anomali	CloudWatch deteksi anomali sekarang dalam pratinjau publik. CloudWatch menerapkan algoritma pembelajaran mesin ke data masa lalu metrik untuk membuat model nilai yang diharapkan metrik. Anda dapat menggunakan model ini untuk membuat visualisasi dan untuk mengatur alarm. Untuk informasi selengkapnya, lihat Menggunakan Deteksi CloudWatch Anomali di CloudWatch Panduan Pengguna Amazon.	9 Juli 2019
CloudWatch Wawasan Aplikasi untuk .NET dan SQL Server	CloudWatch Application Insights untuk .NET dan SQL Server memfasilitasi observabilitas untuk aplikasi. NET dan SQL Server. Aplikasi ini akan dapat membantu Anda mengatur monitor terbaik untuk sumber daya aplikasi Anda agar dapat secara terus-menerus melakukan analisis data untuk melihat tanda-tanda dari munculnya masalah dengan aplikasi Anda.	21 Juni 2019

CloudWatch bagian agen diatur ulang	Dokumentasi CloudWatch agen telah ditulis ulang untuk meningkatkan kejelasan, terutama bagi pelanggan yang menggunakan baris perintah untuk menginstal dan mengkonfigurasi agen. Untuk informasi selengkapnya, lihat Mengumpulkan Metrik dan Log dari Instans Amazon EC2 dan Server Lokal dengan CloudWatch Agen di Panduan Pengguna Amazon. CloudWatch	28 Maret 2019
Fungsi SEARCH ditambahkan ke ekspresi matematika metrik	Sekarang Anda dapat menggunakan fungsi SEARCH dalam ekspresi matematika metrik. Hal ini akan memungkinkan Anda untuk membuat dasbor yang diperbarui secara otomatis ketika sumber daya baru yang dibuat cocok dengan kueri pencarian. Untuk informasi selengkapnya, lihat Menggunakan Ekspresi Penelusuran dalam Grafik di Panduan CloudWatch Pengguna Amazon.	21 Maret 2019

AWS Metrik SDK untuk Dukungan Perusahaan	SDK Metrics membantu Anda menilai kesehatan AWS layanan Anda dan mendiagnosis latensi yang disebabkan oleh mencapai batas penggunaan akun Anda atau oleh pemadaman layanan. Untuk informasi selengkapnya, lihat Memantau Aplikasi Menggunakan Metrik AWS SDK di CloudWatch Panduan Pengguna Amazon.	11 Desember 2018
Alarm-alarm tentang ekspresi matematika	CloudWatch mendukung pembuatan alarm berdasarkan ekspresi matematika metrik. Untuk informasi selengkapnya, lihat Alarm tentang Ekspresi Matematika di Panduan CloudWatch Pengguna Amazon.	20 November 2018
Halaman beranda CloudWatch konsol baru	Amazon telah membuat halaman beranda baru di CloudWatch konsol, yang secara otomatis menampilkan metrik kunci dan alarm untuk semua AWS layanan yang Anda gunakan. Untuk informasi selengkapnya, lihat Memulai Amazon CloudWatch di Panduan CloudWatch Pengguna Amazon.	19 November 2018

AWS CloudFormation template untuk CloudWatch Agen	Amazon telah mengunggah AWS CloudFormation template yang dapat Anda gunakan untuk menginstal dan memperbarui CloudWatch agen. Untuk informasi selengkapnya, lihat Menginstal CloudWatch Agen di Instans Baru yang Digunakan AWS CloudFormation di Panduan CloudWatch Pengguna Amazon.	9 November 2018
Penyempurnaan untuk Agen CloudWatch	CloudWatch Agen telah diperbarui untuk bekerja dengan protokol StatSD dan collectd. Hal ini juga telah meningkatkan dukungan lintas akun. Untuk informasi selengkapnya, lihat Mengambil Metrik Kustom dengan StatSD, Mengambil Metrik Kustom dengan collectd, dan Mengirim Metrik dan Log ke Akun Berbeda di Panduan Pengguna Amazon. AWS CloudWatch	28 September 2018

Dukungan untuk titik akhir VPC Amazon VPC	Anda sekarang dapat membuat koneksi pribadi antara VPC Anda dan. CloudWatch Untuk informasi selengkapnya, lihat Menggunakan CloudWatch dengan Titik Akhir VPC Antarmuka di Panduan Pengguna Amazon CloudWatch .	28 Juni 2018

Tabel berikut menjelaskan perubahan penting pada Panduan CloudWatch Pengguna Amazon sebelum Juni 2018.

