aws

Panduan Developer

Amazon Timestream

Amazon Timestream: Panduan Developer

Table of Contents

Untuk apa Amazon Timestream? LiveAnalytics

Amazon Timestream for LiveAnalytics adalah database deret waktu yang cepat, terukur, dikelola sepenuhnya, dan dibuat khusus yang memudahkan penyimpanan dan analisis triliunan titik data deret waktu per hari. Timestream untuk LiveAnalytics menghemat waktu dan biaya dalam mengelola siklus hidup data deret waktu dengan menyimpan data terbaru dalam memori dan memindahkan data historis ke tingkat penyimpanan yang dioptimalkan biaya berdasarkan kebijakan yang ditentukan pengguna. Timestream untuk LiveAnalytics mesin kueri yang dibuat khusus memungkinkan Anda mengakses dan menganalisis data terbaru dan historis bersama-sama, tanpa harus menentukan lokasinya. Amazon Timestream for LiveAnalytics memiliki fungsi analitik deret waktu bawaan, membantu Anda mengidentifikasi tren dan pola dalam data Anda dalam waktu dekat. Timestream for LiveAnalytics adalah tanpa server dan secara otomatis menskalakan naik atau turun untuk menyesuaikan kapasitas dan kinerja. Karena Anda tidak perlu mengelola infrastruktur yang mendasarinya, Anda dapat fokus pada pengoptimalan dan pembuatan aplikasi Anda.

Timestream for LiveAnalytics juga terintegrasi dengan layanan yang umum digunakan untuk pengumpulan data, visualisasi, dan pembelajaran mesin. Anda dapat mengirim data ke Amazon Timestream untuk LiveAnalytics digunakan AWS IoT Core, Amazon Kinesis, MSK Amazon, dan Telegraf open source. Anda dapat memvisualisasikan data menggunakan Amazon QuickSight, Grafana, dan alat intelijen bisnis melalui. JDBC Anda juga dapat menggunakan Amazon SageMaker dengan Timestream LiveAnalytics untuk pembelajaran mesin.

Timestream untuk manfaat LiveAnalytics utama

Manfaat utama Amazon Timestream adalah: LiveAnalytics

- Tanpa server dengan auto-scaling - Dengan Amazon Timestream LiveAnalytics untuk, tidak ada server untuk dikelola dan tidak ada kapasitas untuk menyediakan. Seiring kebutuhan aplikasi Anda berubah, Timestream untuk LiveAnalytics secara otomatis menskalakan untuk menyesuaikan kapasitas.
- Manajemen siklus hidup data - Amazon Timestream LiveAnalytics untuk menyederhanakan proses kompleks manajemen siklus hidup data. Ini menawarkan tingkat penyimpanan, dengan penyimpanan memori untuk data terbaru dan penyimpanan magnetik untuk data historis. Amazon Timestream mengotomatiskan transfer data dari penyimpanan memori ke penyimpanan magnetik berdasarkan kebijakan yang dapat dikonfigurasi pengguna.

- Akses data yang disederhanakan - Dengan Amazon Timestream for LiveAnalytics, Anda tidak perlu lagi menggunakan alat yang berbeda untuk mengakses data terbaru dan historis. Mesin kueri Amazon Timestream for LiveAnalytics yang dibuat khusus secara transparan mengakses dan menggabungkan data di seluruh tingkatan penyimpanan tanpa Anda harus menentukan lokasi data.
- Dibuat khusus untuk deret waktu - Anda dapat dengan cepat menganalisis data deret waktu menggunakanSQL, dengan fungsi deret waktu bawaan untuk perataan, perkiraan, dan interpolasi. Timestream untuk LiveAnalytics juga mendukung agregat lanjutan, fungsi jendela, dan tipe data yang kompleks seperti array dan baris.
- Selalu dienkripsi - Amazon Timestream LiveAnalytics untuk memastikan bahwa data deret waktu Anda selalu dienkripsi, baik saat istirahat atau dalam perjalanan. Amazon Timestream for LiveAnalytics juga memungkinkan Anda menentukan kunci terkelola AWS KMS pelanggan (CMK) untuk mengenkripsi data di penyimpanan magnetik.
- Ketersediaan tinggi - Amazon Timestream memastikan ketersediaan tinggi permintaan tulis dan baca Anda dengan secara otomatis mereplikasi data dan mengalokasikan sumber daya di setidaknya 3 Availability Zone yang berbeda dalam satu Wilayah. AWS Untuk informasi selengkapnya, lihat Perjanjian Tingkat Layanan Timestream.
- Daya Tahan - Amazon Timestream memastikan daya tahan data Anda dengan secara otomatis mereplikasi memori dan data penyimpanan magnetik Anda di berbagai Availability Zone dalam satu Wilayah. AWS Semua data Anda ditulis ke disk sebelum mengakui permintaan tulis Anda sebagai lengkap.

Timestream untuk kasus LiveAnalytics penggunaan

Contoh daftar kasus penggunaan yang terus bertambah untuk Timestream untuk LiveAnalytics meliputi:

- Memantau metrik untuk meningkatkan kinerja dan ketersediaan aplikasi Anda.
- Penyimpanan dan analisis telemetri industri untuk merampingkan manajemen dan pemeliharaan peralatan.
- Melacak interaksi pengguna dengan aplikasi dari waktu ke waktu.
- Penyimpanan dan analisis data sensor IoT.

Memulai Timestream untuk LiveAnalytics

Kami menyarankan Anda untuk memulai dengan membaca bagian berikut:

- Tutorial- Untuk membuat database yang diisi dengan kumpulan data sampel dan menjalankan kueri sampel.
- Amazon Timestream untuk konsep LiveAnalytics - Untuk mempelajari Timestream penting untuk LiveAnalytics konsep.
- Mengakses Timestream untuk LiveAnalytics- Untuk mempelajari cara mengakses Timestream untuk LiveAnalytics menggunakan konsol, AWS CLI, atauAPI.
- Kuota- Untuk mempelajari tentang kuota pada jumlah Timestream untuk LiveAnalytics komponen yang dapat Anda berikan.

Untuk mempelajari cara cepat mulai mengembangkan aplikasi untuk Timestream LiveAnalytics, lihat berikut ini:

- Menggunakan AWS SDKs
- Referensi bahasa kueri

Cara kerjanya

Bagian berikut memberikan ikhtisar tentang Amazon Timestream untuk komponen layanan Live Analytics dan bagaimana mereka berinteraksi.

Setelah membaca pendahuluan ini, lihat Mengakses Timestream untuk LiveAnalytics bagian untuk mempelajari cara mengakses Timestream for Live Analytics menggunakan konsol AWS CLI, atauSDKs.

Topik

- Amazon Timestream untuk konsep LiveAnalytics
- Arsitektur
- Menulis
- Penyimpanan
- Kueri

- Pertanyaan terjadwal
- Unit Komputasi Aliran Waktu () TCU

Amazon Timestream untuk konsep LiveAnalytics

Data deret waktu adalah urutan titik data yang direkam selama interval waktu. Jenis data ini digunakan untuk mengukur peristiwa yang berubah seiring waktu. Contohnya meliputi hal berikut.

- Harga saham dari waktu ke waktu
- Pengukuran suhu dari waktu ke waktu
- CPUpemanfaatan sebuah EC2 instance dari waktu ke waktu

Dengan data deret waktu, setiap titik data terdiri dari stempel waktu, satu atau lebih atribut, dan peristiwa yang berubah seiring waktu. Data ini dapat digunakan untuk memperoleh wawasan tentang kinerja dan kesehatan aplikasi, mendeteksi anomali, dan mengidentifikasi peluang optimasi. Misalnya, DevOps insinyur mungkin ingin melihat data yang mengukur perubahan dalam metrik kinerja infrastruktur. Produsen mungkin ingin melacak data sensor IoT yang mengukur perubahan peralatan di seluruh fasilitas. Pemasar online mungkin ingin menganalisis data clickstream yang menangkap bagaimana pengguna menavigasi situs web dari waktu ke waktu. Karena data deret waktu dihasilkan dari berbagai sumber dalam volume yang sangat tinggi, perlu dikumpulkan secara hemat biaya dalam waktu dekat, dan oleh karena itu memerlukan penyimpanan yang efisien yang membantu mengatur dan menganalisis data.

Berikut ini adalah konsep kunci Timestream untuk LiveAnalytics.

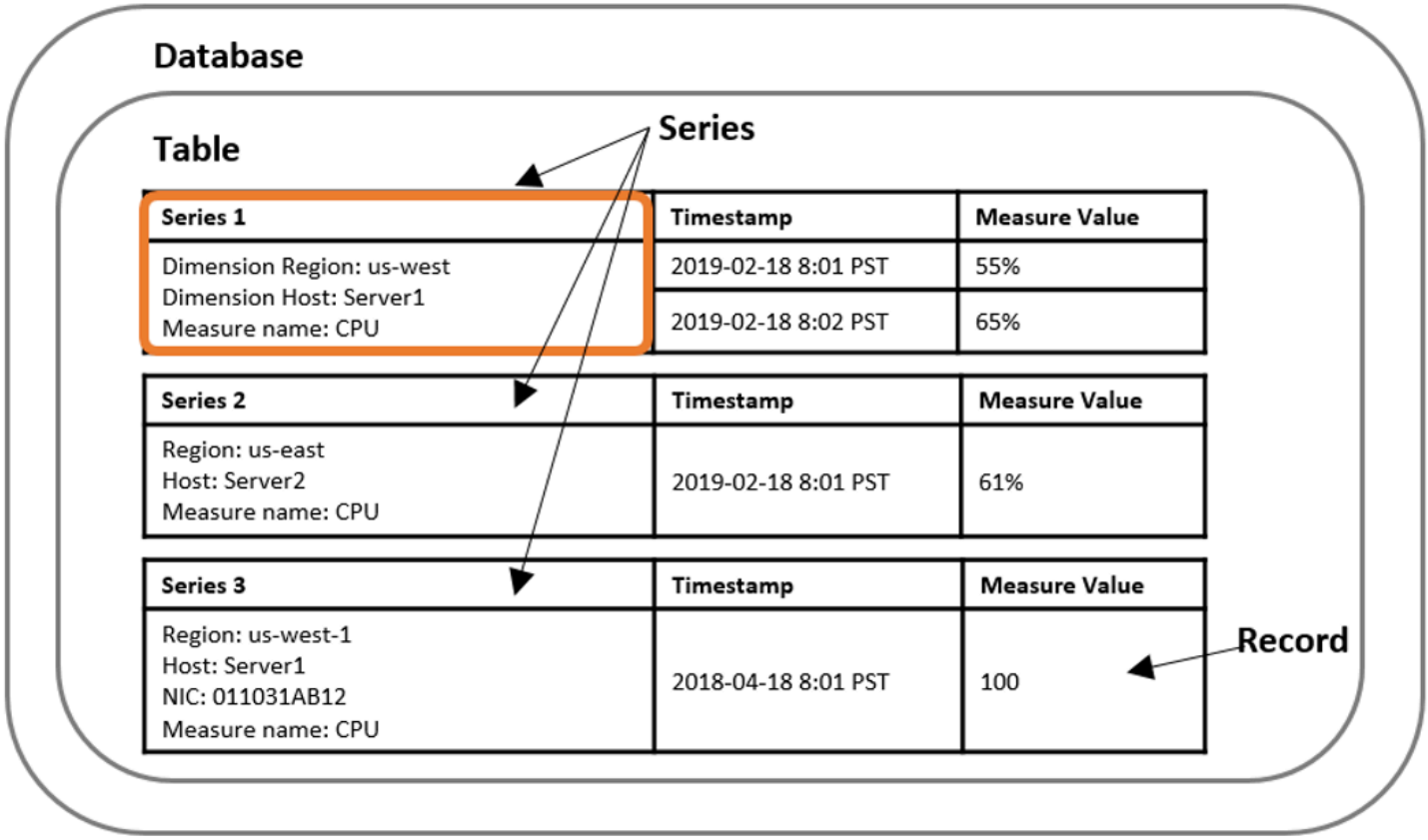
- Deret waktu - Urutan satu atau lebih titik data (atau catatan) yang direkam selama interval waktu. Contohnya adalah harga stok dari waktu ke waktu, CPU atau pemanfaatan memori suatu EC2 instance dari waktu ke waktu, dan pembacaan suhu/tekanan sensor IoT dari waktu ke waktu.
- Rekam - Titik data tunggal dalam deret waktu.
- Dimensi - Atribut yang menggambarkan meta-data dari deret waktu. Dimensi terdiri dari nama dimensi dan nilai dimensi. Pertimbangkan contoh berikut:
  - Ketika mempertimbangkan bursa saham sebagai dimensi, nama dimensinya adalah “bursa saham” dan nilai dimensinya adalah "NYSE”
  - Saat mempertimbangkan AWS Wilayah sebagai dimensi, nama dimensinya adalah “wilayah” dan nilai dimensinya adalah “us-timur-1"

  - Untuk sensor IoT, nama dimensinya adalah “ID perangkat” dan nilai dimensinya adalah “12345"
- Ukur - Nilai aktual yang diukur dengan catatan. Contohnya adalah harga saham, pemanfaatan memori, dan pembacaan suhu atau kelembaban. CPU Ukuran terdiri dari nama ukuran dan nilai ukur. Pertimbangkan contoh berikut:
  - Untuk harga saham, nama ukurannya adalah “harga saham” dan nilai ukurannya adalah harga saham aktual pada suatu titik waktu.
  - Untuk CPU pemanfaatan, nama ukurannya adalah “CPUpemanfaatan” dan nilai ukuran adalah pemanfaatan yang sebenarnyaCPU.

  Ukuran dapat dimodelkan dalam Timestream LiveAnalytics sebagai catatan multi-ukuran atau ukuran tunggal. Untuk informasi selengkapnya, lihat Catatan multi-ukuran vs. catatan ukuran tunggal.
- Timestamp - Menunjukkan kapan ukuran dikumpulkan untuk catatan tertentu. Timestream untuk LiveAnalytics mendukung stempel waktu dengan granularitas nanodetik.
- Tabel - Wadah untuk satu set deret waktu terkait.
- Database - Sebuah wadah tingkat atas untuk tabel.

Ringkasan Timestream untuk konsep LiveAnalytics

Database berisi 0 atau lebih tabel. Setiap tabel berisi 0 atau lebih deret waktu. Setiap deret waktu terdiri dari urutan catatan selama interval waktu tertentu pada perincian tertentu. Setiap deret waktu dapat dijelaskan menggunakan meta-data atau dimensi, data atau ukurannya, dan stempel waktunya.

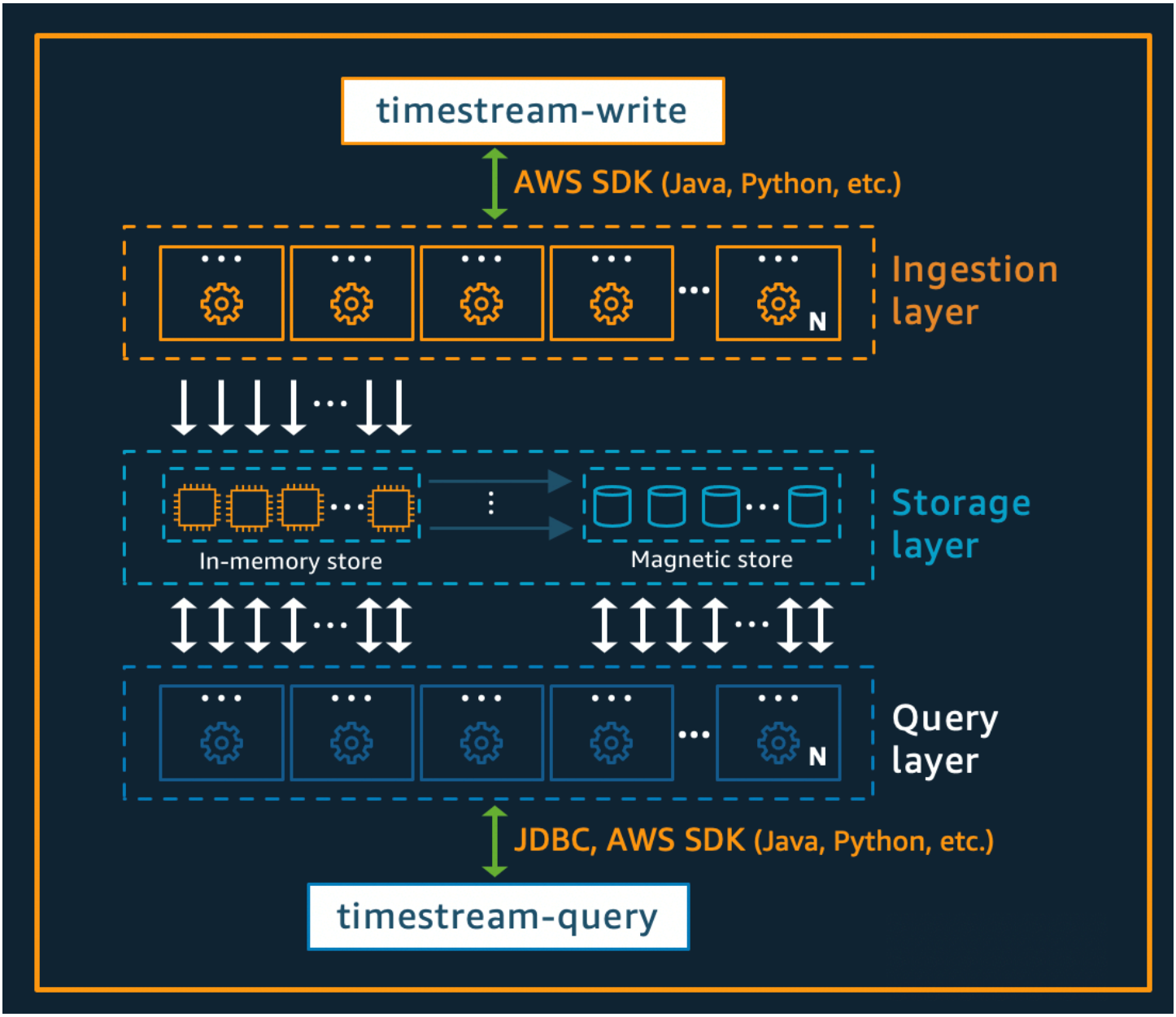


Arsitektur

Amazon Timestream untuk Live Analytics telah dirancang dari bawah ke atas untuk mengumpulkan, menyimpan, dan memproses data deret waktu dalam skala besar. Arsitektur tanpa servernya mendukung penyerapan data, penyimpanan, dan sistem pemrosesan kueri yang sepenuhnya dipisahkan yang dapat diskalakan secara independen. Desain ini menyederhanakan setiap subsistem, membuatnya lebih mudah untuk mencapai keandalan yang tak tergoyahkan, menghilangkan hambatan penskalaan, dan mengurangi kemungkinan kegagalan sistem yang berkorelasi. Masing-masing faktor ini menjadi lebih penting seiring dengan skala sistem.

Topik

- Tulis arsitektur
- Arsitektur penyimpanan
- Arsitektur kueri
- Arsitektur seluler



Tulis arsitektur

Saat menulis data deret waktu, rute Amazon Timestream untuk Live Analytics menulis untuk tabel, partisi, ke instance penyimpanan memori toleran kesalahan yang memproses penulisan data throughput tinggi. Penyimpanan memori pada gilirannya mencapai daya tahan dalam sistem penyimpanan terpisah yang mereplikasi data di tiga Availability Zones ()AZs. Replikasi berbasis kuorum sehingga hilangnya node, atau seluruh AZ, tidak akan mengganggu ketersediaan penulisan. Dalam waktu dekat, node penyimpanan dalam memori lainnya disinkronkan ke data untuk melayani kueri. Node replika pembaca AZs juga menjangkau, untuk memastikan ketersediaan baca yang tinggi.

Timestream untuk Live Analytics mendukung penulisan data langsung ke penyimpanan magnetik, untuk aplikasi yang menghasilkan data yang datang terlambat dengan throughput yang lebih rendah. Data yang datang terlambat adalah data dengan stempel waktu lebih awal dari waktu saat ini. Mirip dengan throughput tinggi yang ditulis di penyimpanan memori, data yang ditulis ke dalam penyimpanan magnetik direplikasi di tiga AZs dan replikasi berbasis kuorum.

Baik data ditulis ke memori atau penyimpanan magnetik, Timestream for Live Analytics secara otomatis mengindeks dan mempartisi data sebelum menulisnya ke penyimpanan. Satu tabel Timestream untuk Live Analytics mungkin memiliki ratusan, ribuan, atau bahkan jutaan partisi. Partisi individu tidak, secara langsung, berkomunikasi satu sama lain dan tidak berbagi data apa pun (arsitektur tanpa berbagi). Sebaliknya, partisi tabel dilacak melalui layanan pelacakan dan pengindeksan partisi yang sangat tersedia. Ini memberikan pemisahan masalah lain yang dirancang khusus untuk meminimalkan efek kegagalan dalam sistem dan membuat kegagalan yang berkorelasi jauh lebih kecil kemungkinannya.

Arsitektur penyimpanan

Ketika data disimpan dalam Timestream untuk Live Analytics, data diatur dalam urutan waktu serta sepanjang waktu berdasarkan atribut konteks yang ditulis dengan data. Memiliki skema partisi yang membagi “ruang” selain waktu penting untuk menskalakan sistem deret waktu secara besar-besaran. Ini karena sebagian besar data deret waktu ditulis pada atau sekitar waktu saat ini. Akibatnya, partisi berdasarkan waktu saja tidak melakukan pekerjaan yang baik dalam mendistribusikan lalu lintas tulis atau memungkinkan pemangkasan data yang efektif pada waktu kueri. Ini penting untuk pemrosesan deret waktu skala ekstrim, dan telah memungkinkan Timestream untuk Live Analytics untuk menskalakan urutan besarnya lebih tinggi daripada sistem terkemuka lainnya di luar sana saat ini dengan cara tanpa server. Partisi yang dihasilkan disebut sebagai “ubin” karena mereka mewakili divisi dari ruang dua dimensi (yang dirancang untuk menjadi ukuran yang sama). Timestream untuk tabel Live Analytics dimulai sebagai partisi tunggal (ubin), dan kemudian dibagi dalam dimensi spasial sesuai kebutuhan throughput. Ketika ubin mencapai ukuran tertentu, mereka kemudian membelah dalam dimensi waktu untuk mencapai paralelisme baca yang lebih baik saat ukuran data tumbuh.

Timestream for Live Analytics dirancang untuk mengelola siklus hidup data deret waktu secara otomatis. Timestream for Live Analytics menawarkan dua penyimpanan data—penyimpanan dalam memori dan penyimpanan magnetik yang hemat biaya. Ini juga mendukung konfigurasi kebijakan tingkat tabel untuk secara otomatis mentransfer data di seluruh toko. Data throughput tinggi yang masuk menulis mendarat di penyimpanan memori tempat data dioptimalkan untuk penulisan, serta pembacaan yang dilakukan sekitar waktu saat ini untuk memberi daya pada dasbor dan mengingatkan kueri jenis. Ketika kerangka waktu utama untuk kebutuhan penulisan, peringatan,

dan dasbor telah berlalu, memungkinkan data mengalir secara otomatis dari penyimpanan memori ke penyimpanan magnetik untuk mengoptimalkan biaya. Timestream untuk Live Analytics memungkinkan pengaturan kebijakan penyimpanan data di penyimpanan memori untuk tujuan ini. Data yang ditulis untuk data yang datang terlambat secara langsung ditulis ke dalam penyimpanan magnetik.

Setelah data tersedia di penyimpanan magnetik (karena berakhirnya periode penyimpanan penyimpanan memori atau karena penulisan langsung ke penyimpanan magnetik), itu direorganisasi ke dalam format yang sangat dioptimalkan untuk pembacaan data volume besar. Penyimpanan magnetik juga memiliki kebijakan retensi data yang dapat dikonfigurasi jika ada ambang waktu di mana data melebihi kegunaannya. Ketika data melebihi rentang waktu yang ditentukan untuk kebijakan penyimpanan penyimpanan magnetik, data akan dihapus secara otomatis. Oleh karena itu, dengan Timestream untuk Live Analytics, selain beberapa konfigurasi, manajemen siklus hidup data terjadi dengan mulus di belakang layar.