Perubahan	Deskripsi	Tanggal rilis
Matematika metrik	Anda sekarang dapat melakukan ekspresi matematika pada CloudWatch metrik, menghasilkan deret waktu baru yang dapat Anda tambahkan ke grafik di dasbor Anda. Untuk informasi selengkapnya, lihat Gunakan matematika metrik.	4 April 2018
Alarm "M dari N"	Sekarang, Anda dapat mengonfigurasi sebuah alarm untuk memicu berdasarkan pada titik data "M dari N" dalam setiap interval evaluasi alarm. Untuk informasi selengkapnya, lihat Melakukan evaluasi alarm.	8 Desember 2017
CloudWatch agen	CloudWatch Agen terpadu baru dirilis. Anda dapat menggunakan agen multi-platform terpadu untuk mengumpulkan metrik sistem dan file log dari instans Amazon EC2 dan server on-premise. Agen yang baru mendukung Windows dan Linux, serta memungkinkan Anda untuk memilih metrik yang akan dikumpulkan, termasuk metrik sub-sumber daya, seperti inti per CPU. Untuk informasi	7 September 2017

Perubahan	Deskripsi	Tanggal rilis
	selengkapnya, lihat Kumpulkan metrik, log, dan jejak dengan agen CloudWatch .	
Metrik NAT gateway	Menambahkan metrik-metrik untuk NAT gateway Amazon VPC.	7 September 2017
Metrik resolusi tinggi	Jika Anda mau, Sekarang Anda dapat menyiapkan metrik-metrik kustom sebagai metrik resolusi tinggi, dengan granularitas serendah satu detik. Untuk informasi selengkapnya, lihat Metrik resolusi tinggi.	26 Juli 2017
API Dasbor	Anda sekarang dapat membuat, mengubah, dan menghapus dasbor dengan menggunakan API dan AWS CLI. Untuk informasi selengkapnya, lihat Membuat CloudWatch dasbor.	6 Juli 2017
AWS Direct Connect metrik	Menambahkan metrik untuk AWS Direct Connect.	29 Juni 2017
Metrik Amazon VPC VPN	Menambahkan metrik untuk Amazon VPC VPN.	15 Mei 2017
AppStream 2.0 metrik	Menambahkan metrik untuk AppStream 2.0.	8 Maret 2017
CloudWatch pemilih warna konsol	Sekarang Anda dapat memilih warna untuk masing-masing metrik pada widget dasbor Anda. Untuk informasi selengkapnya, lihat Edit grafik di CloudWatch dasbor .	27 Februari 2017
Alarm pada dasbor	Sekarang alarm dapat Anda tambahkan ke dasbor. Untuk informasi selengkapnya, lihat Menambahkan atau menghapus widget alarm dari CloudWatch dasbor .	15 Februari 2017

Perubahan	Deskripsi	Tanggal rilis
Menambahkan metrik untuk Amazon Polly	Menambahkan metrik untuk Amazon Polly.	1 Desember 2016
Menambahkan metrik untuk Layanan Terkelola Amazon untuk Apache Flink	Menambahkan metrik untuk Layanan Terkelola Amazon untuk Apache Flink.	1 Desember 2016
Menambahkan dukungan untuk statistik persentil	Anda dapat menentukan persentil apa pun dengan menggunakan hingga dua tempat desimal (misalnya, p95.45). Untuk informasi selengkapnya, lihat Persentil.	17 November 2016
Menambahkan metrik untuk Amazon Simple Email Service	Menambahkan metrik untuk Amazon Simple Email Service.	2 November 2016
Memperbarui retensi metrik	Amazon CloudWatch sekarang menyimpan data metrik selama 15 bulan, bukan 14 hari.	1 November 2016
Antarmuka konsol metrik telah diperbarui	CloudWatch Konsol diperbarui dengan peningkatan fungsionalitas yang ada dan fungsionalitas baru.	1 November 2016
Menambahkan metrik untuk Amazon Elastic Transcoder	Menambahkan metrik untuk Amazon Elastic Transcoder.	20 September 2016

Perubahan	Deskripsi	Tanggal rilis
Menambahkan metrik untuk Amazon API Gateway	Menambahkan metrik untuk Amazon API Gateway.	9 September 2016
Menambahkan metrik untuk AWS Key Management Service	Menambahkan metrik untuk AWS Key Management Service.	9 September 2016
Menambahkan metrik-metrik untuk Penyeimbang Beban Aplikasi baru yang didukung oleh Penyeimbang Beban Elastis	Menambahkan metrik-metrik untuk Penyeimbang Beban Aplikasi.	11 Agustus 2016
Ditambahkan baru NetworkPacketsIn dan NetworkPacketsOut metrik untuk Amazon EC2	Ditambahkan baru NetworkPacketsIn dan NetworkPacketsOut metrik untuk Amazon EC2.	23 Maret 2016
Menambahkan metrik baru untuk armada spot Amazon EC2	Menambahkan metrik baru untuk armada spot Amazon EC2.	21 Maret 2016

Perubahan	Deskripsi	Tanggal rilis
Menambahkan metrik CloudWatch Log baru	Menambahkan metrik CloudWatch Log baru.	10 Maret 2016
Menambahkan OpenSearch Layanan Amazon dan AWS WAF metrik dan dimensi	Menambahkan OpenSearch Layanan Amazon serta AWS WAF metrik dan dimensi.	14 Oktober 2015
Menambahkan dukungan untuk CloudWatch dasbor	Dasbor adalah halaman beranda yang dapat disesuaikan di CloudWatch konsol yang dapat Anda gunakan untuk memantau sumber daya Anda dalam satu tampilan, bahkan yang tersebar di berbagai Wilayah. Untuk informasi selengkapnya, lihat Menggunakan CloudWatch dasbor Amazon.	8 Oktober 2015
Menambahkan AWS Lambda metrik dan dimensi	Menambahkan AWS Lambda metrik dan dimensi.	4 September 2015
Menambahkan metrik dan dimensi Amazon Elastic Container Service	Menambahkan metrik dan dimensi Amazon Elastic Container Service.	17 Agustus 2015

Perubahan	Deskripsi	Tanggal rilis
Menambahkan metrik dan dimensi Amazon Simple Storage Service	Menambahkan metrik dan dimensi Amazon Simple Storage Service.	26 Juli 2015
Fitur baru: Mulai ulang tindakan alarm	Tindakan alarm boot ulang dan peran IAM baru untuk digunakan dengan tindakan alarm ditambahkan. Untuk informasi selengkapnya, lihat Membuat alarm untuk menghentikan, mengakhiri, menyalakan ulang, atau memulihkan instans EC2.	23 Juli 2015
Menambahkan WorkSpaces metrik dan dimensi Amazon	Menambahkan WorkSpaces metrik dan dimensi Amazon.	30 April 2015
Menambahkan metrik dan dimensi Amazon Machine Learning	Menambahkan metrik dan dimensi Amazon Machine Learning.	9 April 2015
Fitur baru: Tindakan-tindaka alarm pemulihan instans Amazon EC2	Tindakan-tindakan alarm yang diperbarui guna menyertakan tindakan pemulihan instans EC2 yang baru. Untuk informasi selengkapnya, lihat Membuat alarm untuk menghentikan, mengakhiri, menyalakan ulang, atau memulihkan instans EC2.	12 Maret 2015
Menambahkan CloudSearch metrik CloudFront dan dimensi Amazon dan Amazon	Menambahkan CloudSearch metrik CloudFront dan dimensi Amazon dan Amazon.	6 Maret 2015