Arsitektur kueri

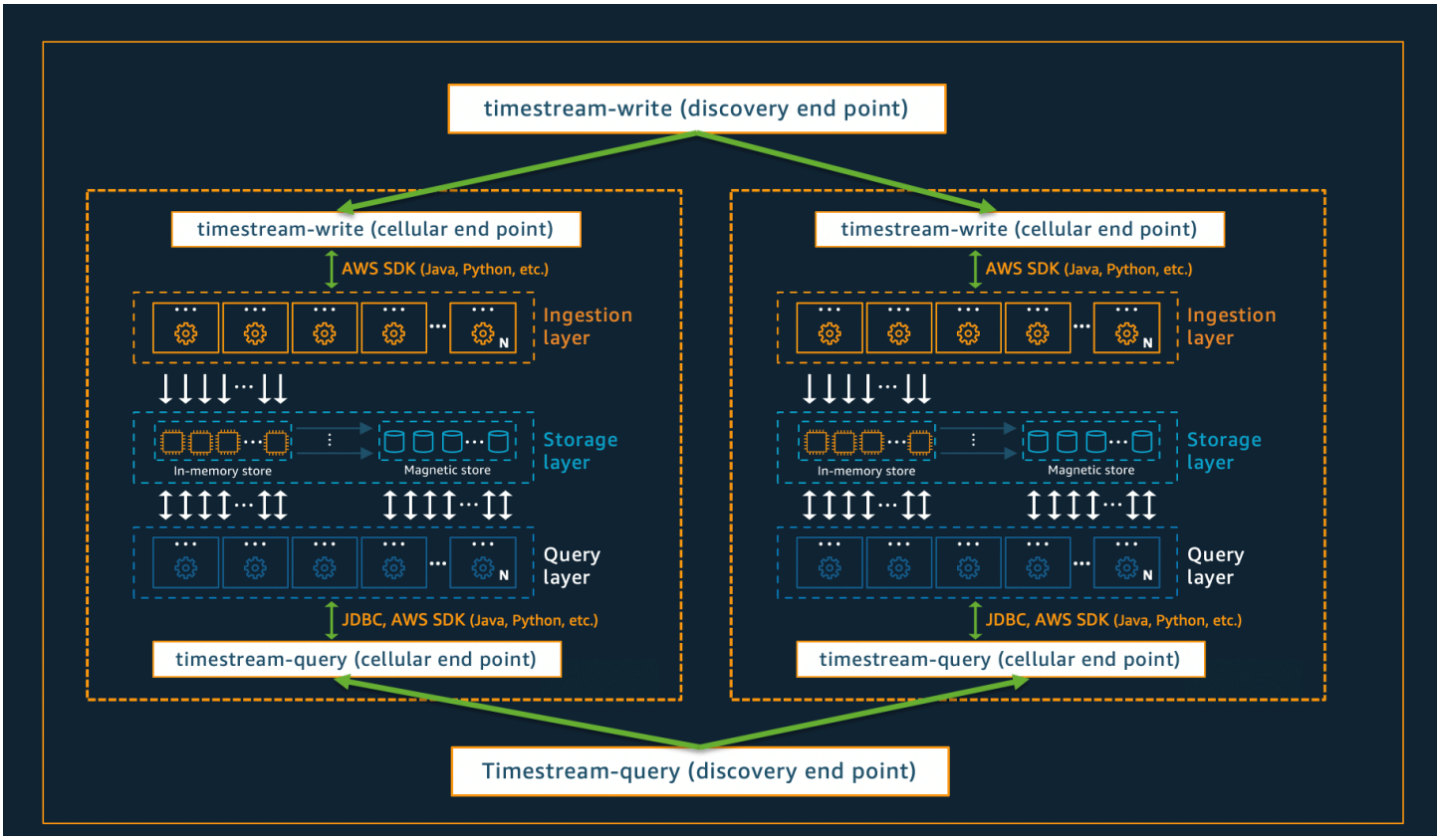
Kueri Timestream untuk Live Analytics dinyatakan dalam SQL tata bahasa yang memiliki ekstensi untuk dukungan khusus seri waktu (tipe dan fungsi data spesifik seri waktu), sehingga kurva pembelajaran mudah bagi pengembang yang sudah terbiasa. SQL Kueri kemudian diproses oleh mesin kueri terdistribusi adaptif yang menggunakan metadata dari layanan pelacakan dan pengindeksan ubin untuk mengakses dan menggabungkan data dengan mulus di seluruh penyimpanan data pada saat kueri dikeluarkan. Ini membuat pengalaman yang beresonansi dengan baik dengan pelanggan karena meruntuhkan banyak kompleksitas Rube Goldberg menjadi abstraksi database yang sederhana dan akrab.

Kueri dijalankan oleh armada pekerja khusus di mana jumlah pekerja yang terdaftar untuk menjalankan kueri tertentu ditentukan oleh kompleksitas kueri dan ukuran data. Kinerja untuk kueri kompleks pada kumpulan data besar dicapai melalui paralelisme masif, baik pada armada runtime kueri maupun armada penyimpanan sistem. Kemampuan untuk menganalisis sejumlah besar data dengan cepat dan efisien adalah salah satu kekuatan terbesar Timestream untuk Live Analytics. Satu kueri yang berjalan di atas terabyte atau bahkan petabyte data mungkin memiliki ribuan mesin yang bekerja pada semuanya pada saat yang bersamaan.

Arsitektur seluler

Untuk memastikan bahwa Timestream for Live Analytics dapat menawarkan skala hampir tak terbatas untuk aplikasi Anda, sekaligus memastikan ketersediaan 99,99%, sistem ini juga dirancang menggunakan arsitektur seluler. Alih-alih menskalakan sistem secara keseluruhan, Timestream

untuk Live Analytics segmen menjadi beberapa salinan yang lebih kecil dari dirinya sendiri, disebut sebagai sel. Hal ini memungkinkan sel untuk diuji pada skala penuh, dan mencegah masalah sistem dalam satu sel mempengaruhi aktivitas di sel lain di wilayah tertentu. Sementara Timestream untuk Live Analytics dirancang untuk mendukung beberapa sel per wilayah, pertimbangkan skenario fiktif berikut, di mana ada 2 sel di suatu wilayah.



Dalam skenario yang digambarkan di atas, konsumsi data dan permintaan kueri pertama kali diproses oleh titik akhir penemuan untuk konsumsi data dan kueri, masing-masing. Kemudian, titik akhir penemuan mengidentifikasi sel yang berisi data pelanggan, dan mengarahkan permintaan ke titik akhir konsumsi atau kueri yang sesuai untuk sel tersebut. Saat menggunakanSDKs, tugas manajemen titik akhir ini ditangani secara transparan untuk Anda.

> **Note**
>
> Saat menggunakan VPC titik akhir dengan Timestream untuk Live Analytics atau langsung mengakses REST API operasi untuk Timestream for Live Analytics, Anda harus berinteraksi langsung dengan titik akhir seluler. Untuk panduan tentang cara melakukannya, lihat VPCTitik Akhir untuk petunjuk tentang cara menyiapkan titik VPC akhir, dan Pola Penemuan Titik Akhir untuk instruksi tentang pemanggilan langsung operasi. REST API

Menulis

Anda dapat mengumpulkan data deret waktu dari perangkat yang terhubung, sistem TI, dan peralatan industri, dan menuliskannya ke Timestream untuk Live Analytics. Timestream for Live Analytics memungkinkan Anda menulis titik data dari satu deret waktu dan/atau titik data dari banyak seri dalam satu permintaan tulis ketika deret waktu termasuk dalam tabel yang sama. Demi kenyamanan Anda, Timestream for Live Analytics menawarkan skema fleksibel yang secara otomatis mendeteksi nama kolom dan tipe data untuk tabel Timestream untuk Live Analytics Anda berdasarkan nama dimensi dan tipe data dari nilai ukuran yang Anda tentukan saat menjalankan penulisan ke dalam database. Anda juga dapat menulis kumpulan data ke Timestream untuk Live Analytics.

> **Note**
>
> Timestream untuk Live Analytics mendukung semantik konsistensi akhir untuk pembacaan. Ini berarti bahwa ketika Anda melakukan kueri data segera setelah menulis kumpulan data ke Timestream for Live Analytics, hasil kueri mungkin tidak mencerminkan hasil operasi penulisan yang baru saja selesai. Hasilnya mungkin juga mencakup beberapa data basi. Demikian pula, saat menulis data deret waktu dengan satu atau lebih dimensi baru, kueri dapat mengembalikan sebagian bagian kolom untuk waktu yang singkat. Jika Anda mengulangi permintaan kueri ini setelah waktu yang singkat, hasilnya akan mengembalikan data terbaru.

Anda dapat menulis data menggunakan AWS SDKs,, AWS CLI, atau melalui AWS LambdaAWS IoT Core,Layanan Terkelola Amazon untuk Apache Flink,Amazon Kinesis,Amazon MSK, danTelegraf sumber terbuka.

Topik

Jenis data

Timestream untuk Live Analytics mendukung tipe data berikut untuk penulisan.

Tipe data	Deskripsi
BIGINT	Merupakan integer bertanda 64-bit.
BOOLEAN	Merupakan dua nilai kebenaran logika, yaitu benar, dan salah.
DOUBLE	Presisi variabel 64-bit mengimplementasikan IEEE Standard 754 untuk Binary Floating-Point Arithmetic. **Note** Ada fungsi bahasa query untuk `Infinity` dan nilai `NaN` ganda yang dapat digunakan dalam query. Tetapi Anda tidak dapat menulis nilai-nilai itu ke Timestream.
VARCHAR	Data karakter panjang variabel dengan panjang maksimum opsional. Batas maksimum adalah 2 KB.
MULTI	Tipe data untuk catatan multi-ukuran. Tipe data ini mencakup satu atau lebih ukuran tipe`BIGINT`,`BOOLEAN`,`DOUBLE`,`VARCHAR`, dan`TIMESTAMP` .
TIMESTAMP	Merupakan instance dalam waktu menggunakan waktu presisi nanodetikUTC, melacak waktu sejak waktu Unix. Tipe data ini saat ini hanya didukung untuk catatan multi-ukuran (yaitu dalam nilai ukuran tipe`MULTI`). *YYYY*-*MM*-*DD* *hh*:*mm*:*ss*.*sssssssss* Menulis stempel waktu dukungan dalam kisaran `1970-01-01 00:00:00.000000000` ke. `2262-04-11 23:47:16.854775807`

Tidak ada definisi skema di muka

Sebelum mengirim data ke Amazon Timestream untuk Live Analytics, Anda harus membuat database dan tabel menggunakan AWS Management Console, Timestream untuk Live AnalyticsSDKs, atau Timestream untuk operasi Live Analytics. API Untuk informasi selengkapnya, silakan lihat Buat database dan Membuat tabel. Saat membuat tabel, Anda tidak perlu mendefinisikan skema di depan. Amazon Timestream for Live Analytics secara otomatis mendeteksi skema berdasarkan ukuran dan dimensi titik data yang dikirim, sehingga Anda tidak perlu lagi mengubah skema secara offline untuk menyesuaikannya dengan data deret waktu yang berubah dengan cepat.

Menulis data (sisipan dan upserts)

Operasi tulis di Amazon Timestream untuk Live Analytics memungkinkan Anda memasukkan dan meningkatkan data. Secara default, menulis di Amazon Timestream untuk Live Analytics mengikuti penulis pertama memenangkan semantik, di mana data disimpan sebagai append saja dan catatan duplikat ditolak. Sementara penulis pertama memenangkan semantik memenuhi persyaratan banyak aplikasi deret waktu, ada skenario di mana aplikasi perlu memperbarui catatan yang ada dengan cara yang idempoten dan/atau menulis data dengan penulis terakhir memenangkan semantik, di mana catatan dengan versi tertinggi disimpan dalam layanan. Untuk mengatasi skenario ini, Amazon Timestream untuk Live Analytics menyediakan kemampuan untuk meningkatkan data. Upsert adalah operasi yang menyisipkan catatan ke dalam sistem ketika catatan tidak ada, atau memperbarui catatan ketika ada. Ketika catatan diperbarui, itu diperbarui dengan cara yang idempoten.

Tidak ada operasi tingkat catatan untuk penghapusan. Tetapi tabel dan database dapat dihapus.

Menulis data ke dalam penyimpanan memori dan penyimpanan magnetik

Amazon Timestream untuk Live Analytics menawarkan kemampuan untuk langsung menulis data ke penyimpanan memori dan penyimpanan magnetik. Penyimpanan memori dioptimalkan untuk penulisan data throughput tinggi dan penyimpanan magnetik dioptimalkan untuk penulisan throughput yang lebih rendah dari data kedatangan terlambat.

Data yang datang terlambat adalah data dengan stempel waktu lebih awal dari waktu saat ini dan di luar periode penyimpanan penyimpanan memori. Anda harus secara eksplisit mengaktifkan kemampuan untuk menulis data yang datang terlambat ke dalam penyimpanan magnetik dengan mengaktifkan penulisan penyimpanan magnetik untuk tabel. Juga, `MagneticStoreRejectedDataLocation` didefinisikan ketika tabel dibuat. Untuk menulis ke penyimpanan magnetik, penelepon `WriteRecords` harus memiliki `S3:PutObject` izin ke bucket

S3 yang ditentukan `MagneticStoreRejectedDataLocation` selama pembuatan tabel. Untuk informasi lebih lanjut, lihat CreateTable, WriteRecords, dan PutObject.

Menulis data dengan catatan ukuran tunggal dan catatan multi-ukuran

Amazon Timestream for Live Analytics menawarkan kemampuan untuk menulis data menggunakan dua jenis catatan, yaitu rekaman ukuran tunggal dan catatan multi-ukuran.

Catatan ukuran tunggal

Catatan ukuran tunggal memungkinkan Anda mengirim satu ukuran per catatan. Saat data dikirim ke Timestream untuk Live Analytics menggunakan format ini, Timestream untuk Live Analytics membuat satu baris tabel per rekaman. Ini berarti bahwa jika perangkat memancarkan 4 metrik dan setiap metrik dikirim sebagai catatan ukuran tunggal, Timestream untuk Live Analytics akan membuat 4 baris dalam tabel untuk menyimpan data ini, dan atribut perangkat akan diulang untuk setiap baris. Format ini direkomendasikan jika Anda ingin memantau satu metrik dari aplikasi atau saat aplikasi Anda tidak memancarkan beberapa metrik secara bersamaan.

Catatan multi-ukuran

Dengan catatan multi-ukuran, Anda dapat menyimpan beberapa ukuran dalam satu baris tabel, alih-alih menyimpan satu ukuran per baris tabel. Oleh karena itu, catatan multi-ukuran memungkinkan Anda untuk memigrasikan data yang ada dari database relasional ke Amazon Timestream untuk Live Analytics dengan sedikit perubahan.

Anda juga dapat mengumpulkan lebih banyak data dalam satu permintaan penulisan daripada catatan ukuran tunggal. Ini meningkatkan throughput dan kinerja penulisan data, dan juga mengurangi biaya penulisan data. Ini karena mengumpulkan lebih banyak data dalam permintaan tulis memungkinkan Amazon TimeStream untuk Live Analytics mengidentifikasi lebih banyak data yang dapat diulang dalam satu permintaan tulis (jika berlaku), dan hanya mengenakan biaya sekali untuk data berulang.

Topik

- Catatan multi-ukuran
- Menulis data dengan stempel waktu yang ada di masa lalu atau di masa depan

Catatan multi-ukuran

Dengan catatan multi-ukuran, Anda dapat menyimpan data deret waktu Anda dalam format yang lebih ringkas di memori dan penyimpanan magnetik, yang membantu menurunkan biaya

penyimpanan data. Selain itu, penyimpanan data ringkas cocok untuk menulis kueri yang lebih sederhana untuk pengambilan data, meningkatkan kinerja kueri, dan menurunkan biaya kueri.

Selain itu, catatan multi-ukuran juga mendukung tipe TIMESTAMP data untuk menyimpan lebih dari satu stempel waktu dalam catatan deret waktu. TIMESTAMPatribut dalam catatan multi-ukuran mendukung stempel waktu di masa depan atau masa lalu. Oleh karena itu, catatan multi-ukuran membantu meningkatkan kinerja, biaya, dan kesederhanaan kueri—dan menawarkan lebih banyak fleksibilitas untuk menyimpan berbagai jenis ukuran yang berkorelasi.

Manfaat

Berikut ini adalah manfaat menggunakan catatan multi-ukuran.

- Kinerja dan biaya - Catatan multi-ukuran memungkinkan Anda menulis beberapa ukuran deret waktu dalam satu permintaan tulis. Ini meningkatkan throughput penulisan dan juga mengurangi biaya penulisan. Dengan catatan multi-ukuran, Anda dapat menyimpan data dengan cara yang lebih ringkas, yang membantu menurunkan biaya penyimpanan data. Penyimpanan data ringkas dari catatan multi-ukuran menghasilkan lebih sedikit data yang diproses oleh kueri. Ini dirancang untuk meningkatkan kinerja kueri secara keseluruhan dan membantu menurunkan biaya kueri.
- Kesederhanaan kueri — Dengan catatan multi-ukuran, Anda tidak perlu menulis ekspresi tabel umum yang kompleks (CTEs) dalam kueri untuk membaca beberapa ukuran dengan stempel waktu yang sama. Ini karena ukuran disimpan sebagai kolom dalam satu baris tabel. Oleh karena itu, catatan multi-ukuran memungkinkan penulisan kueri yang lebih sederhana.
- Fleksibilitas pemodelan data — Anda dapat menulis stempel waktu masa depan ke Timestream for Live Analytics dengan menggunakan tipe TIMESTAMP data dan catatan multi-ukuran. Catatan multi-ukuran dapat memiliki beberapa atribut tipe TIMESTAMP data, selain bidang waktu dalam catatan. TIMESTAMPatribut, dalam catatan multi-ukuran, dapat memiliki stempel waktu di masa depan atau masa lalu dan berperilaku seperti bidang waktu kecuali Timestream untuk Live Analytics tidak mengindeks nilai tipe TIMESTAMP dalam catatan multi-ukuran.

Kasus penggunaan

Anda dapat menggunakan catatan multi-ukuran untuk aplikasi deret waktu apa pun yang menghasilkan lebih dari satu pengukuran dari perangkat yang sama pada waktu tertentu. Berikut ini adalah beberapa contoh aplikasi.

- Platform streaming video yang menghasilkan ratusan metrik pada waktu tertentu.

- Perangkat medis yang menghasilkan pengukuran seperti kadar oksigen darah, detak jantung, dan denyut nadi.
- Peralatan industri seperti rig minyak yang menghasilkan metrik, suhu, dan sensor cuaca.
- Aplikasi lain yang dirancang dengan satu atau lebih layanan mikro.

Contoh: Memantau kinerja dan kesehatan aplikasi streaming video

Pertimbangkan aplikasi streaming video yang berjalan pada 200 EC2 instance. Anda ingin menggunakan Amazon Timestream for Live Analytics untuk menyimpan dan menganalisis metrik yang dipancarkan dari aplikasi, sehingga Anda dapat memahami kinerja dan kesehatan aplikasi Anda, mengidentifikasi anomali dengan cepat, menyelesaikan masalah, dan menemukan peluang pengoptimalan.

Kami akan memodelkan skenario ini dengan catatan ukuran tunggal dan catatan multi-ukuran, dan kemudian membandingkan/membedakan kedua pendekatan. Untuk setiap pendekatan, kami membuat asumsi berikut.

- Setiap EC2 instance memancarkan empat ukuran (video_startup_time, rebuffering_ratio, video_playback_failure, dan average_frame_rate) dan empat dimensi (device_id, device_type, os_version, dan region) per detik.
- Anda ingin menyimpan 6 jam data di penyimpanan memori dan 6 bulan data di penyimpanan magnetik.
- Untuk mengidentifikasi anomali, Anda telah menyiapkan 10 kueri yang berjalan setiap menit untuk mengidentifikasi aktivitas yang tidak biasa selama beberapa menit terakhir. Anda juga telah membangun dasbor dengan delapan widget yang menampilkan data 6 jam terakhir, sehingga Anda dapat memantau aplikasi Anda secara efektif. Dasbor ini diakses oleh lima pengguna pada waktu tertentu dan disegarkan secara otomatis setiap jam.

Menggunakan catatan ukuran tunggal

Pemodelan data: Dengan catatan ukuran tunggal, kami akan membuat satu catatan untuk masing-masing dari empat ukuran (waktu startup video, rasio rebuffering, kegagalan pemutaran video, dan kecepatan bingkai rata-rata). Setiap record akan memiliki empat dimensi (device_id, device_type, os_version, dan region) dan stempel waktu.

Menulis: Saat Anda menulis data ke Amazon Timestream untuk Live Analytics, catatan dibuat sebagai berikut.

```
public void writeRecords() {
    System.out.println("Writing records");
    // Specify repeated values for all records
    List<Record> records = new ArrayList<>();
    final long time = System.currentTimeMillis();

    List<Dimension> dimensions = new ArrayList<>();

    final Dimension device_id = new
 Dimension().withName("device_id").withValue("12345678");
    final Dimension device_type = new
 Dimension().withName("device_type").withValue("iPhone 11");
    final Dimension os_version = new
 Dimension().withName("os_version").withValue("14.8");
    final Dimension region = new Dimension().withName("region").withValue("us-east-1");

    dimensions.add(device_id);
    dimensions.add(device_type);
    dimensions.add(os_version);
    dimensions.add(region);

    Record videoStartupTime = new Record()
        .withDimensions(dimensions)
        .withMeasureName("video_startup_time")
        .withMeasureValue("200")
        .withMeasureValueType(MeasureValueType.BIGINT)
        .withTime(String.valueOf(time));
    Record rebufferingRatio = new Record()
        .withDimensions(dimensions)
        .withMeasureName("rebuffering_ratio")
        .withMeasureValue("0.5")
        .withMeasureValueType(MeasureValueType.DOUBLE)
        .withTime(String.valueOf(time));
    Record videoPlaybackFailures = new Record()
        .withDimensions(dimensions)
        .withMeasureName("video_playback_failures")
        .withMeasureValue("0")
        .withMeasureValueType(MeasureValueType.BIGINT)
        .withTime(String.valueOf(time));
    Record averageFrameRate = new Record()
        .withDimensions(dimensions)
        .withMeasureName("average_frame_rate")
        .withMeasureValue("0.5")
```

```
        .withMeasureValueType(MeasureValueType.DOUBLE)
        .withTime(String.valueOf(time));

    records.add(videoStartupTime);
    records.add(rebufferingRatio);
    records.add(videoPlaybackFailures);
    records.add(averageFrameRate);

    WriteRecordsRequest writeRecordsRequest = new WriteRecordsRequest()
        .withDatabaseName(DATABASE_NAME)
        .withTableName(TABLE_NAME)
        .withRecords(records);

    try {
      WriteRecordsResult writeRecordsResult =
amazonTimestreamWrite.writeRecords(writeRecordsRequest);
      System.out.println("WriteRecords Status: " +
writeRecordsResult.getSdkHttpMetadata().getHttpStatusCode());
    } catch (RejectedRecordsException e) {
      System.out.println("RejectedRecords: " + e);
      for (RejectedRecord rejectedRecord : e.getRejectedRecords()) {
        System.out.println("Rejected Index " + rejectedRecord.getRecordIndex() + ": "
            + rejectedRecord.getReason());
      }
      System.out.println("Other records were written successfully. ");
    } catch (Exception e) {
      System.out.println("Error: " + e);
    }
  }
```

Saat Anda menyimpan catatan ukuran tunggal, data secara logis direpresentasikan sebagai berikut.

Waktu	device_id	device_type	os_versi	region	ukuran_nama	ukuran_nilai: :bigint	ukuran_nilai: :ganda
2021-09-07 21:48:44 .00	12345678	iPhone 11	14.8	us-east-1	video_startup_waktu	200	
2021-09-07	12345678	iPhone 11	14.8	us-east-1	rebuffering_ratio		0,5

Waktu	device_id	device_type	os_versi	region	ukuran_nama	ukuran_nilai: :bigint	ukuran_nilai: :ganda
21:48:44 .00							
2021-09-07 21:48:44 .00	12345678	iPhone 11	14.8	us-east-1	video_playback_failure	0	
2021-09-07 21:48:44 .00	12345678	iPhone 11	14.8	us-east-1	rata-rata_frame_rate		0,85
2021-09-07 21:53:44 .00	12345678	iPhone 11	14.8	us-east-1	video_startup_waktu	500	
2021-09-07 21:53:44 .00	12345678	iPhone 11	14.8	us-east-1	rebuffering_ratio		1.5
2021-09-07 21:53:44 .00	12345678	iPhone 11	14.8	us-east-1	video_playback_failure	10	
2021-09-07 21:53:44 .00	12345678	iPhone 11	14.8	us-east-1	rata-rata_frame_rate		0.2

Kueri: Anda dapat menulis kueri yang mengambil semua titik data dengan stempel waktu yang sama yang diterima selama 15 menit terakhir sebagai berikut.

```
with cte_video_startup_time as ( SELECT time, device_id, device_type, os_version,
 region, measure_value::bigint as video_startup_time FROM table where time >= ago(15m)
 and measure_name=”video_startup_time”),
cte_rebuffering_ratio as ( SELECT time, device_id, device_type, os_version, region,
 measure_value::double as rebuffering_ratio FROM table where time >= ago(15m) and
 measure_name=”rebuffering_ratio”),
cte_video_playback_failures as ( SELECT time, device_id, device_type, os_version,
 region, measure_value::bigint as video_playback_failures FROM table where time >=
 ago(15m) and measure_name=”video_playback_failures”),
cte_average_frame_rate as ( SELECT time, device_id, device_type, os_version, region,
 measure_value::double as average_frame_rate FROM table where time >= ago(15m) and
 measure_name=”average_frame_rate”)
SELECT a.time, a.device_id, a.os_version, a.region, a.video_startup_time,
 b.rebuffering_ratio, c.video_playback_failures, d.average_frame_rate FROM
 cte_video_startup_time a, cte_buffering_ratio b, cte_video_playback_failures c,
 cte_average_frame_rate d WHERE
a.time = b.time AND a.device_id = b.device_id AND a.os_version = b.os_version AND
 a.region=b.region AND
a.time = c.time AND a.device_id = c.device_id AND a.os_version = c.os_version AND
 a.region=c.region AND
a.time = d.time AND a.device_id = d.device_id AND a.os_version = d.os_version AND
 a.region=d.region
```

Biaya beban kerja: Biaya beban kerja ini diperkirakan $373,23 per bulan dengan catatan ukuran tunggal

Menggunakan catatan multi-ukuran

Pemodelan data: Dengan catatan multi-ukuran, kami akan membuat satu catatan yang berisi keempat ukuran (waktu startup video, rasio rebuffering, kegagalan pemutaran video, dan kecepatan bingkai rata-rata), keempat dimensi (device_id, device_type, os_version, dan region), dan stempel waktu.

Menulis: Saat Anda menulis data ke Amazon Timestream untuk Live Analytics, catatan dibuat sebagai berikut.

```
public void writeRecords() {
    System.out.println("Writing records");
    // Specify repeated values for all records
    List<Record> records = new ArrayList<>();
    final long time = System.currentTimeMillis();
```

```
    List<Dimension> dimensions = new ArrayList<>();

    final Dimension device_id = new
Dimension().withName("device_id").withValue("12345678");
    final Dimension device_type = new
Dimension().withName("device_type").withValue("iPhone 11");
    final Dimension os_version = new
Dimension().withName("os_version").withValue("14.8");
    final Dimension region = new Dimension().withName("region").withValue("us-east-1");

    dimensions.add(device_id);
    dimensions.add(device_type);
    dimensions.add(os_version);
    dimensions.add(region);

    Record videoMetrics = new Record()
        .withDimensions(dimensions)
        .withMeasureName("video_metrics")
        .withTime(String.valueOf(time));
        .withMeasureValueType(MeasureValueType.MULTI)
        .withMeasureValues(
          new MeasureValue()
         .withName("video_startup_time")
         .withValue("0")
         .withValueType(MeasureValueType.BIGINT),
         new MeasureValue()
  .withName("rebuffering_ratio")
         .withValue("0.5")
         .withType(MeasureValueType.DOUBLE),
          new MeasureValue()
         .withName("video_playback_failures")
         .withValue("0")
         .withValueType(MeasureValueType.BIGINT),
   new MeasureValue()
          .withName("average_frame_rate")
         .withValue("0.5")
         .withValueType(MeasureValueType.DOUBLE))

    records.add(videoMetrics);

    WriteRecordsRequest writeRecordsRequest = new WriteRecordsRequest()
        .withDatabaseName(DATABASE_NAME)
        .withTableName(TABLE_NAME)
        .withRecords(records);
```

```
   try {
     WriteRecordsResult writeRecordsResult =
amazonTimestreamWrite.writeRecords(writeRecordsRequest);
     System.out.println("WriteRecords Status: " +
writeRecordsResult.getSdkHttpMetadata().getHttpStatusCode());
   } catch (RejectedRecordsException e) {
     System.out.println("RejectedRecords: " + e);
     for (RejectedRecord rejectedRecord : e.getRejectedRecords()) {
       System.out.println("Rejected Index " + rejectedRecord.getRecordIndex() + ": "
           + rejectedRecord.getReason());
     }
     System.out.println("Other records were written successfully. ");
   } catch (Exception e) {
     System.out.println("Error: " + e);
   }
 }
```

Saat Anda menyimpan catatan multi-ukuran, data secara logis direpresentasikan sebagai berikut.

Waktu	device_id	device_type	os_versi	region	ukuran_rma	video_startup_waktu	rebuffering_ratio	video_pemutara_kegagaln	rata-rata_frame_rate
2021-09-7 21:48:440	1234567	iPhone 11	14.8	us-east-1	video_mrik	200	0,5	0	0,85
2021-09-7 21:53:440	1234567	iPhone 11	14.8	us-east-1	video_mrik	500	1.5	10	0.2

Kueri: Anda dapat menulis kueri yang mengambil semua titik data dengan stempel waktu yang sama yang diterima selama 15 menit terakhir sebagai berikut.

```
SELECT time, device_id, device_type, os_version, region, video_startup_time,
 rebuffering_ratio, video_playback_failures, average_frame_rate FROM table where time
 >= ago(15m)
```

Biaya beban kerja: Biaya beban kerja diperkirakan $127,43 dengan catatan multi-ukuran.

> **Note**
>
> Dalam hal ini, menggunakan catatan multi-ukuran mengurangi perkiraan pengeluaran bulanan keseluruhan sebesar 2,5x, dengan biaya penulisan data berkurang 3,3x, biaya penyimpanan berkurang 3,3x, dan biaya kueri berkurang 1,2x.

Menulis data dengan stempel waktu yang ada di masa lalu atau di masa depan

Timestream for Live Analytics menawarkan kemampuan untuk menulis data dengan stempel waktu yang berada di luar jendela penyimpanan penyimpanan memori melalui beberapa mekanisme berbeda.

- Menulis toko magnetik — Anda dapat menulis data yang datang terlambat langsung ke toko magnetik melalui penulisan penyimpanan magnetik. Untuk menggunakan penulisan penyimpanan magnetik, Anda harus terlebih dahulu mengaktifkan penulisan penyimpanan magnetik untuk sebuah tabel. Anda kemudian dapat menelan data ke dalam tabel menggunakan mekanisme yang sama yang digunakan untuk menulis data ke dalam penyimpanan memori. Amazon Timestream untuk Live Analytics akan secara otomatis menulis data ke penyimpanan magnetik berdasarkan stempel waktunya.

  > **Note**
  >
  > write-to-readLatensi untuk penyimpanan magnetik bisa sampai 6 jam, tidak seperti menulis data ke penyimpanan memori, di mana write-to-read latensi berada dalam kisaran sub-detik.

- TIMESTAMPtipe data untuk pengukuran — Anda dapat menggunakan tipe TIMESTAMP data untuk menyimpan data dari masa lalu, sekarang, atau masa depan. Catatan multi-ukuran dapat memiliki beberapa atribut tipe TIMESTAMP data, selain bidang waktu dalam catatan. TIMESTAMPatribut, dalam catatan multi-ukuran, dapat memiliki stempel waktu di masa depan

atau masa lalu dan berperilaku seperti bidang waktu kecuali Timestream untuk Live Analytics tidak mengindeks nilai tipe TIMESTAMP dalam catatan multi-ukuran.

> **Note**
>
> Tipe TIMESTAMP data hanya didukung untuk catatan multi-ukuran.

Konsistensi akhir untuk membaca

Timestream untuk Live Analytics mendukung semantik konsistensi akhir untuk pembacaan. Ini berarti bahwa ketika Anda melakukan kueri data segera setelah menulis kumpulan data ke Timestream for Live Analytics, hasil kueri mungkin tidak mencerminkan hasil operasi penulisan yang baru saja selesai. Jika Anda mengulangi permintaan kueri ini setelah waktu yang singkat, hasilnya akan mengembalikan data terbaru.

Batching menulis dengan WriteRecords API

Amazon Timestream untuk Live Analytics memungkinkan Anda menulis titik data dari satu deret waktu dan/atau titik data dari banyak seri dalam satu permintaan tulis. Mengelompokkan beberapa titik data dalam satu operasi penulisan bermanfaat dari perspektif kinerja dan biaya. Lihat Menulis di bagian Pengukuran dan Harga untuk detail selengkapnya.

> **Note**
>
> Permintaan penulisan Anda ke Timestream untuk Live Analytics dapat dibatasi karena Timestream untuk skala Live Analytics untuk beradaptasi dengan kebutuhan konsumsi data aplikasi Anda. Jika aplikasi Anda mengalami pengecualian pembatasan, Anda harus terus mengirim data pada throughput yang sama (atau lebih tinggi) agar Timestream for Live Analytics dapat secara otomatis menskalakan kebutuhan aplikasi Anda.

Beban batch

Dengan pemuatan batch untuk Amazon Timestream LiveAnalytics, Anda dapat menelan CSV file yang disimpan di Amazon S3 ke Timestream dalam batch. Dengan fungsionalitas baru ini, Anda dapat memiliki data Anda di Timestream LiveAnalytics tanpa harus bergantung pada alat lain atau menulis kode khusus. Anda dapat menggunakan pemuatan batch untuk mengisi ulang data dengan waktu tunggu yang fleksibel, seperti data yang tidak segera diperlukan untuk kueri atau analisis.

Anda dapat membuat tugas pemuatan batch dengan menggunakan AWS Management Console, AWS CLI, dan file AWS SDKs. Lihat informasi selengkapnya di Menggunakan pemuatan batch dengan konsol, Menggunakan beban batch dengan AWS CLI, dan Menggunakan beban batch dengan AWS SDKs.

Untuk informasi selengkapnya tentang pemuatan batch, lihatMenggunakan pemuatan batch di Timestream untuk LiveAnalytics.

Memilih antara WriteRecords API operasi dan beban batch

Dengan WriteRecords API operasi ini, Anda dapat menulis data deret waktu streaming Anda ke Timestream LiveAnalytics karena dihasilkan oleh sistem Anda. Dengan menggunakan WriteRecords, Anda dapat terus menelan satu titik data atau kumpulan data yang lebih kecil secara real time. Timestream for LiveAnalytics menawarkan skema fleksibel yang secara otomatis mendeteksi nama kolom dan tipe data untuk Timestream Anda untuk LiveAnalytics tabel, berdasarkan nama dimensi dan tipe data dari titik data yang Anda tentukan saat menjalankan penulisan ke dalam database.

Sebaliknya, pemuatan batch memungkinkan penyerapan data deret waktu batch yang kuat dari file sumber (CSVfile) ke Timestream untuk LiveAnalytics, menggunakan model data yang Anda tentukan. Beberapa contoh kapan menggunakan pemuatan batch dengan file sumber adalah mengimpor data deret waktu secara massal untuk evaluasi Timestream LiveAnalytics melalui bukti konsep, mengimpor data deret waktu secara massal dari perangkat IoT yang offline selama beberapa waktu, dan memigrasi data deret waktu historis dari Amazon S3 ke Timestream untuk. LiveAnalytics Untuk informasi tentang pemuatan batch, lihatMenggunakan pemuatan batch di Timestream untuk LiveAnalytics.

Kedua solusi tersebut aman, andal, dan berkinerja baik.

Gunakan WriteRecords saat:

- Streaming data dalam jumlah yang lebih kecil (kurang dari 10 MB) per permintaan.
- Mengisi tabel yang ada.
- Menelan data dari aliran log.
- Melakukan analitik waktu nyata.
- Membutuhkan latensi yang lebih rendah.

Gunakan pemuatan batch saat:

- Menelan beban data yang lebih besar yang berasal dari Amazon CSV S3 dalam file. Untuk informasi selengkapnya tentang batasn, lihat Kuota.
- Mengisi tabel baru, seperti dalam kasus migrasi data.
- Memperkaya database dengan data historis (konsumsi ke dalam tabel baru).
- Anda memiliki sumber data yang berubah perlahan atau tidak sama sekali.
- Anda memiliki waktu tunggu yang fleksibel karena tugas pemuatan batch mungkin dalam status tertunda hingga sumber daya tersedia, terutama jika Anda memuat data dalam jumlah yang sangat besar. Batch load cocok untuk data yang tidak perlu tersedia untuk kueri atau analisis untuk menambah kejelasan.

Penyimpanan

Timestream untuk Live Analytics menyimpan dan mengatur data deret waktu Anda untuk mengoptimalkan waktu pemrosesan kueri dan mengurangi biaya penyimpanan. Ini menawarkan tiering penyimpanan data dan mendukung dua tingkatan penyimpanan: penyimpanan memori dan penyimpanan magnetik. Penyimpanan memori dioptimalkan untuk penulisan data throughput tinggi dan point-in-time kueri cepat. Penyimpanan magnetik dioptimalkan untuk throughput yang lebih rendah penulisan data yang datang terlambat, penyimpanan data jangka panjang, dan kueri analitik cepat.

Timestream untuk Live Analytics memastikan daya tahan data Anda dengan secara otomatis mereplikasi memori dan data penyimpanan magnetik Anda di berbagai Availability Zone dalam satu. Wilayah AWS Semua data Anda ditulis ke disk sebelum mengakui permintaan tulis Anda sebagai lengkap.

Timestream untuk Live Analytics memungkinkan Anda mengonfigurasi kebijakan retensi untuk memindahkan data dari penyimpanan memori ke penyimpanan magnetik. Saat data mencapai nilai yang dikonfigurasi, Timestream untuk Live Analytics secara otomatis memindahkan data ke penyimpanan magnetik. Anda juga dapat mengatur nilai retensi pada penyimpanan magnetik. Ketika data kedaluwarsa keluar dari penyimpanan magnetik, itu dihapus secara permanen.

Misalnya, pertimbangkan skenario di mana Anda mengonfigurasi penyimpanan memori untuk menyimpan data selama seminggu dan penyimpanan magnetik untuk menyimpan data selama 1 tahun. Usia data dihitung menggunakan stempel waktu yang terkait dengan titik data. Ketika data di penyimpanan memori menjadi berumur seminggu, secara otomatis dipindahkan ke penyimpanan magnetik. Itu kemudian disimpan di toko magnet selama setahun. Ketika data menjadi satu tahun, itu

dihapus dari Timestream untuk Live Analytics. Nilai retensi penyimpanan memori dan penyimpanan magnetik secara kumulatif menentukan jumlah waktu penyimpanan data Anda di Timestream untuk Live Analytics. Ini berarti bahwa untuk skenario di atas, sejak saat kedatangan data, data disimpan dalam Timestream untuk Live Analytics untuk jangka waktu total 1 tahun dan 1 minggu.

> **Note**
>
> Saat Anda meningkatkan periode retensi memori atau penyimpanan magnetik, perubahan retensi mulai berlaku sejak saat itu dan seterusnya. Misalnya, jika periode retensi penyimpanan memori diatur ke 2 jam dan kemudian diubah menjadi 24 jam dengan memperbarui kebijakan retensi tabel, penyimpanan memori akan mampu menyimpan 24 jam data, tetapi akan diisi dengan 24 jam data 22 jam setelah perubahan ini dilakukan. Timestream untuk Live Analytics tidak mengambil data dari penyimpanan magnetik untuk mengisi penyimpanan memori.

Untuk memastikan keamanan data deret waktu Anda, data Anda di Timestream for Live Analytics selalu dienkripsi secara default. Ini berlaku untuk data dalam perjalanan dan saat istirahat. Selain itu, Timestream for Live Analytics memungkinkan Anda menggunakan kunci yang dikelola pelanggan untuk mengamankan data Anda di penyimpanan magnetik. Untuk informasi selengkapnya tentang kunci yang dikelola pelanggan, lihat AWS KMS keys.

Kueri

Dengan Timestream for Live Analytics, Anda dapat dengan mudah menyimpan dan menganalisis metrik DevOps, sensor data untuk aplikasi IoT, dan data telemetri industri untuk pemeliharaan peralatan, serta banyak kasus penggunaan lainnya. Mesin kueri adaptif yang dibuat khusus di Timestream for Live Analytics memungkinkan Anda mengakses data di seluruh tingkatan penyimpanan menggunakan satu pernyataan. SQL Ini secara transparan mengakses dan menggabungkan data di seluruh tingkatan penyimpanan tanpa mengharuskan Anda menentukan lokasi data. Anda dapat menggunakan SQL kueri data di Timestream untuk Live Analytics untuk mengambil data deret waktu dari satu atau beberapa tabel. Anda dapat mengakses informasi metadata untuk database dan tabel. Timestream untuk Live Analytics SQL juga mendukung fungsi bawaan untuk analitik deret waktu. Anda dapat merujuk ke Referensi bahasa kueri referensi untuk detail tambahan.

Timestream for Live Analytics dirancang untuk memiliki arsitektur penyerapan data, penyimpanan, dan kueri yang sepenuhnya dipisahkan di mana setiap komponen dapat menskalakan secara

independen dari komponen lain (memungkinkannya menawarkan skala yang hampir tak terbatas untuk kebutuhan aplikasi). Ini berarti bahwa Timestream untuk Live Analytics tidak “memberi tip” ketika aplikasi Anda mengirim ratusan terabyte data per hari atau menjalankan jutaan kueri yang memproses data dalam jumlah kecil atau besar. Seiring pertumbuhan data Anda dari waktu ke waktu, latensi kueri di Timestream untuk Live Analytics sebagian besar tetap tidak berubah. Ini karena arsitektur kueri Timestream for Live Analytics dapat memanfaatkan paralelisme dalam jumlah besar untuk memproses volume data yang lebih besar dan secara otomatis menskalakan agar sesuai dengan kebutuhan throughput kueri aplikasi.

Model data

Timestream mendukung dua model data untuk kueri — model datar dan model deret waktu.

> **Note**
>
> Data dalam Timestream disimpan menggunakan model datar dan itu adalah model default untuk query data. Model deret waktu adalah konsep waktu kueri dan digunakan untuk analitik deret waktu.

- Model datar
- Model deret waktu

Model datar

Model datar adalah model data default Timestream untuk kueri. Ini mewakili data deret waktu dalam format tabel. Nama dimensi, waktu, nama ukuran dan nilai ukuran muncul sebagai kolom. Setiap baris dalam tabel adalah titik data atom yang sesuai dengan pengukuran pada waktu tertentu dalam deret waktu. Database Timestream, tabel, dan kolom memiliki beberapa kendala penamaan. Itu dijelaskan dalamBatas layanan.

Tabel di bawah ini menunjukkan contoh ilustrasi tentang bagaimana Timestream menyimpan data yang mewakili CPU pemanfaatan, pemanfaatan memori, dan aktivitas jaringan EC2 instance, saat data dikirim sebagai catatan ukuran tunggal. Dalam hal ini, dimensinya adalah wilayah, zona ketersediaan, cloud pribadi virtual, dan instance instanceIDs. EC2 Langkah-langkahnya adalah CPU pemanfaatan, pemanfaatan memori, dan data jaringan yang masuk untuk instance. EC2 Wilayah kolom, az, vpc, dan instance_id berisi nilai dimensi. Waktu kolom berisi stempel waktu untuk setiap

catatan. Kolom measurement _name berisi nama-nama ukuran yang diwakili oleh cpu-utilization, memory_utilization, dan network_bytes_in. Kolom measure _value: :double berisi pengukuran yang dipancarkan sebagai ganda (misalnya pemanfaatan dan pemanfaatan memori). CPU Kolom measure _value: :bigint berisi pengukuran yang dipancarkan sebagai bilangan bulat misalnya data jaringan yang masuk.

Waktu	region	az	vpc	instance_id	ukuran_nama	ukuran_nilai: :ganda	ukuran_nilai: :bigint
2019-12-04 19:00:00.000 000000	us-east-1	kami-timur-1d	vpc-1a2b3c4d	i-1234567890abcdef0	pemanfaatan cpu_	35,0	null
2019-12-04 19:00:01.000 000000	us-east-1	kami-timur-1d	vpc-1a2b3c4d	i-1234567890abcdef0	pemanfaatan cpu_	38.2	null
2019-12-04 19:00:02.000 000000	us-east-1	kami-timur-1d	vpc-1a2b3c4d	i-1234567890abcdef0	pemanfaatan cpu_	45.3	null
2019-12-04 19:00:00.000 000000	us-east-1	kami-timur-1d	vpc-1a2b3c4d	i-1234567890abcdef0	memory_utilization	54,9	null
2019-12-04 19:00:01.000 000000	us-east-1	kami-timur-1d	vpc-1a2b3c4d	i-1234567890abcdef0	memory_utilization	42.6	null

Waktu	region	az	vpc	instance_id	ukuran_nama	ukuran_nilai: :ganda	ukuran_nilai: :bigint
2019-12-04 19:00:02.000 000000	us-east-1	kami-timur-1d	vpc-1a2b3c4d	i-1234567890abcdef0	memory_utilization	33.3	null
2019-12-04 19:00:00.000 000000	us-east-1	kami-timur-1d	vpc-1a2b3c4d	i-1234567890abcdef0	network_bytes	34.400	null
2019-12-04 19:00:01.000 000000	us-east-1	kami-timur-1d	vpc-1a2b3c4d	i-1234567890abcdef0	network_bytes	1.500	null
2019-12-04 19:00:02.000 000000	us-east-1	kami-timur-1d	vpc-1a2b3c4d	i-1234567890abcdef0	network_bytes	6.000	null

Tabel di bawah ini menunjukkan contoh ilustrasi tentang bagaimana Timestream menyimpan data yang mewakili CPU pemanfaatan, pemanfaatan memori, dan aktivitas jaringan EC2 instance, saat data dikirim sebagai catatan multi-ukuran.

Waktu	region	az	vpc	instance_id	ukuran_nama	pemanfaatan cpu_	memory_ilization	network_bytes
2019-12-04 19:00:00.	us-east-1	kami-timur-1d	vpc-1a2b3c4d	i-1234567890abcde0	metrik	35,0	54,9	34.400

Waktu	region	az	vpc	instance_id	ukuran_nama	pemanfaatan cpu_	memory_utilization	network_bytes
000 000000								
2019-12-04 19:00:01.000 000000	us-east-1	kami-timur-1d	vpc-1a2b3c4d	i-1234567890abcde0	metrik	38.2	42.6	1.500
2019-12-04 19:00:02.000 000000	us-east-1	kami-timur-1d	vpc-1a2b3c4d	i-1234567890abcde0	metrik	45.3	33.3	6.600

Model deret waktu

Model deret waktu adalah konstruksi waktu kueri yang digunakan untuk analitik deret waktu. Ini mewakili data sebagai urutan urutan pasangan (waktu, nilai ukur). Timestream mendukung fungsi deret waktu seperti interpolasi untuk memungkinkan Anda mengisi celah dalam data Anda. Untuk menggunakan fungsi-fungsi ini, Anda harus mengonversi data Anda menjadi model deret waktu menggunakan fungsi seperti create_time_series. Lihat Referensi bahasa kueri untuk lebih jelasnya.

Menggunakan contoh contoh sebelumnya, berikut adalah data CPU pemanfaatan yang dinyatakan sebagai timeseries. EC2

region	az	vpc	instance_id	pemanfaatan cpu_
us-east-1	kami-timur-1d	vpc-1a2b3c4d	i-1234567890abcdef0	[{waktu: 2019-12-04 19:00:00.000 000000, nilai: 35}, {waktu: 2019-12-04

region	az	vpc	instance_id	pemanfaatan cpu_
				19:00:01.000 000000, nilai: 38.2}, {waktu: 2019-12-04 19:00:02.000 000000, nilai: 45.3}]

Pertanyaan terjadwal

Fitur kueri terjadwal di Amazon Timestream untuk Live Analytics adalah solusi yang dikelola sepenuhnya, tanpa server, dan dapat diskalakan untuk menghitung dan menyimpan agregat, rollup, dan bentuk data pra-proses lainnya yang biasanya digunakan untuk memberi daya pada dasbor operasional, laporan bisnis, analitik ad hoc, dan aplikasi lainnya. Kueri terjadwal membuat analitik real-time lebih berkinerja dan hemat biaya, sehingga Anda dapat memperoleh wawasan tambahan dari data Anda, dan dapat terus membuat keputusan bisnis yang lebih baik.

Untuk informasi selengkapnya tentang kueri terjadwal, lihatMenggunakan kueri terjadwal di Timestream untuk LiveAnalytics.

Unit Komputasi Aliran Waktu () TCU

Amazon Timestream untuk Live Analytics mengukur kapasitas komputasi yang dialokasikan kepada Anda untuk kebutuhan kueri di unit komputasi Timestream (). TCU Satu TCU terdiri dari 4 vCPUs dan 16 GB memori. Saat Anda menjalankan kueri di Timestream untuk Live Analytics, layanan mengalokasikan TCUs sesuai permintaan berdasarkan kompleksitas kueri Anda dan jumlah data yang sedang diproses. Jumlah TCUs yang dikonsumsi kueri menentukan biaya terkait.