Perubahan	Deskripsi	Tanggal rilis
Menambahkan metrik dan dimensi Amazon Simple Workflow Service	Menambahkan metrik dan dimensi Amazon Simple Workflow Service.	9 Mei 2014
Panduan yang diperbarui untuk menambahkan dukungan AWS CloudTrail	Menambahkan topik baru untuk menjelaskan bagaimana Anda dapat menggunakan AWS CloudTrail untuk mencatat aktivitas di Amazon CloudWatch. Untuk informasi selengkapnya, lihat Mencatat panggilan CloudWatch API Amazon dengan AWS CloudTrail.	30 April 2014
Panduan yang diperbarui untuk menggunakan new AWS Command Line Interface (AWS CLI)	CLI adalah AWS CLI lintas layanan dengan instalasi yang disederhanakan, konfigurasi terpadu, dan sintaks baris perintah yang konsisten. AWS CLI didukung di Linux/Unix, Windows, dan Mac. Contoh CLI dalam panduan ini telah diperbarui untuk menggunakan CLI AWS baru. Untuk informasi tentang cara menginstal dan mengkonfigurasi AWS CLI baru, lihat Menyiapkan dengan AWS CLI Antarmuka di AWS Command Line Interface Panduan Pengguna.	21 Februari 2014
Menambahkan Amazon Redshift serta AWS OpsWorks metrik dan dimensi	Menambahkan Amazon Redshift serta AWS OpsWorks metrik dan dimensi.	16 Juli 2013
Menambahkan metrik dan dimensi Amazon Route 53	Menambahkan metrik dan dimensi Amazon Route 53.	26 Juni 2013

Perubahan	Deskripsi	Tanggal rilis
Fitur baru: Tindakan CloudWatch Alarm Amazon	Menambahkan bagian baru untuk mendokumentasikan tindakan CloudWatch alarm Amazon, yang dapat Anda gunakan untuk menghentikan atau menghentikan instans Amazon Elastic Compute Cloud. Untuk informasi selengkapnya, lihat Membuat alarm untuk menghentikan, mengakhiri, menyalakan ulang, atau memulihkan instans EC2.	8 Januari 2013
Pembaruan metrik EBS	Metrik EBS yang sudah diperbarui untuk menyertakan dua metrik baru untuk volume IOPS yang Tersedia.	20 November 2012
Peringatan penagihan baru	Sekarang Anda dapat memantau AWS tagihan menggunakan CloudWatch metrik Amazon dan membuat alarm untuk memberi tahu Anda ketika Anda telah melampaui ambang batas yang ditentukan. Untuk informasi selengkapnya, lihat Buat alarm penagihan untuk memantau perkiraan AWS biaya Anda.	10 Mei 2012
Metrik baru	Sekarang Anda dapat mengakses enam metrik Penyeimbangan Beban Elastis baru yang memberikan hitungan dari berbagai kode respons HTTP.	19 Oktober 2011
Fitur baru	Sekarang Anda dapat mengakses metrik dari Amazon EMR.	30 Juni 2011
Fitur baru	Anda sekarang dapat mengakses metrik dari Amazon Simple Notification Service dan Amazon Simple Queue Service.	14 Juli 2011

Perubahan	Deskripsi	Tanggal rilis
Fitur baru	Menambahkan informasi tentang menggunakan API `PutMetricData` untuk menerbitkan metrik kustom. Untuk informasi selengkapnya, lihat Menerbitkan metrik kustom.	10 Mei 2011
Memperbarui retensi metrik	Amazon CloudWatch sekarang mempertahankan riwayat alarm selama dua minggu, bukan enam minggu. Dengan perubahan ini, periode retensi alarm sesuai dengan periode retensi untuk data metrik.	7 April 2011
Fitur baru	Menambahkan kemampuan untuk mengirim Amazon Simple Notification Service atau notifikasi penskalaan otomatis (Auto Scaling) ketika ada sebuah metrik yang telah melampaui ambang batas. Untuk informasi selengkapnya, lihat Alarm.	2 Desember 2010
Fitur baru	Sejumlah CloudWatch tindakan sekarang termasuk MaxRecords dan NextToken parameter, yang memungkinkan Anda untuk mengontrol halaman hasil untuk ditampilkan.	2 Desember 2010
Fitur baru	Layanan ini sekarang terintegrasi dengan AWS Identity and Access Management (IAM).	2 Desember 2010

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.