> **Note**
>
> Semua Akun AWS yang ada di layanan setelah 29 April 2024 akan digunakan TCUs secara default untuk harga kueri.

Dalam topik ini:

MaxQuery TCU

Pengaturan ini menentukan jumlah maksimum unit komputasi yang akan digunakan layanan kapan saja untuk melayani kueri Anda. Untuk menjalankan kueri, Anda harus mengatur kapasitas minimum menjadi 4TCUs. Anda dapat mengatur jumlah maksimum TCUs dalam kelipatan 4, misalnya, 4, 8, 16, 32, dan seterusnya. Anda hanya dikenakan biaya untuk sumber daya komputasi yang Anda gunakan untuk beban kerja Anda. Misalnya, jika Anda mengatur maksimum TCUs ke 128, tetapi secara konsisten hanya menggunakan 8TCUs. Anda akan dikenakan biaya hanya untuk durasi selama Anda menggunakan 8TCUs. Default `MaxQueryTCU` di akun Anda diatur ke 200. Anda dapat menyesuaikan `MaxQueryTCU` dari 4 hingga 1000, menggunakan AWS Management Console atau UpdateAccountSettingsAPIoperasi dengan AWS SDK atau AWS CLI.

Kami merekomendasikan pengaturan `MaxQueryTCU` untuk akun Anda. Menetapkan TCU batas maksimum membantu mengontrol biaya dengan membatasi jumlah unit komputasi yang dapat digunakan layanan untuk beban kerja kueri Anda. Ini memungkinkan Anda untuk memprediksi dan mengelola pengeluaran kueri Anda dengan lebih baik.

Tagihan untuk TCU

Masing-masing TCU ditagih setiap jam dengan granularitas per detik dan minimal 30 detik. Unit penggunaan unit komputasi ini adalah TCU -jam.

Saat menjalankan kueri, Anda ditagih untuk yang TCUs digunakan selama waktu eksekusi kueri, diukur dalam TCU -jam. Sebagai contoh:

- Beban kerja Anda menggunakan 20 TCUs selama 3 jam. Anda ditagih selama 60 TCU jam (20 TCUs x 3 jam).

- Beban kerja Anda menggunakan 10 TCUs selama 30 menit, dan kemudian 20 TCUs untuk 30 menit berikutnya. Anda ditagih selama 15 TCU jam (10 TCUs x 0,5 jam +20 TCUs x 0,5 jam).

Harga TCU per jam bervariasi menurut Wilayah AWS. Lihat harga Amazon Timestream untuk detail tambahan. Seiring bertambahnya beban kerja Anda, layanan secara otomatis menskalakan kapasitas komputasi hingga TCU batas maksimum (`MaxQueryTCU`) yang ditentukan untuk mempertahankan kinerja yang konsisten. `MaxQueryTCU`Pengaturan bertindak sebagai plafon untuk kapasitas komputasi yang dapat diskalakan oleh layanan. Pengaturan ini membantu Anda mengontrol jumlah sumber daya komputasi dan akibatnya biayanya.

Mengkonfigurasi TCU

Ketika Anda onboard layanan, masing-masing Akun AWS memiliki `MaxQueryTCU` batas default 200. Anda dapat memperbarui batas ini sesuai kebutuhan kapan saja menggunakan AWS Management Console atau UpdateAccountSettingsAPIoperasi dengan AWS SDK atau AWS CLI.

Jika Anda tidak yakin tentang nilai yang akan dikonfigurasi, pantau `QueryTCU` metrik untuk akun Anda. Metrik ini tersedia di AWS Management Console dan Amazon CloudWatch. Metrik ini memberikan wawasan tentang jumlah maksimum yang TCUs digunakan pada perincian satu menit. Berdasarkan data historis dan estimasi Anda tentang pertumbuhan masa depan, tetapkan `MaxQueryTCU` untuk mengakomodasi lonjakan penggunaan Anda. Kami merekomendasikan memiliki ruang kepala setidaknya 4-16 di TCUs atas penggunaan puncak Anda. Misalnya, jika puncak Anda `QueryTCU` dalam 30 hari terakhir adalah 128, kami sarankan pengaturan `MaxQueryTCU` antara 132 hingga 144.

Memperkirakan unit komputasi yang diperlukan

Unit komputasi dapat memproses kueri secara bersamaan. Untuk menentukan jumlah unit komputasi yang diperlukan, pertimbangkan pedoman umum dalam tabel berikut:

Kueri bersamaan	TCUs
7	4
14	8
21	12

> **Note**
>
> - Ini adalah pedoman umum dan jumlah aktual unit komputasi yang diperlukan tergantung pada beberapa faktor, seperti:
>   - Konkurensi kueri yang efektif.
>   - Pola kueri.
>   - Jumlah partisi yang dipindai.
>   - Karakteristik khusus beban kerja lainnya.
> - Pedoman ini berkaitan dengan kueri yang memindai data selama beberapa menit terakhir hingga satu jam dan mematuhi praktik terbaik kueri Timestream dan pedoman pemodelan Data.
> - Pantau performa dan `QueryTCU` metrik aplikasi Anda untuk menyesuaikan unit komputasi, sesuai kebutuhan.

Kapan harus meningkat MaxQuery TCU

Anda harus mempertimbangkan untuk `MaxQueryTCU` meningkatkan skenario berikut:

- Konsumsi kueri puncak Anda mendekati atau mencapai kueri maksimum yang dikonfigurasi saat iniTCU. Sebaiknya atur kueri maksimum TCU setidaknya 4-16 TCUs lebih tinggi dari konsumsi puncak Anda.
- Kueri Anda mengembalikan kesalahan 4xx dengan pesan MaxQuery TCU terlampaui. Jika Anda mengantisipasi peningkatan beban kerja yang direncanakan, tinjau kembali dan sesuaikan kueri maksimum yang dikonfigurasi. TCU

Kapan harus menurun MaxQuery TCU

Anda harus mempertimbangkan untuk mengurangi skenario `MaxQueryTCU` berikut:

- Beban kerja Anda memiliki pola penggunaan yang dapat diprediksi dan stabil, dan Anda memiliki pemahaman yang baik tentang persyaratan penggunaan komputasi Anda. Menurunkan kueri maksimum TCU ke dalam 4-16 TCU di atas konsumsi puncak Anda dapat membantu mencegah penggunaan dan biaya yang tidak disengaja. Anda dapat memodifikasi nilai menggunakan UpdateAccountSettingsAPIoperasi.

- Penggunaan puncak beban kerja Anda telah menurun dari waktu ke waktu, baik karena perubahan dalam aplikasi atau pola perilaku pengguna. Menurunkan maksimum TCU dapat membantu mengurangi biaya yang tidak disengaja.

> **Note**
>
> Bergantung pada penggunaan Anda saat ini, mengurangi perubahan TCU batas maksimum mungkin memakan waktu hingga 24 jam agar efektif. Anda ditagih hanya untuk kueri TCUs yang benar-benar digunakan oleh kueri Anda. Memiliki TCU batas kueri maksimum yang lebih tinggi tidak memengaruhi biaya Anda kecuali jika TCUs digunakan oleh beban kerja Anda.

Memantau penggunaan dengan CloudWatch metrik

Untuk memantau TCU penggunaan Anda, Timestream untuk Live Analytics menyediakan CloudWatch metrik berikut:`QueryTCU`. Metrik ini menentukan jumlah satuan komputasi yang digunakan dalam satu menit dan dipancarkan setiap menit. Anda dapat memilih untuk memantau maksimum dan minimum yang TCUs digunakan dalam satu menit. Anda juga dapat mengatur alarm pada metrik ini untuk melacak biaya kueri Anda secara real-time.

Memahami variasi dalam penggunaan unit komputasi

Jumlah sumber daya komputasi yang diperlukan untuk kueri Anda dapat bertambah atau berkurang berdasarkan beberapa parameter. Misalnya, volume data, pola konsumsi data, latensi kueri, bentuk kueri, efisiensi kueri, dan kombinasi kueri yang menggunakan kueri real-time dan analitis. Parameter ini dapat menyebabkan TCU unit yang lebih tinggi atau lebih rendah yang diperlukan untuk beban kerja Anda. Dalam kondisi mapan di mana parameter ini tidak berubah, Anda mungkin mengamati bahwa jumlah unit komputasi yang diperlukan untuk beban kerja Anda berkurang. Akibatnya, ini dapat menurunkan biaya bulanan Anda.

Selain itu, jika salah satu parameter ini dalam beban kerja atau data Anda berubah, jumlah unit komputasi yang diperlukan mungkin meningkat. Saat Timestream menerima kueri, tergantung pada partisi data yang diakses kueri, Timestream memutuskan jumlah sumber daya komputasi untuk menangani kueri secara kinerja.

Pada interval periodik, berdasarkan pola akses konsumsi dan kueri Anda, Timestream mengoptimalkan tata letak data. Timestream melakukan optimasi dengan clubbing partisi yang

kurang diakses menjadi satu partisi atau membagi partisi panas menjadi beberapa partisi untuk kinerja. Akibatnya, kapasitas komputasi yang digunakan oleh kueri yang sama mungkin sedikit berbeda pada titik waktu yang berbeda.

> **Memilih untuk menggunakan TCU harga untuk kueri Anda**
>
> Sebagai pengguna yang sudah ada, Anda dapat melakukan satu kali opt-in untuk digunakan TCUs untuk manajemen biaya yang lebih baik dan penghapusan per byte minimum kueri yang diukur. Anda dapat memilih untuk menggunakan AWS Management Console atau UpdateAccountSettingsAPIoperasi dengan AWS SDK atau AWS CLI. Dalam API operasi, atur `QueryPricingModel` parameter ke`COMPUTE_UNITS`.
> Memilih model penetapan harga berbasis komputasi adalah perubahan yang tidak dapat diubah.

Mengakses Timestream untuk LiveAnalytics

Anda dapat mengakses Timestream untuk LiveAnalytics menggunakan konsol, CLI atau. API Untuk informasi tentang mengakses Timestream LiveAnalytics, tinjau hal-hal berikut:

Topik

- Mendaftar untuk Akun AWS
- Buat pengguna dengan akses administratif
- Menyediakan Timestream untuk akses LiveAnalytics
- Memberikan akses programatis

Mendaftar untuk Akun AWS

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

Untuk mendaftar untuk Akun AWS

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

   Bagian dari prosedur pendaftaran melibatkan tindakan menerima panggilan telepon dan memasukkan kode verifikasi di keypad telepon.

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

AWS mengirimi Anda email konfirmasi setelah proses pendaftaran selesai. Kapan saja, Anda dapat melihat aktivitas akun Anda saat ini dan mengelola akun Anda dengan masuk https://aws.amazon.comke/ dan memilih Akun Saya.

Buat pengguna dengan akses administratif

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

Amankan Anda Pengguna root akun AWS

1. Masuk ke AWS Management Consolesebagai pemilik akun dengan memilih pengguna Root dan memasukkan alamat Akun AWS email Anda. Di laman berikutnya, masukkan kata sandi.

   Untuk bantuan masuk dengan menggunakan pengguna root, lihat Masuk sebagai pengguna root di AWS Sign-In Panduan Pengguna.
2. Aktifkan otentikasi multi-faktor (MFA) untuk pengguna root Anda.

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

Buat pengguna dengan akses administratif

1. Aktifkan Pusat IAM Identitas.

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

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

Masuk sebagai pengguna dengan akses administratif

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

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

Tetapkan akses ke pengguna tambahan

1. Di Pusat IAM Identitas, buat set izin yang mengikuti praktik terbaik menerapkan izin hak istimewa paling sedikit.

   Untuk petunjuknya, lihat Membuat set izin di Panduan AWS IAM Identity Center Pengguna.
2. Tetapkan pengguna ke grup, lalu tetapkan akses masuk tunggal ke grup.

   Untuk petunjuk, lihat Menambahkan grup di Panduan AWS IAM Identity Center Pengguna.

Menyediakan Timestream untuk akses LiveAnalytics

Izin yang diperlukan untuk mengakses Timestream LiveAnalytics sudah diberikan kepada administrator. Untuk pengguna lain, Anda harus memberi mereka Timestream untuk LiveAnalytics akses menggunakan kebijakan berikut:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "timestream:*",
        "kms:DescribeKey",
        "kms:CreateGrant",
        "kms:Decrypt",
        "dbqms:CreateFavoriteQuery",
        "dbqms:DescribeFavoriteQueries",
        "dbqms:UpdateFavoriteQuery",
        "dbqms:DeleteFavoriteQueries",
        "dbqms:GetQueryString",
        "dbqms:CreateQueryHistory",
```

```
        "dbqms:UpdateQueryHistory",
        "dbqms:DeleteQueryHistory",
        "dbqms:DescribeQueryHistory",
        "s3:ListAllMyBuckets"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> Untuk selengkapnyadbqms, lihat Tindakan, sumber daya, dan kunci kondisi untuk Layanan Metadata Kueri Database. Untuk selengkapnya, kms lihat Kunci tindakan, sumber daya, dan kondisi untuk Layanan Manajemen AWS Kunci.

Memberikan akses programatis

Pengguna membutuhkan akses terprogram jika mereka ingin berinteraksi dengan AWS luar. AWS Management Console Cara untuk memberikan akses terprogram tergantung pada jenis pengguna yang mengakses AWS.

Untuk memberi pengguna akses programatis, pilih salah satu opsi berikut.

Pengguna mana yang membutuhkan akses programatis?	Untuk	Oleh
Identitas tenaga kerja (Pengguna dikelola di Pusat IAM Identitas)	Gunakan kredensi sementara untuk menandatangani permintaan terprogram ke AWS CLI,, AWS SDKs atau. AWS APIs	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk AWS CLI, lihat Mengkonfigurasi yang akan AWS CLI digunakan AWS IAM Identity Center dalam Panduan AWS Command Line Interface Pengguna.

Pengguna mana yang membutuhkan akses programatis?	Untuk	Oleh
		• Untuk AWS SDKs, alat, dan AWS APIs, lihat otentikasi Pusat IAM Identitas di Panduan Referensi Alat AWS SDKs dan Alat.
IAM	Gunakan kredensi sementara untuk menandatangani permintaan terprogram ke AWS CLI,, AWS SDKs atau. AWS APIs	Mengikuti petunjuk dalam Menggunakan kredensil sementara dengan AWS sumber daya di IAMPanduan Pengguna.
IAM	(Tidak direkomendasikan) Gunakan kredensi jangka panjang untuk menandatangani permintaan terprogram ke AWS CLI,, AWS SDKs atau. AWS APIs	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk mengetahui AWS CLI, lihat Mengautentikasi menggunakan kredensil IAM pengguna di Panduan Pengguna.AWS Command Line Interface • Untuk AWS SDKs dan alat, lihat Mengautentikasi menggunakan kredensil jangka panjang di Panduan Referensi Alat AWS SDKs dan Alat. • Untuk AWS APIs, lihat Mengelola kunci akses untuk IAM pengguna di Panduan IAM Pengguna.

Menggunakan konsol

Anda dapat menggunakan AWS Management Console untuk Timestream Live Analytics untuk membuat, mengedit, menghapus, mendeskripsikan, dan mencantumkan database dan tabel. Anda juga dapat menggunakan konsol untuk menjalankan kueri.

Topik

Tutorial

Tutorial ini menunjukkan cara membuat database yang diisi dengan kumpulan data sampel dan menjalankan query sampel. Contoh dataset yang digunakan dalam tutorial ini sering terlihat di IoT dan skenario. DevOps Dataset IoT berisi data deret waktu seperti kecepatan, lokasi, dan beban truk, untuk merampingkan manajemen armada dan mengidentifikasi peluang pengoptimalan. DevOps Dataset berisi metrik EC2 instance sepertiCPU, jaringan, dan pemanfaatan memori untuk meningkatkan kinerja dan ketersediaan aplikasi. Berikut adalah video tutorial untuk petunjuk yang dijelaskan di bagian ini

Ikuti langkah-langkah ini untuk membuat database yang diisi dengan kumpulan data sampel dan menjalankan kueri sampel menggunakan AWS Konsol.

1. Buka AWS konsol.
2. Di panel navigasi, pilih Databases
3. Klik Buat database.

4. Pada halaman buat database, masukkan yang berikut ini:
   - Pilih konfigurasi —Pilih database Sampel.
   - Nama —Masukkan nama database pilihan Anda.
   - Pilih kumpulan data sampel —Pilih IoT dan. DevOps
   - Klik Buat database untuk membuat database yang berisi dua tabel—IoT dan DevOps diisi dengan data sampel.
5. Di panel navigasi, pilih Editor kueri
6. Pilih Contoh kueri dari menu atas.
7. Klik salah satu contoh kueri. Ini akan membawa Anda kembali ke editor kueri dengan editor diisi dengan kueri sampel.
8. Klik Jalankan untuk menjalankan kueri dan melihat hasil kueri.

Buat database

Ikuti langkah-langkah ini untuk membuat database menggunakan AWS Konsol.

1. Buka AWS konsol.
2. Di panel navigasi, pilih Databases
3. Klik Buat database.
4. Pada halaman buat database, masukkan yang berikut ini.
   - Pilih konfigurasi —Pilih Database standar.
   - Nama —Masukkan nama database pilihan Anda.
   - Enkripsi —Pilih KMS kunci atau gunakan opsi default, di mana Timestream Live Analytics akan membuat KMS kunci di akun Anda jika belum ada.
5. Klik Buat database untuk membuat database.

Membuat tabel

Ikuti langkah-langkah ini untuk membuat tabel menggunakan AWS Konsol.

1. Buka AWS konsol.
2. Di panel navigasi, pilih Tabel

3. Klik Buat tabel.
4. Pada halaman buat tabel, masukkan yang berikut ini.
   - Nama database —Pilih nama database yang dibuat diBuat database.
   - Nama tabel —Masukkan nama tabel pilihan Anda.
   - Penyimpanan penyimpanan memori —Tentukan berapa lama Anda ingin menyimpan data di penyimpanan memori. Penyimpanan memori memproses data yang masuk, termasuk data yang datang terlambat (data dengan stempel waktu lebih awal dari waktu saat ini) dan dioptimalkan untuk kueri cepat. point-in-time
   - Retensi penyimpanan magnetik —Tentukan berapa lama Anda ingin menyimpan data di penyimpanan magnetik. Penyimpanan magnetik dimaksudkan untuk penyimpanan jangka panjang dan dioptimalkan untuk kueri analitik cepat.
5. Klik Buat tabel.

Jalankankueri

Ikuti langkah-langkah ini untuk menjalankan kueri menggunakan AWS Konsol.

1. Buka AWS konsol.
2. Di panel navigasi, pilih Editor kueri
3. Di panel kiri, pilih database yang dibuat diBuat database.
4. Di panel kiri, pilih database yang dibuat diMembuat tabel.
5. Di editor kueri, Anda dapat menjalankan kueri. Untuk melihat 10 baris terbaru dalam tabel, jalankan:

   ```
   SELECT * FROM <database_name>.<table_name> ORDER BY time DESC LIMIT 10
   ```
6. (Opsional) Aktifkan Aktifkan Wawasan untuk mendapatkan wawasan tentang efisiensi kueri Anda.

Buat kueri terjadwal

Ikuti langkah-langkah ini untuk membuat kueri terjadwal menggunakan AWS Konsol.

1. Buka AWS konsol.
2. Di panel navigasi, pilih Kueri terjadwal.

3. Klik Buat kueri terjadwal.
4. Di bagian Nama Kueri dan Tabel Tujuan, masukkan yang berikut ini.
   - Nama —Masukkan nama kueri.
   - Nama database —Pilih nama database yang dibuat diBuat database.
   - Nama tabel —Pilih nama tabel yang dibuat diMembuat tabel.
5. Di bagian Pernyataan Kueri, masukkan pernyataan kueri yang valid. Kemudian klik Validasi kueri.
6. Dari model tabel Tujuan, tentukan model untuk atribut yang tidak ditentukan. Anda dapat menggunakan Visual Builder atauJSON.
7. Di bagian Jalankan jadwal, pilih Tingkat tetap atau ekspresi Chron. Untuk ekspresi chron, lihat Ekspresi Jadwal untuk Kueri Terjadwal untuk detail selengkapnya tentang ekspresi jadwal.
8. Di bagian SNStopik, masukkan SNS topik yang akan digunakan untuk pemberitahuan.
9. Di bagian Laporan log kesalahan masukkan lokasi S3 yang akan digunakan untuk melaporkan kesalahan.

   Pilih jenis kunci Enkripsi.
10. Di bagian Pengaturan keamanan dari AWS KMSkunci, pilih jenis AWS KMS kunci.

    Masukkan IAMperan yang LiveAnalytics akan digunakan Timestream untuk menjalankan kueri terjadwal. Lihat contoh IAM kebijakan untuk kueri terjadwal untuk detail tentang izin yang diperlukan dan hubungan kepercayaan untuk peran tersebut.
11. Klik Buat kueri terjadwal.

Hapus kueri terjadwal

Ikuti langkah-langkah berikut untuk menghapus atau menonaktifkan kueri terjadwal menggunakan AWS Konsol.

1. Buka AWS konsol.
2. Di panel navigasi, pilih Kueri terjadwal
3. Pilih kueri terjadwal yang dibuat diBuat kueri terjadwal.
4. Pilih Tindakan.
5. Pilih Nonaktifkan atau Hapus.

6. Jika Anda memilih Hapus, konfirmasikan tindakan dan pilih Hapus.

Menghapus tabel

Ikuti langkah-langkah berikut untuk menghapus database menggunakan AWS Konsol.

1. Buka AWS konsol.
2. Di panel navigasi, pilih Tabel
3. Pilih tabel yang Anda buatMembuat tabel.
4. Klik Hapus.
5. Ketik hapus di kotak konfirmasi.

Hapus database

Ikuti langkah-langkah berikut untuk menghapus database menggunakan AWS Konsol:

1. Buka AWS konsol.
2. Di panel navigasi, pilih Databases
3. Pilih database yang Anda buat di Buat database.
4. Klik Hapus.
5. Ketik hapus di kotak konfirmasi.

Mengedit tabel

Ikuti langkah-langkah ini untuk mengedit tabel menggunakan AWS Konsol.

1. Buka AWS konsol.
2. Di panel navigasi, pilih Tabel
3. Pilih tabel yang Anda buatMembuat tabel.
4. Klik Edit
5. Edit detail tabel dan simpan.

    - Penyimpanan penyimpanan memori —Tentukan berapa lama Anda ingin menyimpan data di penyimpanan memori. Penyimpanan memori memproses data yang masuk, termasuk

data yang datang terlambat (data dengan stempel waktu lebih awal dari waktu saat ini) dan dioptimalkan untuk kueri cepat. point-in-time

- Retensi penyimpanan magnetik —Tentukan berapa lama Anda ingin menyimpan data di penyimpanan magnetik. Penyimpanan magnetik dimaksudkan untuk penyimpanan jangka panjang dan dioptimalkan untuk kueri analitik cepat.

Mengedit database

Ikuti langkah-langkah ini untuk mengedit database menggunakan AWS Konsol.

1. Buka AWS konsol.
2. Di panel navigasi, pilih Databases
3. Pilih database yang Anda buat di Buat database.
4. Klik Edit
5. Edit detail database dan simpan.

Mengakses Amazon Timestream LiveAnalytics untuk menggunakan AWS CLI

Anda dapat menggunakan AWS Command Line Interface (AWS CLI) untuk mengontrol beberapa AWS layanan dari baris perintah dan mengotomatiskannya melalui skrip. Anda dapat menggunakan AWS CLI untuk operasi ad hoc. Anda juga dapat menggunakannya untuk menyematkan Amazon Timestream LiveAnalytics untuk operasi dalam skrip utilitas.

Sebelum Anda dapat menggunakan Timestream for LiveAnalytics, Anda harus mengatur akses terprogram. AWS CLI Untuk informasi selengkapnya, lihat Memberikan akses programatis.

Untuk daftar lengkap semua perintah yang tersedia untuk Timestream for LiveAnalytics Query API di AWS CLI, lihat Referensi AWS CLI Perintah.

Untuk daftar lengkap semua perintah yang tersedia untuk Timestream for LiveAnalytics Write API di AWS CLI, lihat Referensi AWS CLI Perintah.

Topik

- Mengunduh dan mengonfigurasi AWS CLI
- Menggunakan Timestream AWS CLI with untuk LiveAnalytics

Mengunduh dan mengonfigurasi AWS CLI

AWS CLI Berjalan di Windows, macOS, atau Linux. Untuk mengunduh, menginstal, dan mengonfigurasinya, ikuti langkah-langkah berikut:

1. Unduh AWS CLI di http://aws.amazon.com/cli.
2. Ikuti petunjuk untuk Menginstal AWS CLI dan Mengkonfigurasi AWS CLI di Panduan AWS Command Line Interface Pengguna.

Menggunakan Timestream AWS CLI with untuk LiveAnalytics

Format baris perintah terdiri dari Amazon Timestream untuk nama LiveAnalytics operasi, diikuti oleh parameter untuk operasi itu. AWS CLI Mendukung sintaks singkatan untuk nilai parameter, selain. JSON

Gunakan `help` untuk mencantumkan semua perintah yang tersedia di Timestream untuk LiveAnalytics. Sebagai contoh:

```
aws timestream-write help
```

```
aws timestream-query help
```

Anda juga dapat menggunakan `help` untuk mendeskripsikan perintah tertentu dan mempelajari lebih lanjut tentang penggunaannya:

```
aws timestream-write create-database help
```

Misalnya, untuk membuat database:

```
aws timestream-write create-database --database-name myFirstDatabase
```

Untuk membuat tabel dengan penyimpanan magnetik menulis diaktifkan:

```
aws timestream-write create-table \
--database-name metricsdb \
--table-name metrics \
--magnetic-store-write-properties "{\"EnableMagneticStoreWrites\": true}"
```

Untuk menulis data menggunakan catatan ukuran tunggal:

```
aws timestream-write write-records \
--database-name metricsdb \
--table-name metrics \
--common-attributes "{\"Dimensions\":[{\"Name\":\"asset_id\", \"Value\":\"100\"}],
 \"Time\":\"1631051324000\",\"TimeUnit\":\"MILLISECONDS\"}" \
--records "[{\"MeasureName\":\"temperature\", \"MeasureValueType\":\"DOUBLE\",
\"MeasureValue\":\"30\"},{\"MeasureName\":\"windspeed\", \"MeasureValueType\":\"DOUBLE
\",\"MeasureValue\":\"7\"},{\"MeasureName\":\"humidity\", \"MeasureValueType\":\"DOUBLE
\",\"MeasureValue\":\"15\"},{\"MeasureName\":\"brightness\", \"MeasureValueType\":
\"DOUBLE\",\"MeasureValue\":\"17\"}]"
```

Untuk menulis data menggunakan catatan multi-ukuran:

```
# wide model helper method to create Multi-measure records
function ingest_multi_measure_records {
  epoch=`date +%s`
  epoch+=$i

  # multi-measure records
  aws timestream-write write-records \
  --database-name $src_db_wide \
  --table-name $src_tbl_wide \
  --common-attributes "{\"Dimensions\":[{\"Name\":\"device_id\", \
              \"Value\":\"12345678\"},\
            {\"Name\":\"device_type\", \"Value\":\"iPhone\"}, \
            {\"Name\":\"os_version\", \"Value\":\"14.8\"}, \
            {\"Name\":\"region\", \"Value\":\"us-east-1\"} ], \
            \"Time\":\"$epoch\",\"TimeUnit\":\"MILLISECONDS\"}" \
--records "[{\"MeasureName\":\"video_metrics\", \"MeasureValueType\":\"MULTI\", \
  \"MeasureValues\": \
  [{\"Name\":\"video_startup_time\",\"Value\":\"0\",\"Type\":\"BIGINT\"}, \
  {\"Name\":\"rebuffering_ratio\",\"Value\":\"0.5\",\"Type\":\"DOUBLE\"}, \
  {\"Name\":\"video_playback_failures\",\"Value\":\"0\",\"Type\":\"BIGINT\"}, \
  {\"Name\":\"average_frame_rate\",\"Value\":\"0.5\",\"Type\":\"DOUBLE\"}]}]" \
--endpoint-url $ingest_endpoint \
  --region  $region
}

# create 5 records
for i in {100..105};
  do ingest_multi_measure_records $i;
```

```
done
```

Untuk menanyakan tabel:

```
aws timestream-query query \
--query-string "SELECT time, device_id, device_type, os_version,
region, video_startup_time, rebuffering_ratio, video_playback_failures, \
average_frame_rate \
FROM metricsdb.metrics \
where time >= ago (15m)"
```

Untuk membuat kueri terjadwal:

```
aws timestream-query create-scheduled-query \
  --name scheduled_query_name \
  --query-string "select bin(time, 1m) as time, \
          avg(measure_value::double) as avg_cpu, min(measure_value::double) as min_cpu,
 region \
          from $src_db.$src_tbl where measure_name = 'cpu' \
          and time BETWEEN @scheduled_runtime - (interval '5' minute)  AND
 @scheduled_runtime \
          group by region, bin(time, 1m)" \
  --schedule-configuration "{\"ScheduleExpression\":\"$cron_exp\"}" \
  --notification-configuration "{\"SnsConfiguration\":{\"TopicArn\":\"$sns_topic_arn
\"}}" \
  --scheduled-query-execution-role-arn "arn:aws:iam::452360119086:role/
TimestreamSQExecutionRole" \
  --target-configuration "{\"TimestreamConfiguration\":{\
          \"DatabaseName\": \"$dest_db\",\
          \"TableName\": \"$dest_tbl\",\
          \"TimeColumn\":\"time\",\
          \"DimensionMappings\":[{\
            \"Name\": \"region\", \"DimensionValueType\": \"VARCHAR\"
          }],\
          \"MultiMeasureMappings\":{\
            \"TargetMultiMeasureName\": \"mma_name\",
            \"MultiMeasureAttributeMappings\":[{\
              \"SourceColumn\": \"avg_cpu\", \"MeasureValueType\": \"DOUBLE\",
 \"TargetMultiMeasureAttributeName\": \"target_avg_cpu\"
            },\
            { \
              \"SourceColumn\": \"min_cpu\", \"MeasureValueType\": \"DOUBLE\",
 \"TargetMultiMeasureAttributeName\": \"target_min_cpu\"
```

```
          }] \
        }\
        }}" \
--error-report-configuration "{\"S3Configuration\": {\
      \"BucketName\": \"$s3_err_bucket\",\
      \"ObjectKeyPrefix\": \"scherrors\",\
      \"EncryptionOption\": \"SSE_S3\"\
      }\
    }"
```

Menggunakan API

Selain itu SDKs, Amazon Timestream untuk LiveAnalytics menyediakan REST API akses langsung melalui pola penemuan titik akhir. Pola penemuan titik akhir dijelaskan di bawah ini, bersama dengan kasus penggunaannya.

Pola penemuan titik akhir

Karena Timestream Live Analytics SDKs dirancang untuk bekerja secara transparan dengan arsitektur layanan, termasuk pengelolaan dan pemetaan titik akhir layanan, disarankan agar Anda menggunakannya untuk sebagian besar aplikasi. SDKs Namun, ada beberapa contoh di mana penggunaan Timestream untuk pola penemuan LiveAnalytics REST API titik akhir diperlukan:

- Anda menggunakan VPCendpoints (AWS PrivateLink) dengan Timestream untuk LiveAnalytics
- Aplikasi Anda menggunakan bahasa pemrograman yang belum memiliki SDK dukungan
- Anda memerlukan kontrol yang lebih baik atas implementasi sisi klien

Bagian ini mencakup informasi tentang cara kerja pola penemuan titik akhir, cara menerapkan pola penemuan titik akhir, dan catatan penggunaan. Pilih topik di bawah ini untuk mempelajari lebih lanjut.

Topik

- Cara kerja pola penemuan titik akhir
- Menerapkan pola penemuan titik akhir

Cara kerja pola penemuan titik akhir

Timestream dibangun menggunakan arsitektur seluler untuk memastikan properti penskalaan dan isolasi lalu lintas yang lebih baik. Karena setiap akun pelanggan dipetakan ke sel tertentu di suatu

wilayah, aplikasi Anda harus menggunakan titik akhir khusus sel yang benar tempat akun Anda telah dipetakan. Saat menggunakanSDKs, pemetaan ini ditangani secara transparan untuk Anda dan Anda tidak perlu mengelola titik akhir khusus sel. Namun, saat mengakses langsung RESTAPI, Anda perlu mengelola dan memetakan titik akhir yang benar sendiri. Proses ini, pola penemuan titik akhir, dijelaskan di bawah ini:

1. Pola penemuan titik akhir dimulai dengan panggilan untuk `DescribeEndpoints` tindakan (dijelaskan di `DescribeEndpoints`bagian).
2. Titik akhir harus di-cache dan digunakan kembali untuk jumlah waktu yang ditentukan oleh nilai () yang dikembalikan time-to-live (theTTL). `CachePeriodInMinutes` Panggilan ke Timestream Live Analytics kemudian API dapat dilakukan selama durasi. TTL
3. Setelah TTL kedaluwarsa, panggilan baru DescribeEndpoints harus dilakukan untuk menyegarkan titik akhir (dengan kata lain, mulai dari awal pada Langkah 1).

> **Note**
>
> Sintaks, parameter, dan informasi penggunaan lainnya untuk `DescribeEndpoints` tindakan dijelaskan dalam APIReferensi. Perhatikan bahwa `DescribeEndpoints` tindakan tersedia melalui keduanyaSDKs, dan identik untuk masing-masing.

Untuk implementasi pola penemuan titik akhir, lihatMenerapkan pola penemuan titik akhir.

Menerapkan pola penemuan titik akhir

Untuk menerapkan pola penemuan titik akhir, pilih API (Tulis atau Kueri), buat DescribeEndpointspermintaan, dan gunakan titik akhir yang dikembalikan selama durasi TTL nilai yang dikembalikan. Prosedur implementasi dijelaskan di bawah ini.

> **Note**
>
> Pastikan Anda terbiasa dengan catatan penggunaan.

Prosedur implementasi

1. Dapatkan titik akhir untuk API Anda ingin melakukan panggilan terhadap (Tulis atau Kueri). menggunakan `DescribeEndpoints`permintaan.

a. Buat permintaan `DescribeEndpoints`yang sesuai dengan minat (Tulis atau Kueri) menggunakan salah satu dari dua titik akhir yang dijelaskan di bawah ini. API Tidak ada parameter input untuk permintaan tersebut. Pastikan Anda membaca catatan di bawah ini.

   MenulisSDK:

   ```
   ingest.timestream.<region>.amazonaws.com
   ```

   PermintaanSDK:

   ```
   query.timestream.<region>.amazonaws.com
   ```

   Contoh CLI panggilan untuk wilayah `us-east-1` berikut.

   ```
   REGION_ENDPOINT="https://query.timestream.us-east-1.amazonaws.com"
   REGION=us-east-1
   aws timestream-write describe-endpoints \
   --endpoint-url $REGION_ENDPOINT \
   --region $REGION
   ```

   > **Note**
   >
   > Header HTTP “Host” juga harus berisi API titik akhir. Permintaan akan gagal jika header tidak diisi. Ini adalah persyaratan standar untuk semua permintaan HTTP /1.1. Jika Anda menggunakan HTTP pustaka yang mendukung 1.1 atau yang lebih baru, HTTP pustaka harus secara otomatis mengisi header untuk Anda.

   > **Note**
   >
   > Pengganti *<region>* dengan pengidentifikasi wilayah untuk wilayah tempat permintaan sedang dibuat, mis. `us-east-1`

b. Parse respon untuk mengekstrak endpoint (s), dan TTL nilai cache (s). Responsnya adalah array dari satu atau lebih `Endpoint`objek. Setiap `Endpoint` objek berisi alamat titik akhir (`Address`) dan TTL untuk titik akhir itu (`CachePeriodInMinutes`).

2. Cache titik akhir hingga yang ditentukanTTL.
3. Ketika TTL kedaluwarsa, ambil titik akhir baru dengan memulai dari awal pada langkah 1 Implementasi.

Catatan penggunaan untuk pola penemuan titik akhir

- DescribeEndpointsTindakan ini adalah satu-satunya tindakan yang dikenali oleh titik akhir regional Timestream Live Analytics.
- Respons berisi daftar titik akhir untuk membuat API panggilan Timestream Live Analytics melawan.
- Pada respons yang berhasil, setidaknya harus ada satu titik akhir dalam daftar. Jika ada lebih dari satu titik akhir dalam daftar, salah satu dari mereka sama-sama dapat digunakan untuk API panggilan, dan pemanggil dapat memilih titik akhir untuk digunakan secara acak.
- Selain DNS alamat titik akhir, setiap titik akhir dalam daftar akan menentukan waktu untuk hidup (TTL) yang diizinkan untuk menggunakan titik akhir yang ditentukan dalam menit.
- Titik akhir harus di-cache dan digunakan kembali untuk jumlah waktu yang ditentukan oleh TTL nilai yang dikembalikan (dalam menit). Setelah TTL kedaluwarsa, panggilan baru DescribeEndpointsharus dilakukan untuk menyegarkan titik akhir yang akan digunakan, karena titik akhir tidak akan berfungsi lagi setelah kedaluwarsaTTL.

Menggunakan AWS SDKs

Anda dapat mengakses Amazon Timestream menggunakan file. AWS SDKs Timestream mendukung dua SDKs per bahasa; yaitu, Write SDK dan QuerySDK. Tulis SDK digunakan untuk melakukan CRUD operasi dan menyisipkan data deret waktu Anda ke Timestream. Kueri SDK digunakan untuk menanyakan data deret waktu yang ada yang disimpan di Timestream.

Setelah Anda menyelesaikan prasyarat yang diperlukan untuk pilihan AndaSDK, Anda dapat memulai dengan. Sampel Kode

Topik

- Java
- Jawa v2
- Go
- Python
- Node.js

- .NET

Java

Untuk memulai dengan Java 1.0 SDK dan Amazon Timestream, lengkapi prasyarat, yang dijelaskan di bawah ini.

Setelah Anda menyelesaikan prasyarat yang diperlukan untuk JavaSDK, Anda dapat memulai dengan. Sampel Kode

Prasyarat

Sebelum Anda memulai dengan Java, Anda harus melakukan hal berikut:

1. Ikuti petunjuk AWS penyiapan diMengakses Timestream untuk LiveAnalytics.
2. Siapkan lingkungan pengembangan Java dengan mengunduh dan menginstal yang berikut ini:
   - Java SE Development Kit 8 (seperti Amazon Corretto 8).
   - Java IDE (seperti Eclipse atau IntelliJ).

     Untuk informasi selengkapnya, lihat Memulai dengan AWS SDK for Java
3. Konfigurasikan AWS kredensil dan Wilayah Anda untuk pengembangan:
   - Siapkan AWS kredensil keamanan Anda untuk digunakan dengan file. AWS SDK for Java
   - Tetapkan AWS Wilayah Anda untuk menentukan Timestream default untuk titik LiveAnalytics akhir.

Menggunakan Apache Maven

Anda dapat menggunakan Apache Maven untuk mengkonfigurasi dan membangun proyek. AWS SDK for Java

> **Note**
>
> Untuk menggunakan Apache Maven, pastikan Java SDK dan runtime Anda 1,8 atau lebih tinggi.

Anda dapat mengkonfigurasi AWS SDK sebagai ketergantungan Maven seperti yang dijelaskan dalam Menggunakan SDK dengan Apache Maven.

Anda dapat menjalankan kompilasi dan menjalankan kode sumber Anda dengan perintah berikut:

```
mvn clean compile
mvn exec:java -Dexec.mainClass=<your source code Main class>
```

> **Note**
>
> `<your source code Main class>`adalah jalur ke kelas utama kode sumber Java Anda.

Mengatur AWS kredensi Anda

Ini AWS SDK for Javamengharuskan Anda memberikan AWS kredensil ke aplikasi Anda saat runtime. Contoh kode dalam panduan ini mengasumsikan bahwa Anda menggunakan file AWS kredensial, seperti yang dijelaskan dalam Mengatur AWS Kredensial dan Wilayah untuk Pengembangan dalam Panduan Pengembang.AWS SDK for Java

Berikut ini adalah contoh file AWS kredensial bernama`~/.aws/credentials`, di mana karakter tilde (~) mewakili direktori home Anda.

```
[default]
aws_access_key_id = AWS access key ID goes here
aws_secret_access_key = Secret key goes here
```

Jawa v2

Untuk memulai dengan Java 2.0 SDK dan Amazon Timestream, lengkapi prasyarat, yang dijelaskan di bawah ini.

Setelah Anda menyelesaikan prasyarat yang diperlukan untuk Java 2.0SDK, Anda dapat memulai dengan. Sampel Kode

Prasyarat

Sebelum Anda memulai dengan Java, Anda harus melakukan hal berikut:

1. Ikuti petunjuk AWS penyiapan diMengakses Timestream untuk LiveAnalytics.

2. Anda dapat mengkonfigurasi AWS SDK sebagai ketergantungan Maven seperti yang dijelaskan dalam Menggunakan SDK dengan Apache Maven.
3. Siapkan lingkungan pengembangan Java dengan mengunduh dan menginstal yang berikut ini:

   - Java SE Development Kit 8 (seperti Amazon Corretto 8).
   - Java IDE (seperti Eclipse atau IntelliJ).

     Untuk informasi selengkapnya, lihat Memulai dengan AWS SDK for Java

Menggunakan Apache Maven

Anda dapat menggunakan Apache Maven untuk mengkonfigurasi dan membangun proyek. AWS SDK for Java

> **Note**
>
> Untuk menggunakan Apache Maven, pastikan Java SDK dan runtime Anda 1,8 atau lebih tinggi.

Anda dapat mengkonfigurasi AWS SDK sebagai ketergantungan Maven seperti yang dijelaskan dalam Menggunakan SDK dengan Apache Maven. Perubahan yang diperlukan pada file pom.xml dijelaskan di sini.

Anda dapat menjalankan kompilasi dan menjalankan kode sumber Anda dengan perintah berikut:

```
mvn clean compile
mvn exec:java -Dexec.mainClass=<your source code Main class>
```

> **Note**
>
> `<your source code Main class>`adalah jalur ke kelas utama kode sumber Java Anda.

Go

Untuk memulai dengan Go SDK dan Amazon Timestream, lengkapi prasyarat, yang dijelaskan di bawah ini.

Setelah Anda menyelesaikan prasyarat yang diperlukan untuk GoSDK, Anda dapat memulai dengan. Sampel Kode

Prasyarat

1. Unduh GO SDK 1.14.
2. Konfigurasikan GO SDK.
3. Bangun klien Anda.

Python

Untuk memulai dengan Python SDK dan Amazon Timestream, lengkapi prasyarat, yang dijelaskan di bawah ini.

Setelah Anda menyelesaikan prasyarat yang diperlukan untuk PythonSDK, Anda dapat memulai dengan. Sampel Kode

Prasyarat

Untuk menggunakan Python, instal dan konfigurasikan Boto3, ikuti instruksi di sini.

Node.js

Untuk memulai dengan Node.js SDK dan Amazon Timestream, lengkapi prasyarat, yang dijelaskan di bawah ini.

Setelah Anda menyelesaikan prasyarat yang diperlukan untuk Node.jsSDK, Anda dapat memulai dengan. Sampel Kode

Prasyarat

Sebelum Anda memulai dengan Node.js, Anda harus melakukan hal berikut:

1. Instal Node.js.
2. Instal AWS SDK untuk JavaScript.

.NET

Untuk memulai dengan. NETSDKdan Amazon Timestream, lengkapi prasyarat, yang dijelaskan di bawah ini.

Setelah Anda menyelesaikan prasyarat yang diperlukan untuk. NETSDK, Anda bisa memulai dengan Sampel Kode.

Prasyarat

Sebelum Anda memulai. NET, instal NuGet paket yang diperlukan dan pastikan AWSSDK versi.Core adalah 3.3.107 atau yang lebih baru dengan menjalankan perintah berikut:

```
dotnet add package AWSSDK.Core
dotnet add package AWSSDK.TimestreamWrite
dotnet add package AWSSDK.TimestreamQuery
```

Memulai

Bagian ini mencakup tutorial untuk membantu Anda memulai dengan Amazon Timestream Live Analytics, serta instruksi untuk menyiapkan aplikasi sampel yang berfungsi penuh. Anda dapat memulai dengan tutorial atau contoh aplikasi dengan memilih salah satu link di bawah ini.

Topik

- Tutorial
- Aplikasi sampel

Tutorial

Tutorial ini menunjukkan cara membuat database yang diisi dengan kumpulan data sampel dan menjalankan query sampel. Kumpulan data sampel yang digunakan dalam tutorial ini sering terlihat di IoT dan DevOps skenario. Kumpulan data IoT berisi data deret waktu seperti kecepatan, lokasi, dan beban truk, untuk merampingkan manajemen armada dan mengidentifikasi peluang pengoptimalan. Kumpulan DevOps data berisi metrik EC2 instance sepertiCPU, jaringan, dan pemanfaatan memori untuk meningkatkan kinerja dan ketersediaan aplikasi. Berikut adalah tutorial video untuk instruksi yang dijelaskan di bagian ini.

Ikuti langkah-langkah berikut untuk membuat database yang diisi dengan kumpulan data sampel dan menjalankan kueri sampel menggunakan AWS Konsol:

Menggunakan konsol

Ikuti langkah-langkah berikut untuk membuat database yang diisi dengan kumpulan data sampel dan menjalankan kueri sampel menggunakan AWS Konsol:

1. Buka AWS Konsol.
2. Di panel navigasi, pilih Databases
3. Klik Buat database.
4. Pada halaman buat database, masukkan yang berikut ini:
   - Pilih konfigurasi —Pilih database Sampel.
   - Nama —Masukkan nama database pilihan Anda.

     > **Note**
     >
     > Setelah membuat database dengan kumpulan data sampel, untuk menggunakan kueri sampel yang tersedia di konsol, Anda dapat menyesuaikan nama database yang direferensikan dalam kueri agar sesuai dengan nama database yang Anda masukkan di sini. Ada kueri sampel untuk setiap kombinasi kumpulan data sampel dan jenis catatan deret waktu.

   - Pilih kumpulan data sampel —Pilih DevOpsIoT dan.
   - Pilih jenis catatan deret waktu —Pilih catatan Multi-ukuran.
   - Klik Buat database untuk membuat database yang berisi dua tabel yang diisi dengan data sampel. Nama tabel untuk kumpulan data sampel dengan catatan multi-ukuran adalah `DevOpsMulti` dan`IoTMulti`. Nama tabel untuk kumpulan data sampel dengan catatan ukuran tunggal adalah dan. `DevOps IoT`
5. Di panel navigasi, pilih Editor kueri
6. Pilih Contoh kueri dari menu atas.
7. Klik salah satu contoh kueri untuk kumpulan data yang Anda pilih saat membuat database sampel. Ini akan membawa Anda kembali ke editor kueri dengan editor diisi dengan kueri sampel.
8. Sesuaikan nama database untuk kueri sampel.
9. Klik Jalankan untuk menjalankan kueri dan melihat hasil kueri.

Menggunakan SDKs

Timestream Live Analytics menyediakan aplikasi sampel yang berfungsi penuh yang menunjukkan cara membuat database dan tabel, mengisi tabel dengan ~126K baris data sampel, dan menjalankan kueri sampel. Aplikasi sampel tersedia GitHubuntuk Java, Python, Node.js, Go, dan. NET.

1. Kloning aplikasi contoh Timestream Live Analytics GitHub repositori mengikuti instruksi dari. GitHub
2. Konfigurasikan AWS SDK untuk terhubung ke Amazon Timestream Live Analytics mengikuti petunjuk yang dijelaskan di. Menggunakan AWS SDKs
3. Kompilasi dan jalankan aplikasi sampel menggunakan instruksi di bawah ini:

   - Petunjuk untuk aplikasi sampel Java.
   - Petunjuk untuk aplikasi sampel Java v2.
   - Petunjuk untuk aplikasi sampel Go.
   - Petunjuk untuk aplikasi sampel Python.
   - Petunjuk untuk aplikasi sampel Node.js.
   - Instruksi untuk. NETaplikasi sampel.

Aplikasi sampel

Timestream dikirimkan dengan aplikasi sampel yang berfungsi penuh yang menunjukkan cara membuat database dan tabel, mengisi tabel dengan ~ 126K baris data sampel, dan menjalankan kueri sampel. Ikuti langkah-langkah di bawah ini untuk memulai aplikasi contoh dalam bahasa yang didukung:

Java

1. Kloning GitHub repositori Timestream untuk LiveAnalytics contoh aplikasi mengikuti instruksi dari. GitHub
2. Konfigurasikan AWS SDK untuk terhubung ke Timestream untuk LiveAnalytics mengikuti petunjuk yang dijelaskan dalam Memulai dengan Java.
3. Jalankan aplikasi sampel Java mengikuti instruksi yang dijelaskan di sini

Java v2

1. Kloning GitHub repositori Timestream untuk LiveAnalytics contoh aplikasi mengikuti instruksi dari. GitHub
2. Konfigurasikan AWS SDK untuk menyambung ke Amazon Timestream untuk LiveAnalytics mengikuti petunjuk yang dijelaskan dalam Memulai dengan. Jawa v2
3. Jalankan aplikasi sampel Java 2.0 mengikuti petunjuk yang dijelaskan di sini

Go

1. Kloning GitHub repositori Timestream untuk LiveAnalytics contoh aplikasi mengikuti instruksi dari. GitHub
2. Konfigurasikan AWS SDK untuk menyambung ke Amazon Timestream untuk LiveAnalytics mengikuti petunjuk yang dijelaskan dalam Memulai dengan. Go
3. Jalankan aplikasi sampel Go mengikuti instruksi yang dijelaskan di sini

Python

1. Kloning GitHub repositori Timestream untuk LiveAnalytics contoh aplikasi mengikuti instruksi dari. GitHub
2. Konfigurasikan AWS SDK untuk menyambung ke Amazon Timestream untuk LiveAnalytics mengikuti petunjuk yang dijelaskan di. Python
3. Jalankan contoh aplikasi Python mengikuti petunjuk yang dijelaskan di sini

Node.js

1. Kloning GitHub repositori Timestream untuk LiveAnalytics contoh aplikasi mengikuti instruksi dari. GitHub
2. Konfigurasikan AWS SDK untuk menyambung ke Amazon Timestream untuk LiveAnalytics mengikuti petunjuk yang dijelaskan dalam Memulai dengan. Node.js
3. Jalankan aplikasi sampel Node.js mengikuti instruksi yang dijelaskan di sini

.NET

1. Kloning GitHub repositori Timestream untuk LiveAnalytics contoh aplikasi mengikuti instruksi dari. GitHub
2. Konfigurasikan AWS SDK untuk menyambung ke Amazon Timestream untuk LiveAnalytics mengikuti petunjuk yang dijelaskan dalam Memulai dengan. .NET
3. Jalankan. NETcontoh aplikasi mengikuti instruksi yang dijelaskan di sini

Sampel Kode

Anda dapat mengakses Amazon Timestream menggunakan file. AWS SDKs Timestream mendukung dua SDKs per bahasa; yaitu, Write SDK dan QuerySDK. Tulis SDK digunakan untuk melakukan CRUD operasi dan menyisipkan data deret waktu Anda ke Timestream. Kueri SDK digunakan untuk menanyakan data deret waktu yang ada yang disimpan di Timestream. Pilih topik dari daftar di bawah ini untuk detail selengkapnya, termasuk contoh kode untuk masing-masing yang didukungSDKs.

Topik

- Jalankan kueri terjadwal
- Perbarui kueri terjadwal
- Hapus kueri terjadwal

Tulis SDK klien

Anda dapat menggunakan cuplikan kode berikut untuk membuat klien Timestream untuk Write. SDK Tulis SDK digunakan untuk melakukan CRUD operasi dan menyisipkan data deret waktu Anda ke Timestream.

> **Note**
>
> Cuplikan kode ini didasarkan pada contoh aplikasi lengkap. GitHub Untuk informasi selengkapnya tentang cara memulai dengan contoh aplikasi, lihatAplikasi sampel.

Java

```
    private static AmazonTimestreamWrite buildWriteClient() {
        final ClientConfiguration clientConfiguration = new ClientConfiguration()
                .withMaxConnections(5000)
                .withRequestTimeout(20 * 1000)
                .withMaxErrorRetry(10);

        return AmazonTimestreamWriteClientBuilder
                .standard()
                .withRegion("us-east-1")
                .withClientConfiguration(clientConfiguration)
                .build();
    }
```

Java v2

```
    private static TimestreamWriteClient buildWriteClient() {
        ApacheHttpClient.Builder httpClientBuilder =
                ApacheHttpClient.builder();
        httpClientBuilder.maxConnections(5000);

        RetryPolicy.Builder retryPolicy =
                RetryPolicy.builder();
```

```
        retryPolicy.numRetries(10);

        ClientOverrideConfiguration.Builder overrideConfig =
                ClientOverrideConfiguration.builder();
        overrideConfig.apiCallAttemptTimeout(Duration.ofSeconds(20));
        overrideConfig.retryPolicy(retryPolicy.build());

        return TimestreamWriteClient.builder()
                .httpClientBuilder(httpClientBuilder)
                .overrideConfiguration(overrideConfig.build())
                .region(Region.US_EAST_1)
                .build();
    }
```

Go

```
tr := &http.Transport{
        ResponseHeaderTimeout: 20 * time.Second,
        // Using DefaultTransport values for other parameters: https://golang.org/
pkg/net/http/#RoundTripper
        Proxy: http.ProxyFromEnvironment,
        DialContext: (&net.Dialer{
            KeepAlive: 30 * time.Second,
            DualStack: true,
            Timeout:   30 * time.Second,
        }).DialContext,
        MaxIdleConns:          100,
        IdleConnTimeout:       90 * time.Second,
        TLSHandshakeTimeout:   10 * time.Second,
        ExpectContinueTimeout: 1 * time.Second,
    }

    // So client makes HTTP/2 requests
    http2.ConfigureTransport(tr)

    sess, err := session.NewSession(&aws.Config{ Region: aws.String("us-east-1"),
 MaxRetries: aws.Int(10), HTTPClient: &http.Client{ Transport: tr }})
    writeSvc := timestreamwrite.New(sess)
```

Python

```
write_client = session.client('timestream-write', config=Config(read_timeout=20,
 max_pool_connections = 5000, retries={'max_attempts': 10}))
```

Node.js

Cuplikan berikut digunakan AWS SDK untuk JavaScript v3. Untuk informasi selengkapnya tentang cara menginstal klien dan penggunaan, lihat Timestream Write Client - AWS SDK untuk JavaScript v3.

Impor perintah tambahan ditampilkan di sini. `CreateDatabaseCommand`Impor tidak diperlukan untuk membuat klien.

```
import { TimestreamWriteClient, CreateDatabaseCommand } from "@aws-sdk/client-
timestream-write";
const writeClient = new TimestreamWriteClient({ region: "us-east-1" });
```

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada contoh aplikasi di Node.js sampel Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
var https = require('https');
var agent = new https.Agent({
    maxSockets: 5000
});
writeClient = new AWS.TimestreamWrite({
        maxRetries: 10,
        httpOptions: {
            timeout: 20000,
            agent: agent
        }
    });
```

.NET

```
var writeClientConfig = new AmazonTimestreamWriteConfig
{
    RegionEndpoint = RegionEndpoint.USEast1,
    Timeout = TimeSpan.FromSeconds(20),
    MaxErrorRetry = 10
};

var writeClient = new AmazonTimestreamWriteClient(writeClientConfig);
```

Kami menyarankan Anda menggunakan konfigurasi berikut.

- Atur hitungan SDK coba lagi ke`10`.
- Gunakan `SDK DEFAULT_BACKOFF_STRATEGY`.
- Setel `RequestTimeout` ke `20` detik.
- Atur koneksi maks ke `5000` atau lebih tinggi.

Kueri SDK klien

Anda dapat menggunakan cuplikan kode berikut untuk membuat klien Timestream untuk Query. SDK Kueri SDK digunakan untuk menanyakan data deret waktu yang ada yang disimpan di Timestream.

> **Note**
>
> Cuplikan kode ini didasarkan pada contoh aplikasi lengkap. GitHub Untuk informasi selengkapnya tentang cara memulai dengan contoh aplikasi, lihatAplikasi sampel.

Java

```
    private static AmazonTimestreamQuery buildQueryClient() {
        AmazonTimestreamQuery client =
AmazonTimestreamQueryClient.builder().withRegion("us-east-1").build();
        return client;
    }
```

Java v2

```
    private static TimestreamQueryClient buildQueryClient() {
        return TimestreamQueryClient.builder()
                .region(Region.US_EAST_1)
                .build();
    }
```

Go

```
sess, err := session.NewSession(&aws.Config{Region: aws.String("us-east-1")})
```

Python

```
query_client = session.client('timestream-query')
```

Node.js

Cuplikan berikut digunakan AWS SDK untuk JavaScript v3. Untuk informasi selengkapnya tentang cara menginstal klien dan penggunaan, lihat Klien Kueri Timestream -,AWS SDK untuk JavaScript v3.

Impor perintah tambahan ditampilkan di sini. `QueryCommand`Impor tidak diperlukan untuk membuat klien.

```
import { TimestreamQueryClient, QueryCommand } from "@aws-sdk/client-timestream-
query";
const queryClient = new TimestreamQueryClient({ region: "us-east-1" });
```

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada aplikasi sampel Node.js Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
queryClient = new AWS.TimestreamQuery();
```

.NET

```
var queryClientConfig = new AmazonTimestreamQueryConfig
{
    RegionEndpoint = RegionEndpoint.USEast1
};

var queryClient = new AmazonTimestreamQueryClient(queryClientConfig);
```

Buat basis data

Anda dapat menggunakan cuplikan kode berikut untuk membuat database.

> **Note**
>
> Cuplikan kode ini didasarkan pada contoh aplikasi lengkap. GitHub Untuk informasi selengkapnya tentang cara memulai dengan contoh aplikasi, lihatAplikasi sampel.

Java

```
  public void createDatabase() {
       System.out.println("Creating database");
       CreateDatabaseRequest request = new CreateDatabaseRequest();
       request.setDatabaseName(DATABASE_NAME);
       try {
           amazonTimestreamWrite.createDatabase(request);
           System.out.println("Database [" + DATABASE_NAME + "] created
successfully");
       } catch (ConflictException e) {
           System.out.println("Database [" + DATABASE_NAME + "] exists. Skipping
database creation");
       }
   }
```

Java v2

```
   public void createDatabase() {
       System.out.println("Creating database");
       CreateDatabaseRequest request =
CreateDatabaseRequest.builder().databaseName(DATABASE_NAME).build();
       try {
           timestreamWriteClient.createDatabase(request);
           System.out.println("Database [" + DATABASE_NAME + "] created
successfully");
       } catch (ConflictException e) {
           System.out.println("Database [" + DATABASE_NAME + "] exists. Skipping
database creation");
       }
   }
```

Go

```
// Create database.
    createDatabaseInput := &timestreamwrite.CreateDatabaseInput{
        DatabaseName: aws.String(*databaseName),
    }

    _, err = writeSvc.CreateDatabase(createDatabaseInput)

    if err != nil {
        fmt.Println("Error:")
```

```
        fmt.Println(err)
    } else {
        fmt.Println("Database successfully created")
    }

    fmt.Println("Describing the database, hit enter to continue")
```

Python

```
    def create_database(self):
        print("Creating Database")
        try:
            self.client.create_database(DatabaseName=Constant.DATABASE_NAME)
            print("Database [%s] created successfully." % Constant.DATABASE_NAME)
        except self.client.exceptions.ConflictException:
            print("Database [%s] exists. Skipping database creation" %
 Constant.DATABASE_NAME)
        except Exception as err:
            print("Create database failed:", err)
```

Node.js

Cuplikan berikut digunakan AWS SDK untuk JavaScript v3. Untuk informasi selengkapnya tentang cara menginstal klien dan penggunaan, lihat Timestream Write Client - AWS SDK untuk JavaScript v3.

Lihat juga kelas CreateDatabaseCommand dan CreateDatabase

```
import { TimestreamWriteClient, CreateDatabaseCommand } from "@aws-sdk/client-
timestream-write";
const writeClient = new TimestreamWriteClient({ region: "us-east-1" });

const params = {
    DatabaseName: "testDbFromNode"
};

const command = new CreateDatabaseCommand(params);

try {
    const data = await writeClient.send(command);
    console.log(`Database ${data.Database.DatabaseName} created successfully`);
} catch (error) {
    if (error.code === 'ConflictException') {
```

```
        console.log(`Database ${params.DatabaseName} already exists. Skipping
 creation.`);
    } else {
        console.log("Error creating database", error);
    }
}
```

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada aplikasi sampel Node.js Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
async function createDatabase() {
    console.log("Creating Database");
    const params = {
        DatabaseName: constants.DATABASE_NAME
    };

    const promise = writeClient.createDatabase(params).promise();

    await promise.then(
        (data) => {
            console.log(`Database ${data.Database.DatabaseName} created
 successfully`);
        },
        (err) => {
            if (err.code === 'ConflictException') {
                console.log(`Database ${params.DatabaseName} already exists.
 Skipping creation.`);
            } else {
                console.log("Error creating database", err);
            }
        }
    );
}
```

.NET

```
        public async Task CreateDatabase()
        {
            Console.WriteLine("Creating Database");

            try
            {
                var createDatabaseRequest = new CreateDatabaseRequest
```

```
                {
                    DatabaseName = Constants.DATABASE_NAME
                };
                CreateDatabaseResponse response = await
writeClient.CreateDatabaseAsync(createDatabaseRequest);
                Console.WriteLine($"Database {Constants.DATABASE_NAME} created");
            }
            catch (ConflictException)
            {
                Console.WriteLine("Database already exists.");
            }
            catch (Exception e)
            {
                Console.WriteLine("Create database failed:" + e.ToString());
            }

        }
```

Jelaskan database

Anda dapat menggunakan cuplikan kode berikut untuk mendapatkan informasi tentang atribut database yang baru Anda buat.

> **Note**
>
> Cuplikan kode ini didasarkan pada contoh aplikasi lengkap. GitHub Untuk informasi selengkapnya tentang cara memulai dengan contoh aplikasi, lihatAplikasi sampel.

Java

```
   public void describeDatabase() {
        System.out.println("Describing database");
        final DescribeDatabaseRequest describeDatabaseRequest = new
DescribeDatabaseRequest();
        describeDatabaseRequest.setDatabaseName(DATABASE_NAME);
        try {
            DescribeDatabaseResult result =
amazonTimestreamWrite.describeDatabase(describeDatabaseRequest);
            final Database databaseRecord = result.getDatabase();
            final String databaseId = databaseRecord.getArn();
```

```
            System.out.println("Database " + DATABASE_NAME + " has id " +
 databaseId);
        } catch (final Exception e) {
            System.out.println("Database doesn't exist = " + e);
            throw e;
        }
    }
```

Java v2

```
    public void describeDatabase() {
        System.out.println("Describing database");
        final DescribeDatabaseRequest describeDatabaseRequest =
 DescribeDatabaseRequest.builder()
                .databaseName(DATABASE_NAME).build();
        try {
            DescribeDatabaseResponse response =
 timestreamWriteClient.describeDatabase(describeDatabaseRequest);
            final Database databaseRecord = response.database();
            final String databaseId = databaseRecord.arn();
            System.out.println("Database " + DATABASE_NAME + " has id " +
 databaseId);
        } catch (final Exception e) {
            System.out.println("Database doesn't exist = " + e);
            throw e;
        }
    }
```

Go

```
describeDatabaseOutput, err := writeSvc.DescribeDatabase(describeDatabaseInput)

    if err != nil {
        fmt.Println("Error:")
        fmt.Println(err)
    } else {
        fmt.Println("Describe database is successful, below is the output:")
        fmt.Println(describeDatabaseOutput)
    }
```

Python

```
    def describe_database(self):
```

```
        print("Describing database")
        try:
            result =
 self.client.describe_database(DatabaseName=Constant.DATABASE_NAME)
            print("Database [%s] has id [%s]" % (Constant.DATABASE_NAME,
 result['Database']['Arn']))
        except self.client.exceptions.ResourceNotFoundException:
            print("Database doesn't exist")
        except Exception as err:
            print("Describe database failed:", err)
```

Node.js

Cuplikan berikut digunakan AWS SDK untuk JavaScript v3. Untuk informasi selengkapnya tentang cara menginstal klien dan penggunaan, lihat Timestream Write Client - AWS SDK untuk JavaScript v3.

Lihat juga kelas DescribeDatabaseCommand dan DescribeDatabase

```
import { TimestreamWriteClient, DescribeDatabaseCommand } from "@aws-sdk/client-
timestream-write";
const writeClient = new TimestreamWriteClient({ region: "us-east-1" });

const params = {
    DatabaseName: "testDbFromNode"
};

const command = new DescribeDatabaseCommand(params);

try {
    const data = await writeClient.send(command);
    console.log(`Database ${data.Database.DatabaseName} has id
 ${data.Database.Arn}`);
} catch (error) {
    if (error.code === 'ResourceNotFoundException') {
        console.log("Database doesn't exist.");
    } else {
        console.log("Describe database failed.", error);
        throw error;
    }
}
```

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada contoh aplikasi di Node.js sampel Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
async function describeDatabase () {
    console.log("Describing Database");
    const params = {
        DatabaseName: constants.DATABASE_NAME
    };

    const promise = writeClient.describeDatabase(params).promise();

    await promise.then(
        (data) => {
            console.log(`Database ${data.Database.DatabaseName} has id
 ${data.Database.Arn}`);
        },
        (err) => {
            if (err.code === 'ResourceNotFoundException') {
                console.log("Database doesn't exist.");
            } else {
                console.log("Describe database failed.", err);
                throw err;
            }
        }
    );
}
```

.NET

```
        public async Task DescribeDatabase()
        {
            Console.WriteLine("Describing Database");

            try
            {
                var describeDatabaseRequest = new DescribeDatabaseRequest
                {
                    DatabaseName = Constants.DATABASE_NAME
                };
                DescribeDatabaseResponse response = await
 writeClient.DescribeDatabaseAsync(describeDatabaseRequest);
                Console.WriteLine($"Database {Constants.DATABASE_NAME} has id:
{response.Database.Arn}");
```

```
            }
            catch (ResourceNotFoundException)
            {
                Console.WriteLine("Database does not exist.");
            }
            catch (Exception e)
            {
                Console.WriteLine("Describe database failed:" + e.ToString());
            }

        }
```

Perbarui basis data

Anda dapat menggunakan cuplikan kode berikut untuk memperbarui database Anda.

> **Note**
>
> Cuplikan kode ini didasarkan pada contoh aplikasi lengkap. GitHub Untuk informasi selengkapnya tentang cara memulai dengan contoh aplikasi, lihatAplikasi sampel.

Java

```
   public void updateDatabase(String kmsId) {
       System.out.println("Updating kmsId to " + kmsId);
       UpdateDatabaseRequest request = new UpdateDatabaseRequest();
       request.setDatabaseName(DATABASE_NAME);
       request.setKmsKeyId(kmsId);
       try {
           UpdateDatabaseResult result =
amazonTimestreamWrite.updateDatabase(request);
           System.out.println("Update Database complete");
       } catch (final ValidationException e) {
           System.out.println("Update database failed:");
           e.printStackTrace();
       } catch (final ResourceNotFoundException e) {
           System.out.println("Database " + DATABASE_NAME + " doesn't exist = " +
e);
       } catch (final Exception e) {
```

```
            System.out.println("Could not update Database " + DATABASE_NAME + " = "
 + e);
            throw e;
        }
    }
```

Java v2

```
    public void updateDatabase(String kmsKeyId) {

        if (kmsKeyId == null) {
            System.out.println("Skipping UpdateDatabase because KmsKeyId was not
 given");
            return;
        }

        System.out.println("Updating database");

        UpdateDatabaseRequest request = UpdateDatabaseRequest.builder()
                .databaseName(DATABASE_NAME)
                .kmsKeyId(kmsKeyId)
                .build();
        try {
            timestreamWriteClient.updateDatabase(request);
            System.out.println("Database [" + DATABASE_NAME + "] updated
 successfully with kmsKeyId " + kmsKeyId);
        } catch (ResourceNotFoundException e) {
            System.out.println("Database [" + DATABASE_NAME + "] does not exist.
 Skipping UpdateDatabase");
        } catch (Exception e) {
            System.out.println("UpdateDatabase failed: " + e);
        }
    }
```

Go

```
// Update Database.
        updateDatabaseInput := &timestreamwrite.UpdateDatabaseInput {
            DatabaseName: aws.String(*databaseName),
            KmsKeyId: aws.String(*kmsKeyId),
        }

        updateDatabaseOutput, err := writeSvc.UpdateDatabase(updateDatabaseInput)
```

```
        if err != nil {
            fmt.Println("Error:")
            fmt.Println(err)
        } else {
            fmt.Println("Update database is successful, below is the output:")
            fmt.Println(updateDatabaseOutput)
        }
```

Python

```
    def update_database(self, kms_id):
        print("Updating database")
        try:
            result =
 self.client.update_database(DatabaseName=Constant.DATABASE_NAME, KmsKeyId=kms_id)
            print("Database [%s] was updated to use kms [%s] successfully" %
 (Constant.DATABASE_NAME,

 result['Database']['KmsKeyId']))
        except self.client.exceptions.ResourceNotFoundException:
            print("Database doesn't exist")
        except Exception as err:
            print("Update database failed:", err)
```

Node.js

Cuplikan berikut digunakan AWS SDK untuk JavaScript v3. Untuk informasi selengkapnya tentang cara menginstal klien dan penggunaan, lihat Timestream Write Client - AWS SDK untuk JavaScript v3.

Lihat juga kelas UpdateDatabaseCommand dan UpdateDatabase

```
import { TimestreamWriteClient, UpdateDatabaseCommand } from "@aws-sdk/client-
timestream-write";
const writeClient = new TimestreamWriteClient({ region: "us-east-1" });
let updatedKmsKeyId = "<updatedKmsKeyId>";

const params = {
    DatabaseName: "testDbFromNode",
    KmsKeyId: updatedKmsKeyId
};
```

```
const command = new UpdateDatabaseCommand(params);

try {
    const data = await writeClient.send(command);
    console.log(`Database ${data.Database.DatabaseName} updated kmsKeyId to
 ${updatedKmsKeyId}`);
} catch (error) {
    if (error.code === 'ResourceNotFoundException') {
        console.log("Database doesn't exist.");
    } else {
        console.log("Update database failed.", error);
    }
}
```

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada contoh aplikasi di Node.js sampel Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
async function updateDatabase(updatedKmsKeyId) {

    if (updatedKmsKeyId === undefined) {
        console.log("Skipping UpdateDatabase; KmsKeyId was not given");
        return;
    }
    console.log("Updating Database");
    const params = {
        DatabaseName: constants.DATABASE_NAME,
        KmsKeyId: updatedKmsKeyId
    }

    const promise = writeClient.updateDatabase(params).promise();

    await promise.then(
        (data) => {
            console.log(`Database ${data.Database.DatabaseName} updated kmsKeyId to
 ${updatedKmsKeyId}`);
        },
        (err) => {
            if (err.code === 'ResourceNotFoundException') {
                console.log("Database doesn't exist.");
            } else {
                console.log("Update database failed.", err);
            }
        }
```

```
    );
}
```

.NET

```
        public async Task UpdateDatabase(String updatedKmsKeyId)
        {
            Console.WriteLine("Updating Database");

            try
            {
                var updateDatabaseRequest = new UpdateDatabaseRequest
                {
                    DatabaseName = Constants.DATABASE_NAME,
                    KmsKeyId = updatedKmsKeyId
                };
                UpdateDatabaseResponse response = await
 writeClient.UpdateDatabaseAsync(updateDatabaseRequest);
                Console.WriteLine($"Database {Constants.DATABASE_NAME} updated with
 KmsKeyId {updatedKmsKeyId}");
            }
            catch (ResourceNotFoundException)
            {
                Console.WriteLine("Database does not exist.");
            }
            catch (Exception e)
            {
                Console.WriteLine("Update database failed: " + e.ToString());
            }

        }

        private void PrintDatabases(List<Database> databases)
        {
            foreach (Database database in databases)
                Console.WriteLine($"Database:{database.DatabaseName}");
        }
```

Hapus basis data

Anda dapat menggunakan cuplikan kode berikut untuk menghapus database.

> **Note**
>
> Cuplikan kode ini didasarkan pada contoh aplikasi lengkap. GitHub Untuk informasi selengkapnya tentang cara memulai dengan contoh aplikasi, lihatAplikasi sampel.

Java

```
   public void deleteDatabase() {
       System.out.println("Deleting database");
       final DeleteDatabaseRequest deleteDatabaseRequest = new
DeleteDatabaseRequest();
       deleteDatabaseRequest.setDatabaseName(DATABASE_NAME);
       try {
           DeleteDatabaseResult result =
                   amazonTimestreamWrite.deleteDatabase(deleteDatabaseRequest);
           System.out.println("Delete database status: " +
result.getSdkHttpMetadata().getHttpStatusCode());
       } catch (final ResourceNotFoundException e) {
           System.out.println("Database " + DATABASE_NAME + " doesn't exist = " +
e);
           throw e;
       } catch (final Exception e) {
           System.out.println("Could not delete Database " + DATABASE_NAME + " = "
+ e);
           throw e;
       }
   }
```

Java v2

```
   public void deleteDatabase() {
       System.out.println("Deleting database");
       final DeleteDatabaseRequest deleteDatabaseRequest = new
DeleteDatabaseRequest();
       deleteDatabaseRequest.setDatabaseName(DATABASE_NAME);
       try {
           DeleteDatabaseResult result =
                   amazonTimestreamWrite.deleteDatabase(deleteDatabaseRequest);
           System.out.println("Delete database status: " +
result.getSdkHttpMetadata().getHttpStatusCode());
       } catch (final ResourceNotFoundException e) {
```

```
            System.out.println("Database " + DATABASE_NAME + " doesn't exist = " +
 e);
            throw e;
        } catch (final Exception e) {
            System.out.println("Could not delete Database " + DATABASE_NAME + " = "
 + e);
            throw e;
        }
    }
```

Go

```
deleteDatabaseInput := &timestreamwrite.DeleteDatabaseInput{
        DatabaseName:   aws.String(*databaseName),
    }

    _, err = writeSvc.DeleteDatabase(deleteDatabaseInput)

    if err != nil {
        fmt.Println("Error:")
        fmt.Println(err)
    } else {
        fmt.Println("Database deleted:", *databaseName)
    }
```

Python

```
    def delete_database(self):
        print("Deleting Database")
        try:
            result =
 self.client.delete_database(DatabaseName=Constant.DATABASE_NAME)
            print("Delete database status [%s]" % result['ResponseMetadata']
['HTTPStatusCode'])
        except self.client.exceptions.ResourceNotFoundException:
            print("database [%s] doesn't exist" % Constant.DATABASE_NAME)
        except Exception as err:
            print("Delete database failed:", err)
```

Node.js

Cuplikan berikut digunakan AWS SDK untuk JavaScript v3. Untuk informasi selengkapnya tentang cara menginstal klien dan penggunaan, lihat Timestream Write Client - AWS SDK untuk JavaScript v3.

Lihat juga kelas DeleteDatabaseCommand dan DeleteDatabase

```
import { TimestreamWriteClient, DeleteDatabaseCommand } from "@aws-sdk/client-
timestream-write";
const writeClient = new TimestreamWriteClient({ region: "us-east-1" });

const params = {
    DatabaseName: "testDbFromNode"
};

const command = new DeleteDatabaseCommand(params);

try {
    const data = await writeClient.send(command);
    console.log("Deleted database");
} catch (error) {
    if (error.code === 'ResourceNotFoundException') {
        console.log(`Database ${params.DatabaseName} doesn't exists.`);
    } else {
        console.log("Delete database failed.", error);
        throw error;
    }
}
```

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada contoh aplikasi di Node.js sampel Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
async function deleteDatabase() {
    console.log("Deleting Database");
    const params = {
        DatabaseName: constants.DATABASE_NAME
    };

    const promise = writeClient.deleteDatabase(params).promise();

    await promise.then(
```

```
        function (data) {
            console.log("Deleted database");
         },
        function(err) {
            if (err.code === 'ResourceNotFoundException') {
                console.log(`Database ${params.DatabaseName} doesn't exists.`);
            } else {
                console.log("Delete database failed.", err);
                throw err;
            }
        }
    );
}
```

.NET

```
        public async Task DeleteDatabase()
        {
            Console.WriteLine("Deleting database");
            try
            {
                var deleteDatabaseRequest = new DeleteDatabaseRequest
                {
                    DatabaseName = Constants.DATABASE_NAME
                };
                DeleteDatabaseResponse response = await
 writeClient.DeleteDatabaseAsync(deleteDatabaseRequest);
                Console.WriteLine($"Database {Constants.DATABASE_NAME} delete
 request status:{response.HttpStatusCode}");
            }
            catch (ResourceNotFoundException)
            {
                Console.WriteLine($"Database {Constants.DATABASE_NAME} does not
 exists");
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception while deleting database:" +
 e.ToString());
            }
        }
```

Daftar database

Anda dapat menggunakan cuplikan kode berikut untuk membuat daftar database Anda.

> **Note**
>
> Cuplikan kode ini didasarkan pada contoh aplikasi lengkap. GitHub Untuk informasi selengkapnya tentang cara memulai dengan contoh aplikasi, lihatAplikasi sampel.

Java

```
    public void listDatabases() {
        System.out.println("Listing databases");
        ListDatabasesRequest request = new ListDatabasesRequest();
        ListDatabasesResult result = amazonTimestreamWrite.listDatabases(request);
        final List<Database> databases = result.getDatabases();
        printDatabases(databases);

        String nextToken = result.getNextToken();
        while (nextToken != null && !nextToken.isEmpty()) {
            request.setNextToken(nextToken);
            ListDatabasesResult nextResult =
amazonTimestreamWrite.listDatabases(request);
            final List<Database> nextDatabases = nextResult.getDatabases();
            printDatabases(nextDatabases);
            nextToken = nextResult.getNextToken();
        }
    }

    private void printDatabases(List<Database> databases) {
        for (Database db : databases) {
            System.out.println(db.getDatabaseName());
        }
    }
```

Java v2

```
    public void listDatabases() {
        System.out.println("Listing databases");
        ListDatabasesRequest request =
ListDatabasesRequest.builder().maxResults(2).build();
```

```
        ListDatabasesIterable listDatabasesIterable =
 timestreamWriteClient.listDatabasesPaginator(request);
        for(ListDatabasesResponse listDatabasesResponse : listDatabasesIterable) {
            final List<Database> databases = listDatabasesResponse.databases();
            databases.forEach(database ->
 System.out.println(database.databaseName()));
        }
    }
```

Go

```
// List databases.
    listDatabasesMaxResult := int64(15)

    listDatabasesInput := &timestreamwrite.ListDatabasesInput{
        MaxResults: &listDatabasesMaxResult,
    }

    listDatabasesOutput, err := writeSvc.ListDatabases(listDatabasesInput)

    if err != nil {
        fmt.Println("Error:")
        fmt.Println(err)
    } else {
        fmt.Println("List databases is successful, below is the output:")
        fmt.Println(listDatabasesOutput)
    }
```

Python

```
    def list_databases(self):
        print("Listing databases")
        try:
            result = self.client.list_databases(MaxResults=5)
            self._print_databases(result['Databases'])
            next_token = result.get('NextToken', None)
            while next_token:
                result = self.client.list_databases(NextToken=next_token,
 MaxResults=5)
                self._print_databases(result['Databases'])
                next_token = result.get('NextToken', None)
        except Exception as err:
            print("List databases failed:", err)
```

Node.js

Cuplikan berikut digunakan AWS SDK untuk JavaScript v3. Untuk informasi selengkapnya tentang cara menginstal klien dan penggunaan, lihat Timestream Write Client - AWS SDK untuk JavaScript v3.

Lihat juga kelas ListDatabasesCommand dan ListDatabases

```
import { TimestreamWriteClient, ListDatabasesCommand } from "@aws-sdk/client-
timestream-write";
const writeClient = new TimestreamWriteClient({ region: "us-east-1" });

const params = {
    MaxResults: 15
};

const command = new ListDatabasesCommand(params);

getDatabasesList(null);

async function getDatabasesList(nextToken) {
    if (nextToken) {
        params.NextToken = nextToken;
    }

    try {
        const data = await writeClient.send(command);

        data.Databases.forEach(function (database) {
            console.log(database.DatabaseName);
        });

        if (data.NextToken) {
            return getDatabasesList(data.NextToken);
        }
    } catch (error) {
        console.log("Error while listing databases", error);
    }
}
```

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada contoh aplikasi di Node.js sampel Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
async function listDatabases() {
    console.log("Listing databases:");
    const databases = await getDatabasesList(null);
    databases.forEach(function(database){
        console.log(database.DatabaseName);
    });
}

function getDatabasesList(nextToken, databases = []) {
    var params = {
        MaxResults: 15
    };

    if(nextToken) {
        params.NextToken = nextToken;
    }

    return writeClient.listDatabases(params).promise()
        .then(
            (data) => {
                databases.push.apply(databases, data.Databases);
                if (data.NextToken) {
                    return getDatabasesList(data.NextToken, databases);
                } else {
                    return databases;
                }
            },
            (err) => {
                console.log("Error while listing databases", err);
            });
}
```

.NET

```
        public async Task ListDatabases()
        {
            Console.WriteLine("Listing Databases");

            try
            {
                var listDatabasesRequest = new ListDatabasesRequest
                {
                    MaxResults = 5
```

```
                };
                ListDatabasesResponse response = await
writeClient.ListDatabasesAsync(listDatabasesRequest);
                PrintDatabases(response.Databases);
                var nextToken = response.NextToken;
                while (nextToken != null)
                {
                    listDatabasesRequest.NextToken = nextToken;
                    response = await
writeClient.ListDatabasesAsync(listDatabasesRequest);
                    PrintDatabases(response.Databases);
                    nextToken = response.NextToken;
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("List database failed:" + e.ToString());
            }

        }
```

Membuat tabel

Topik

- Toko memori menulis
- Toko magnetik menulis

Toko memori menulis

Anda dapat menggunakan cuplikan kode berikut untuk membuat tabel yang memiliki penulisan penyimpanan magnetik dinonaktifkan, sebagai hasilnya Anda hanya dapat menulis data ke jendela penyimpanan penyimpanan memori Anda.

> **Note**
>
> Cuplikan kode ini didasarkan pada contoh aplikasi lengkap. GitHub Untuk informasi selengkapnya tentang cara memulai dengan contoh aplikasi, lihatAplikasi sampel.

Java

```
   public void createTable() {
       System.out.println("Creating table");
       CreateTableRequest createTableRequest = new CreateTableRequest();
       createTableRequest.setDatabaseName(DATABASE_NAME);
       createTableRequest.setTableName(TABLE_NAME);
       final RetentionProperties retentionProperties = new RetentionProperties()
               .withMemoryStoreRetentionPeriodInHours(HT_TTL_HOURS)
               .withMagneticStoreRetentionPeriodInDays(CT_TTL_DAYS);
       createTableRequest.setRetentionProperties(retentionProperties);

       try {
           amazonTimestreamWrite.createTable(createTableRequest);
           System.out.println("Table [" + TABLE_NAME + "] successfully created.");
       } catch (ConflictException e) {
           System.out.println("Table [" + TABLE_NAME + "] exists on database [" +
DATABASE_NAME + "] . Skipping database creation");
       }
   }
```

Java v2

```
   public void createTable() {
       System.out.println("Creating table");

       final RetentionProperties retentionProperties =
RetentionProperties.builder()
               .memoryStoreRetentionPeriodInHours(HT_TTL_HOURS)
               .magneticStoreRetentionPeriodInDays(CT_TTL_DAYS).build();
       final CreateTableRequest createTableRequest = CreateTableRequest.builder()

.databaseName(DATABASE_NAME).tableName(TABLE_NAME).retentionProperties(retentionProperties)

       try {
           timestreamWriteClient.createTable(createTableRequest);
           System.out.println("Table [" + TABLE_NAME + "] successfully created.");
       } catch (ConflictException e) {
           System.out.println("Table [" + TABLE_NAME + "] exists on database [" +
DATABASE_NAME + "] . Skipping database creation");
       }
   }
```

Go

```
// Create table.
    createTableInput := &timestreamwrite.CreateTableInput{
        DatabaseName: aws.String(*databaseName),
        TableName:    aws.String(*tableName),
    }
    _, err = writeSvc.CreateTable(createTableInput)

    if err != nil {
        fmt.Println("Error:")
        fmt.Println(err)
    } else {
        fmt.Println("Create table is successful")
    }
```

Python

```
    def create_table(self):
        print("Creating table")
        retention_properties = {
            'MemoryStoreRetentionPeriodInHours': Constant.HT_TTL_HOURS,
            'MagneticStoreRetentionPeriodInDays': Constant.CT_TTL_DAYS
        }
        try:
            self.client.create_table(DatabaseName=Constant.DATABASE_NAME,
 TableName=Constant.TABLE_NAME,
                                     RetentionProperties=retention_properties)
            print("Table [%s] successfully created." % Constant.TABLE_NAME)
        except self.client.exceptions.ConflictException:
            print("Table [%s] exists on database [%s]. Skipping table creation" % (
                Constant.TABLE_NAME, Constant.DATABASE_NAME))
        except Exception as err:
            print("Create table failed:", err)
```

Node.js

Cuplikan berikut digunakan AWS SDK untuk JavaScript v3. Untuk informasi selengkapnya tentang cara menginstal klien dan penggunaan, lihat Timestream Write Client - AWS SDK untuk JavaScript v3.

Lihat juga kelas CreateTableCommand dan CreateTable.

```
import { TimestreamWriteClient, CreateTableCommand } from "@aws-sdk/client-
timestream-write";
const writeClient = new TimestreamWriteClient({ region: "us-east-1" });

const params = {
    DatabaseName: "testDbFromNode",
    TableName: "testTableFromNode",
    RetentionProperties: {
        MemoryStoreRetentionPeriodInHours: 24,
        MagneticStoreRetentionPeriodInDays: 365
    }
};

const command = new CreateTableCommand(params);

try {
    const data = await writeClient.send(command);
    console.log(`Table ${data.Table.TableName} created successfully`);
} catch (error) {
    if (error.code === 'ConflictException') {
        console.log(`Table ${params.TableName} already exists on db
 ${params.DatabaseName}. Skipping creation.`);
    } else {
        console.log("Error creating table. ", error);
        throw error;
    }
}
```

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada aplikasi sampel Node.js Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
async function createTable() {
    console.log("Creating Table");
    const params = {
        DatabaseName: constants.DATABASE_NAME,
        TableName: constants.TABLE_NAME,
        RetentionProperties: {
            MemoryStoreRetentionPeriodInHours: constants.HT_TTL_HOURS,
            MagneticStoreRetentionPeriodInDays: constants.CT_TTL_DAYS
        }
    };

    const promise = writeClient.createTable(params).promise();
```

```
    await promise.then(
        (data) => {
            console.log(`Table ${data.Table.TableName} created successfully`);
        },
        (err) => {
            if (err.code === 'ConflictException') {
                console.log(`Table ${params.TableName} already exists on db
 ${params.DatabaseName}. Skipping creation.`);
            } else {
                console.log("Error creating table. ", err);
                throw err;
            }
        }
    );
}
```

.NET

```
        public async Task CreateTable()
        {
            Console.WriteLine("Creating Table");

            try
            {
                var createTableRequest = new CreateTableRequest
                {
                    DatabaseName = Constants.DATABASE_NAME,
                    TableName = Constants.TABLE_NAME,
                    RetentionProperties = new RetentionProperties
                    {
                        MagneticStoreRetentionPeriodInDays = Constants.CT_TTL_DAYS,
                        MemoryStoreRetentionPeriodInHours = Constants.HT_TTL_HOURS
                    }
                };
                CreateTableResponse response = await
 writeClient.CreateTableAsync(createTableRequest);
                Console.WriteLine($"Table {Constants.TABLE_NAME} created");
            }
            catch (ConflictException)
            {
                Console.WriteLine("Table already exists.");
            }
```

```
            catch (Exception e)
            {
                Console.WriteLine("Create table failed:" + e.ToString());
            }

        }
```

Toko magnetik menulis

Anda dapat menggunakan cuplikan kode berikut untuk membuat tabel dengan penulisan penyimpanan magnetik diaktifkan. Dengan penulisan penyimpanan magnetik, Anda dapat menulis data ke jendela penyimpanan penyimpanan memori dan jendela retensi penyimpanan magnetik.

> **Note**
>
> Cuplikan kode ini didasarkan pada contoh aplikasi lengkap. GitHub Untuk informasi selengkapnya tentang cara memulai dengan contoh aplikasi, lihatAplikasi sampel.

Java

```
   public void createTable(String databaseName, String tableName) {
       System.out.println("Creating table");
       CreateTableRequest createTableRequest = new CreateTableRequest();
       createTableRequest.setDatabaseName(databaseName);
       createTableRequest.setTableName(tableName);
       final RetentionProperties retentionProperties = new RetentionProperties()
               .withMemoryStoreRetentionPeriodInHours(HT_TTL_HOURS)
               .withMagneticStoreRetentionPeriodInDays(CT_TTL_DAYS);
       createTableRequest.setRetentionProperties(retentionProperties);
       // Enable MagneticStoreWrite
       final MagneticStoreWriteProperties magneticStoreWriteProperties = new
MagneticStoreWriteProperties()
               .withEnableMagneticStoreWrites(true);

createTableRequest.setMagneticStoreWriteProperties(magneticStoreWriteProperties);
       try {
           amazonTimestreamWrite.createTable(createTableRequest);
           System.out.println("Table [" + tableName + "] successfully created.");
       } catch (ConflictException e) {
```

```
            System.out.println("Table [" + tableName + "] exists on database [" +
 databaseName + "] . Skipping table creation");
            //We do not throw exception here, we use the existing table instead
        }
    }
```

Java v2

```
    public void createTable(String databaseName, String tableName) {
        System.out.println("Creating table");

        // Enable MagneticStoreWrite
        final MagneticStoreWriteProperties magneticStoreWriteProperties =
                MagneticStoreWriteProperties.builder()
                        .enableMagneticStoreWrites(true)
                        .build();

        CreateTableRequest createTableRequest =
                CreateTableRequest.builder()
                        .databaseName(databaseName)
                        .tableName(tableName)
                        .retentionProperties(RetentionProperties.builder()
                                .memoryStoreRetentionPeriodInHours(HT_TTL_HOURS)
                                .magneticStoreRetentionPeriodInDays(CT_TTL_DAYS)
                                .build())
                        .magneticStoreWriteProperties(magneticStoreWriteProperties)
                        .build();
        try {
            timestreamWriteClient.createTable(createTableRequest);
            System.out.println("Table [" + tableName + "] successfully created.");
        } catch (ConflictException e) {
            System.out.println("Table [" + tableName + "] exists in database [" +
 databaseName + "] . Skipping table creation");
        }
    }
```

Go

```
// Create table.
    createTableInput := &timestreamwrite.CreateTableInput{
        DatabaseName: aws.String(*databaseName),
        TableName:    aws.String(*tableName),
    // Enable MagneticStoreWrite
```

```
        MagneticStoreWriteProperties: &timestreamwrite.MagneticStoreWriteProperties{
            EnableMagneticStoreWrites: aws.Bool(true),
             },
      }
    _, err = writeSvc.CreateTable(createTableInput)
```

Python

```
    def create_table(self):
        print("Creating table")
        retention_properties = {
            'MemoryStoreRetentionPeriodInHours': Constant.HT_TTL_HOURS,
            'MagneticStoreRetentionPeriodInDays': Constant.CT_TTL_DAYS
        }
        magnetic_store_write_properties = {
            'EnableMagneticStoreWrites': True
        }
        try:
            self.client.create_table(DatabaseName=Constant.DATABASE_NAME,
 TableName=Constant.TABLE_NAME,
                                     RetentionProperties=retention_properties,

 MagneticStoreWriteProperties=magnetic_store_write_properties)
            print("Table [%s] successfully created." % Constant.TABLE_NAME)
        except self.client.exceptions.ConflictException:
            print("Table [%s] exists on database [%s]. Skipping table creation" % (
                Constant.TABLE_NAME, Constant.DATABASE_NAME))
        except Exception as err:
            print("Create table failed:", err)
```

Node.js

```
async function createTable() {
    console.log("Creating Table");

    const params = {
        DatabaseName: constants.DATABASE_NAME,
        TableName: constants.TABLE_NAME,
        RetentionProperties: {
            MemoryStoreRetentionPeriodInHours: constants.HT_TTL_HOURS,
            MagneticStoreRetentionPeriodInDays: constants.CT_TTL_DAYS
        },
        MagneticStoreWriteProperties: {
```

```
            EnableMagneticStoreWrites: true
        }
    };

    const promise = writeClient.createTable(params).promise();

    await promise.then(
        (data) => {
            console.log(`Table ${data.Table.TableName} created successfully`);
        },
        (err) => {
            if (err.code === 'ConflictException') {
                console.log(`Table ${params.TableName} already exists on db
 ${params.DatabaseName}. Skipping creation.`);
            } else {
                console.log("Error creating table. ", err);
                throw err;
            }
        }
    );
}
```

.NET

```
        public async Task CreateTable()
        {
            Console.WriteLine("Creating Table");

            try
            {
                var createTableRequest = new CreateTableRequest
                {
                    DatabaseName = Constants.DATABASE_NAME,
                    TableName = Constants.TABLE_NAME,
                    RetentionProperties = new RetentionProperties
                    {
                        MagneticStoreRetentionPeriodInDays = Constants.CT_TTL_DAYS,
                        MemoryStoreRetentionPeriodInHours = Constants.HT_TTL_HOURS
                    },
                    // Enable MagneticStoreWrite
                    MagneticStoreWriteProperties = new MagneticStoreWriteProperties
                    {
                        EnableMagneticStoreWrites = true,
```

```
                }
            };
            CreateTableResponse response = await
writeClient.CreateTableAsync(createTableRequest);
            Console.WriteLine($"Table {Constants.TABLE_NAME} created");
        }
        catch (ConflictException)
        {
            Console.WriteLine("Table already exists.");
        }
        catch (Exception e)
        {
            Console.WriteLine("Create table failed:" + e.ToString());
        }

    }
```

Jelaskan tabel

Anda dapat menggunakan cuplikan kode berikut untuk mendapatkan informasi tentang atribut tabel Anda.

> **Note**
>
> Cuplikan kode ini didasarkan pada contoh aplikasi lengkap. GitHub Untuk informasi selengkapnya tentang cara memulai dengan contoh aplikasi, lihatAplikasi sampel.

Java

```
   public void describeTable() {
       System.out.println("Describing table");
       final DescribeTableRequest describeTableRequest = new
DescribeTableRequest();
       describeTableRequest.setDatabaseName(DATABASE_NAME);
       describeTableRequest.setTableName(TABLE_NAME);
       try {
           DescribeTableResult result =
amazonTimestreamWrite.describeTable(describeTableRequest);
           String tableId = result.getTable().getArn();
           System.out.println("Table " + TABLE_NAME + " has id " + tableId);
```

```
        } catch (final Exception e) {
            System.out.println("Table " + TABLE_NAME + " doesn't exist = " + e);
            throw e;
        }
    }
```

Java v2

```
    public void describeTable() {
        System.out.println("Describing table");
        final DescribeTableRequest describeTableRequest =
 DescribeTableRequest.builder()
                .databaseName(DATABASE_NAME).tableName(TABLE_NAME).build();
        try {
            DescribeTableResponse response =
 timestreamWriteClient.describeTable(describeTableRequest);
            String tableId = response.table().arn();
            System.out.println("Table " + TABLE_NAME + " has id " + tableId);
        } catch (final Exception e) {
            System.out.println("Table " + TABLE_NAME + " doesn't exist = " + e);
            throw e;
        }
    }
```

Go

```
// Describe table.
    describeTableInput := &timestreamwrite.DescribeTableInput{
        DatabaseName: aws.String(*databaseName),
        TableName:    aws.String(*tableName),
    }
    describeTableOutput, err := writeSvc.DescribeTable(describeTableInput)

    if err != nil {
        fmt.Println("Error:")
        fmt.Println(err)
    } else {
        fmt.Println("Describe table is successful, below is the output:")
        fmt.Println(describeTableOutput)
    }
```

Python

```
    def describe_table(self):
        print("Describing table")
        try:
            result = self.client.describe_table(DatabaseName=Constant.DATABASE_NAME,
 TableName=Constant.TABLE_NAME)
            print("Table [%s] has id [%s]" % (Constant.TABLE_NAME, result['Table']
['Arn']))
        except self.client.exceptions.ResourceNotFoundException:
            print("Table doesn't exist")
        except Exception as err:
            print("Describe table failed:", err)
```

Node.js

Cuplikan berikut digunakan AWS SDK untuk JavaScript v3. Untuk informasi selengkapnya tentang cara menginstal klien dan penggunaan, lihat Timestream Write Client - AWS SDK untuk JavaScript v3.

Lihat juga kelas DescribeTableCommand dan DescribeTable

```
import { TimestreamWriteClient, DescribeTableCommand } from "@aws-sdk/client-
timestream-write";
const writeClient = new TimestreamWriteClient({ region: "us-east-1" });

const params = {
    DatabaseName: "testDbFromNode",
    TableName: "testTableFromNode"
};

const command = new DescribeTableCommand(params);

try {
    const data = await writeClient.send(command);
    console.log(`Table ${data.Table.TableName} has id ${data.Table.Arn}`);
} catch (error) {
    if (error.code === 'ResourceNotFoundException') {
        console.log("Table or Database doesn't exist.");
    } else {
        console.log("Describe table failed.", error);
        throw error;
    }
```

```
}
```

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada contoh aplikasi di Node.js sampel Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
async function describeTable() {
    console.log("Describing Table");
    const params = {
        DatabaseName: constants.DATABASE_NAME,
        TableName: constants.TABLE_NAME
    };

    const promise = writeClient.describeTable(params).promise();

    await promise.then(
        (data) => {
            console.log(`Table ${data.Table.TableName} has id ${data.Table.Arn}`);
        },
        (err) => {
            if (err.code === 'ResourceNotFoundException') {
                console.log("Table or Database doesn't exists.");
            } else {
                console.log("Describe table failed.", err);
                throw err;
            }
        }
    );
}
```

.NET

```
        public async Task DescribeTable()
        {
            Console.WriteLine("Describing Table");

            try
            {
                var describeTableRequest = new DescribeTableRequest
                {
                    DatabaseName = Constants.DATABASE_NAME,
                    TableName = Constants.TABLE_NAME
                };
```

```
                DescribeTableResponse response = await
 writeClient.DescribeTableAsync(describeTableRequest);
                Console.WriteLine($"Table {Constants.TABLE_NAME} has id:
{response.Table.Arn}");
            }
            catch (ResourceNotFoundException)
            {
                Console.WriteLine("Table does not exist.");
            }
            catch (Exception e)
            {
                Console.WriteLine("Describe table failed:" + e.ToString());
            }

        }
```

Perbarui Tabel

Anda dapat menggunakan cuplikan kode berikut untuk memperbarui tabel.

> **Note**
>
> Cuplikan kode ini didasarkan pada contoh aplikasi lengkap. GitHub Untuk informasi selengkapnya tentang cara memulai dengan contoh aplikasi, lihatAplikasi sampel.

Java

```
    public void updateTable() {
        System.out.println("Updating table");
        UpdateTableRequest updateTableRequest = new UpdateTableRequest();
        updateTableRequest.setDatabaseName(DATABASE_NAME);
        updateTableRequest.setTableName(TABLE_NAME);

        final RetentionProperties retentionProperties = new RetentionProperties()
                .withMemoryStoreRetentionPeriodInHours(HT_TTL_HOURS)
                .withMagneticStoreRetentionPeriodInDays(CT_TTL_DAYS);

        updateTableRequest.setRetentionProperties(retentionProperties);

        amazonTimestreamWrite.updateTable(updateTableRequest);
```

```
        System.out.println("Table updated");
    }
```

Java v2

```
    public void updateTable() {
        System.out.println("Updating table");

        final RetentionProperties retentionProperties =
 RetentionProperties.builder()
                .memoryStoreRetentionPeriodInHours(HT_TTL_HOURS)
                .magneticStoreRetentionPeriodInDays(CT_TTL_DAYS).build();
        final UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()

 .databaseName(DATABASE_NAME).tableName(TABLE_NAME).retentionProperties(retentionProperties)

        timestreamWriteClient.updateTable(updateTableRequest);
        System.out.println("Table updated");
    }
```

Go

```
// Update table.
    magneticStoreRetentionPeriodInDays := int64(7 * 365)
    memoryStoreRetentionPeriodInHours := int64(24)

    updateTableInput := &timestreamwrite.UpdateTableInput{
        DatabaseName: aws.String(*databaseName),
        TableName:    aws.String(*tableName),
        RetentionProperties: &timestreamwrite.RetentionProperties{
            MagneticStoreRetentionPeriodInDays: &magneticStoreRetentionPeriodInDays,
            MemoryStoreRetentionPeriodInHours:  &memoryStoreRetentionPeriodInHours,
        },
    }
    updateTableOutput, err := writeSvc.UpdateTable(updateTableInput)

    if err != nil {
        fmt.Println("Error:")
        fmt.Println(err)
    } else {
        fmt.Println("Update table is successful, below is the output:")
        fmt.Println(updateTableOutput)
    }
```

Python

```
    def update_table(self):
        print("Updating table")
        retention_properties = {
            'MemoryStoreRetentionPeriodInHours': Constant.HT_TTL_HOURS,
            'MagneticStoreRetentionPeriodInDays': Constant.CT_TTL_DAYS
        }
        try:
            self.client.update_table(DatabaseName=Constant.DATABASE_NAME,
 TableName=Constant.TABLE_NAME,
                                     RetentionProperties=retention_properties)
            print("Table updated.")
        except Exception as err:
            print("Update table failed:", err)
```

Node.js

Cuplikan berikut digunakan AWS SDK untuk JavaScript v3. Untuk informasi selengkapnya tentang cara menginstal klien dan penggunaan, lihat Timestream Write Client - AWS SDK untuk JavaScript v3.

Lihat juga kelas UpdateTableCommand dan UpdateTable.

```
import { TimestreamWriteClient, UpdateTableCommand } from "@aws-sdk/client-
timestream-write";
const writeClient = new TimestreamWriteClient({ region: "us-east-1" });

const params = {
    DatabaseName: "testDbFromNode",
    TableName: "testTableFromNode",
    RetentionProperties: {
        MemoryStoreRetentionPeriodInHours: 24,
        MagneticStoreRetentionPeriodInDays: 180
    }
};

const command = new UpdateTableCommand(params);

try {
    const data = await writeClient.send(command);
    console.log("Table updated")
} catch (error) {
```

```
    console.log("Error updating table. ", error);
}
```

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada aplikasi sampel Node.js Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
async function updateTable() {
    console.log("Updating Table");
    const params = {
        DatabaseName: constants.DATABASE_NAME,
        TableName: constants.TABLE_NAME,
        RetentionProperties: {
            MemoryStoreRetentionPeriodInHours: constants.HT_TTL_HOURS,
            MagneticStoreRetentionPeriodInDays: constants.CT_TTL_DAYS
        }
    };

    const promise = writeClient.updateTable(params).promise();

    await promise.then(
        (data) => {
            console.log("Table updated")
        },
        (err) => {
            console.log("Error updating table. ", err);
            throw err;
        }
    );
}
```

.NET

```
        public async Task UpdateTable()
        {
            Console.WriteLine("Updating Table");

            try
            {
                var updateTableRequest = new UpdateTableRequest
                {
                    DatabaseName = Constants.DATABASE_NAME,
                    TableName = Constants.TABLE_NAME,
                    RetentionProperties = new RetentionProperties
```

```
                    {
                        MagneticStoreRetentionPeriodInDays = Constants.CT_TTL_DAYS,
                        MemoryStoreRetentionPeriodInHours = Constants.HT_TTL_HOURS
                    }
                };
                UpdateTableResponse response = await
writeClient.UpdateTableAsync(updateTableRequest);
                Console.WriteLine($"Table {Constants.TABLE_NAME} updated");
            }
            catch (ResourceNotFoundException)
            {
                Console.WriteLine("Table does not exist.");
            }
            catch (Exception e)
            {
                Console.WriteLine("Update table failed:" + e.ToString());
            }

        }
```

Hapus tabel

Anda dapat menggunakan cuplikan kode berikut untuk menghapus tabel.

> **Note**
>
> Cuplikan kode ini didasarkan pada contoh aplikasi lengkap. GitHub Untuk informasi selengkapnya tentang cara memulai dengan contoh aplikasi, lihatAplikasi sampel.

Java

```
    public void deleteTable() {
        System.out.println("Deleting table");
        final DeleteTableRequest deleteTableRequest = new DeleteTableRequest();
        deleteTableRequest.setDatabaseName(DATABASE_NAME);
        deleteTableRequest.setTableName(TABLE_NAME);
        try {
            DeleteTableResult result =
                    amazonTimestreamWrite.deleteTable(deleteTableRequest);
```

```
            System.out.println("Delete table status: " +
 result.getSdkHttpMetadata().getHttpStatusCode());
        } catch (final ResourceNotFoundException e) {
            System.out.println("Table " + TABLE_NAME + " doesn't exist = " + e);
            throw e;
        } catch (final Exception e) {
            System.out.println("Could not delete table " + TABLE_NAME + " = " + e);
            throw e;
        }
    }
```

Java v2

```
    public void deleteTable() {
        System.out.println("Deleting table");
        final DeleteTableRequest deleteTableRequest = DeleteTableRequest.builder()
                .databaseName(DATABASE_NAME).tableName(TABLE_NAME).build();
        try {
            DeleteTableResponse response =
                    timestreamWriteClient.deleteTable(deleteTableRequest);
            System.out.println("Delete table status: " +
 response.sdkHttpResponse().statusCode());
        } catch (final ResourceNotFoundException e) {
            System.out.println("Table " + TABLE_NAME + " doesn't exist = " + e);
            throw e;
        } catch (final Exception e) {
            System.out.println("Could not delete table " + TABLE_NAME + " = " + e);
            throw e;
        }
    }
```

Go

```
deleteTableInput := &timestreamwrite.DeleteTableInput{
        DatabaseName:   aws.String(*databaseName),
        TableName:    aws.String(*tableName),
    }
    _, err = writeSvc.DeleteTable(deleteTableInput)

    if err != nil {
        fmt.Println("Error:")
        fmt.Println(err)
    } else {
```

```
        fmt.Println("Table deleted", *tableName)
    }
```

Python

```
    def delete_table(self):
        print("Deleting Table")
        try:
            result = self.client.delete_table(DatabaseName=Constant.DATABASE_NAME,
 TableName=Constant.TABLE_NAME)
            print("Delete table status [%s]" % result['ResponseMetadata']
['HTTPStatusCode'])
        except self.client.exceptions.ResourceNotFoundException:
            print("Table [%s] doesn't exist" % Constant.TABLE_NAME)
        except Exception as err:
            print("Delete table failed:", err)
```

Node.js

Cuplikan berikut digunakan AWS SDK untuk JavaScript v3. Untuk informasi selengkapnya tentang cara menginstal klien dan penggunaan, lihat Timestream Write Client - AWS SDK untuk JavaScript v3.

Lihat juga kelas DeleteTableCommand dan DeleteTable

```
import { TimestreamWriteClient, DeleteTableCommand } from "@aws-sdk/client-
timestream-write";
const writeClient = new TimestreamWriteClient({ region: "us-east-1" });

const params = {
    DatabaseName: "testDbFromNode",
    TableName: "testTableFromNode"
};

const command = new DeleteTableCommand(params);

try {
    const data = await writeClient.send(command);
    console.log("Deleted table");
} catch (error) {
    if (error.code === 'ResourceNotFoundException') {
        console.log(`Table ${params.TableName} or Database ${params.DatabaseName}
 doesn't exist.`);
```

```
    } else {
        console.log("Delete table failed.", error);
        throw error;
    }
}
```

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada contoh aplikasi di Node.js sampel Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
async function deleteTable() {
    console.log("Deleting Table");
    const params = {
        DatabaseName: constants.DATABASE_NAME,
        TableName: constants.TABLE_NAME
    };

    const promise = writeClient.deleteTable(params).promise();

    await promise.then(
        function (data) {
            console.log("Deleted table");
        },
        function(err) {
            if (err.code === 'ResourceNotFoundException') {
                console.log(`Table ${params.TableName} or Database
 ${params.DatabaseName} doesn't exists.`);
            } else {
                console.log("Delete table failed.", err);
                throw err;
            }
        }
    );
}
```

.NET

```
        public async Task DeleteTable()
        {
            Console.WriteLine("Deleting table");
            try
            {
                var deleteTableRequest = new DeleteTableRequest
                {
```

```
                    DatabaseName = Constants.DATABASE_NAME,
                    TableName = Constants.TABLE_NAME
                };
                DeleteTableResponse response = await
writeClient.DeleteTableAsync(deleteTableRequest);
                Console.WriteLine($"Table {Constants.TABLE_NAME} delete request
status: {response.HttpStatusCode}");
            }
            catch (ResourceNotFoundException)
            {
                Console.WriteLine($"Table {Constants.TABLE_NAME} does not exists");
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception while deleting table:" + e.ToString());
            }
        }
```

Mencantumkan tabel

Anda dapat menggunakan cuplikan kode berikut untuk daftar tabel.

> **Note**
>
> Cuplikan kode ini didasarkan pada contoh aplikasi lengkap. GitHub Untuk informasi selengkapnya tentang cara memulai dengan contoh aplikasi, lihatAplikasi sampel.

Java

```
    public void listTables() {
        System.out.println("Listing tables");
        ListTablesRequest request = new ListTablesRequest();
        request.setDatabaseName(DATABASE_NAME);
        ListTablesResult result = amazonTimestreamWrite.listTables(request);
        printTables(result.getTables());

        String nextToken = result.getNextToken();
        while (nextToken != null && !nextToken.isEmpty()) {
            request.setNextToken(nextToken);
            ListTablesResult nextResult = amazonTimestreamWrite.listTables(request);
```

```
            printTables(nextResult.getTables());
            nextToken = nextResult.getNextToken();
        }
    }

     private void printTables(List<Table> tables) {
        for (Table table : tables) {
            System.out.println(table.getTableName());
        }
    }
```

Java v2

```
    public void listTables() {
        System.out.println("Listing tables");
        ListTablesRequest request =
 ListTablesRequest.builder().databaseName(DATABASE_NAME).maxResults(2).build();
        ListTablesIterable listTablesIterable =
 timestreamWriteClient.listTablesPaginator(request);
        for(ListTablesResponse listTablesResponse : listTablesIterable) {
            final List<Table> tables = listTablesResponse.tables();
            tables.forEach(table -> System.out.println(table.tableName()));
        }
    }
```

Go

```
listTablesMaxResult := int64(15)

    listTablesInput := &timestreamwrite.ListTablesInput{
        DatabaseName: aws.String(*databaseName),
        MaxResults:   &listTablesMaxResult,
    }
    listTablesOutput, err := writeSvc.ListTables(listTablesInput)

    if err != nil {
        fmt.Println("Error:")
        fmt.Println(err)
    } else {
        fmt.Println("List tables is successful, below is the output:")
        fmt.Println(listTablesOutput)
    }
```

Python

```
    def list_tables(self):
        print("Listing tables")
        try:
            result = self.client.list_tables(DatabaseName=Constant.DATABASE_NAME,
 MaxResults=5)
            self.__print_tables(result['Tables'])
            next_token = result.get('NextToken', None)
            while next_token:
                result =
 self.client.list_tables(DatabaseName=Constant.DATABASE_NAME,
                                                    NextToken=next_token, MaxResults=5)
                self.__print_tables(result['Tables'])
                next_token = result.get('NextToken', None)
        except Exception as err:
            print("List tables failed:", err)
```

Node.js

Cuplikan berikut digunakan AWS SDK untuk JavaScript v3. Untuk informasi selengkapnya tentang cara menginstal klien dan penggunaan, lihat Timestream Write Client - AWS SDK untuk JavaScript v3.

Lihat juga kelas ListTablesCommand dan ListTables.

```
import { TimestreamWriteClient, ListTablesCommand } from "@aws-sdk/client-
timestream-write";
const writeClient = new TimestreamWriteClient({ region: "us-east-1" });

const params = {
    DatabaseName: "testDbFromNode",
    MaxResults: 15
};

const command = new ListTablesCommand(params);

getTablesList(null);

async function getTablesList(nextToken) {
    if (nextToken) {
        params.NextToken = nextToken;
    }
```

```
    try {
        const data = await writeClient.send(command);

        data.Tables.forEach(function (table) {
            console.log(table.TableName);
        });

        if (data.NextToken) {
            return getTablesList(data.NextToken);
        }
    } catch (error) {
        console.log("Error while listing tables", error);
    }
}
```

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada aplikasi sampel Node.js Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
async function listTables() {
    console.log("Listing tables:");
    const tables = await getTablesList(null);
    tables.forEach(function(table){
        console.log(table.TableName);
    });
}

function getTablesList(nextToken, tables = []) {
    var params = {
        DatabaseName: constants.DATABASE_NAME,
        MaxResults: 15
    };

    if(nextToken) {
        params.NextToken = nextToken;
    }

    return writeClient.listTables(params).promise()
        .then(
            (data) => {
                tables.push.apply(tables, data.Tables);
                if (data.NextToken) {
                    return getTablesList(data.NextToken, tables);
```

```
                } else {
                    return tables;
                }
            },
            (err) => {
                console.log("Error while listing databases", err);
            });
}
```

.NET

```
        public async Task ListTables()
        {
            Console.WriteLine("Listing Tables");

            try
            {
                var listTablesRequest = new ListTablesRequest
                {
                    MaxResults = 5,
                    DatabaseName = Constants.DATABASE_NAME
                };
                ListTablesResponse response = await
 writeClient.ListTablesAsync(listTablesRequest);
                PrintTables(response.Tables);
                string nextToken = response.NextToken;
                while (nextToken != null)
                {
                    listTablesRequest.NextToken = nextToken;
                    response = await writeClient.ListTablesAsync(listTablesRequest);
                    PrintTables(response.Tables);
                    nextToken = response.NextToken;
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("List table failed:" + e.ToString());
            }

        }

        private void PrintTables(List<Table> tables)
        {
```

```
            foreach (Table table in tables)
                Console.WriteLine($"Table: {table.TableName}");
        }
```

Menulis data (sisipan dan upserts)

Topik

- Menulis batch catatan
- Menulis kumpulan catatan dengan atribut umum
- Menambah catatan
- Contoh atribut multi-ukuran
- Menangani kegagalan menulis

Menulis batch catatan

Anda dapat menggunakan cuplikan kode berikut untuk menulis data ke dalam tabel Amazon Timestream. Menulis data dalam batch membantu mengoptimalkan biaya penulisan. Untuk informasi selengkapnya, lihat Menghitung jumlah penulisan.

> **Note**
>
> Cuplikan kode ini didasarkan pada contoh aplikasi lengkap. GitHub Untuk informasi selengkapnya tentang cara memulai dengan contoh aplikasi, lihatAplikasi sampel.

Java

```
  public void writeRecords() {
    System.out.println("Writing records");
    // Specify repeated values for all records
    List<Record> records = new ArrayList<>();
    final long time = System.currentTimeMillis();

    List<Dimension> dimensions = new ArrayList<>();
    final Dimension region = new Dimension().withName("region").withValue("us-
east-1");
    final Dimension az = new Dimension().withName("az").withValue("az1");
```

```
   final Dimension hostname = new
Dimension().withName("hostname").withValue("host1");

   dimensions.add(region);
   dimensions.add(az);
   dimensions.add(hostname);

   Record cpuUtilization = new Record()
       .withDimensions(dimensions)
       .withMeasureName("cpu_utilization")
       .withMeasureValue("13.5")
       .withMeasureValueType(MeasureValueType.DOUBLE)
       .withTime(String.valueOf(time));
   Record memoryUtilization = new Record()
       .withDimensions(dimensions)
       .withMeasureName("memory_utilization")
       .withMeasureValue("40")
       .withMeasureValueType(MeasureValueType.DOUBLE)
       .withTime(String.valueOf(time));

   records.add(cpuUtilization);
   records.add(memoryUtilization);

   WriteRecordsRequest writeRecordsRequest = new WriteRecordsRequest()
       .withDatabaseName(DATABASE_NAME)
       .withTableName(TABLE_NAME)
       .withRecords(records);

   try {
     WriteRecordsResult writeRecordsResult =
amazonTimestreamWrite.writeRecords(writeRecordsRequest);
     System.out.println("WriteRecords Status: " +
writeRecordsResult.getSdkHttpMetadata().getHttpStatusCode());
   } catch (RejectedRecordsException e) {
     System.out.println("RejectedRecords: " + e);
     for (RejectedRecord rejectedRecord : e.getRejectedRecords()) {
       System.out.println("Rejected Index " + rejectedRecord.getRecordIndex() + ":
"
           + rejectedRecord.getReason());
     }
     System.out.println("Other records were written successfully. ");
   } catch (Exception e) {
     System.out.println("Error: " + e);
   }
```

```
  }
```

Java v2

```
  public void writeRecords() {
    System.out.println("Writing records");
    // Specify repeated values for all records
    List<Record> records = new ArrayList<>();
    final long time = System.currentTimeMillis();

    List<Dimension> dimensions = new ArrayList<>();
    final Dimension region = Dimension.builder().name("region").value("us-
east-1").build();
    final Dimension az = Dimension.builder().name("az").value("az1").build();
    final Dimension hostname =
 Dimension.builder().name("hostname").value("host1").build();

    dimensions.add(region);
    dimensions.add(az);
    dimensions.add(hostname);

    Record cpuUtilization = Record.builder()
        .dimensions(dimensions)
        .measureValueType(MeasureValueType.DOUBLE)
        .measureName("cpu_utilization")
        .measureValue("13.5")
        .time(String.valueOf(time)).build();

    Record memoryUtilization = Record.builder()
        .dimensions(dimensions)
        .measureValueType(MeasureValueType.DOUBLE)
        .measureName("memory_utilization")
        .measureValue("40")
        .time(String.valueOf(time)).build();

    records.add(cpuUtilization);
    records.add(memoryUtilization);

    WriteRecordsRequest writeRecordsRequest = WriteRecordsRequest.builder()
        .databaseName(DATABASE_NAME).tableName(TABLE_NAME).records(records).build();

    try {
```

```
      WriteRecordsResponse writeRecordsResponse =
 timestreamWriteClient.writeRecords(writeRecordsRequest);
      System.out.println("WriteRecords Status: " +
 writeRecordsResponse.sdkHttpResponse().statusCode());
    } catch (RejectedRecordsException e) {
      System.out.println("RejectedRecords: " + e);
      for (RejectedRecord rejectedRecord : e.rejectedRecords()) {
        System.out.println("Rejected Index " + rejectedRecord.recordIndex() + ": "
            + rejectedRecord.reason());
      }
      System.out.println("Other records were written successfully. ");
    } catch (Exception e) {
      System.out.println("Error: " + e);
    }
  }
```

Go

```
now := time.Now()
currentTimeInSeconds := now.Unix()
writeRecordsInput := &timestreamwrite.WriteRecordsInput{
  DatabaseName: aws.String(*databaseName),
  TableName:  aws.String(*tableName),
  Records: []*timestreamwrite.Record{
    &timestreamwrite.Record{
      Dimensions: []*timestreamwrite.Dimension{
        &timestreamwrite.Dimension{
          Name:  aws.String("region"),
          Value: aws.String("us-east-1"),
        },
        &timestreamwrite.Dimension{
          Name:  aws.String("az"),
          Value: aws.String("az1"),
        },
        &timestreamwrite.Dimension{
          Name:  aws.String("hostname"),
          Value: aws.String("host1"),
        },
      },
      MeasureName:    aws.String("cpu_utilization"),
      MeasureValue:   aws.String("13.5"),
      MeasureValueType: aws.String("DOUBLE"),
      Time:       aws.String(strconv.FormatInt(currentTimeInSeconds, 10)),
```

```
      TimeUnit:  aws.String("SECONDS"),
    },
    &timestreamwrite.Record{
      Dimensions: []*timestreamwrite.Dimension{
        &timestreamwrite.Dimension{
          Name:  aws.String("region"),
          Value: aws.String("us-east-1"),
        },
        &timestreamwrite.Dimension{
          Name:  aws.String("az"),
          Value: aws.String("az1"),
        },
        &timestreamwrite.Dimension{
          Name:  aws.String("hostname"),
          Value: aws.String("host1"),
        },
      },
      MeasureName:    aws.String("memory_utilization"),
      MeasureValue:   aws.String("40"),
      MeasureValueType: aws.String("DOUBLE"),
      Time:       aws.String(strconv.FormatInt(currentTimeInSeconds, 10)),
      TimeUnit:  aws.String("SECONDS"),
    },
  },
}

_, err = writeSvc.WriteRecords(writeRecordsInput)

if err != nil {
  fmt.Println("Error:")
  fmt.Println(err)
} else {
  fmt.Println("Write records is successful")
}
```

Python

```
  def write_records(self):
    print("Writing records")
    current_time = self._current_milli_time()

    dimensions = [
      {'Name': 'region', 'Value': 'us-east-1'},
```

```
      {'Name': 'az', 'Value': 'az1'},
      {'Name': 'hostname', 'Value': 'host1'}
    ]

    cpu_utilization = {
      'Dimensions': dimensions,
      'MeasureName': 'cpu_utilization',
      'MeasureValue': '13.5',
      'MeasureValueType': 'DOUBLE',
      'Time': current_time
    }

    memory_utilization = {
      'Dimensions': dimensions,
      'MeasureName': 'memory_utilization',
      'MeasureValue': '40',
      'MeasureValueType': 'DOUBLE',
      'Time': current_time
    }

    records = [cpu_utilization, memory_utilization]

    try:
      result = self.client.write_records(DatabaseName=Constant.DATABASE_NAME,
 TableName=Constant.TABLE_NAME,
                          Records=records, CommonAttributes={})
      print("WriteRecords Status: [%s]" % result['ResponseMetadata']
['HTTPStatusCode'])
    except self.client.exceptions.RejectedRecordsException as err:
      self._print_rejected_records_exceptions(err)
    except Exception as err:
      print("Error:", err)

  @staticmethod
  def _print_rejected_records_exceptions(err):
    print("RejectedRecords: ", err)
    for rr in err.response["RejectedRecords"]:
      print("Rejected Index " + str(rr["RecordIndex"]) + ": " + rr["Reason"])
      if "ExistingVersion" in rr:
        print("Rejected record existing version: ", rr["ExistingVersion"])

  @staticmethod
  def _current_milli_time():
```

```
    return str(int(round(time.time() * 1000)))
```

Node.js

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada aplikasi sampel Node.js Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
async function writeRecords() {
  console.log("Writing records");
  const currentTime = Date.now().toString(); // Unix time in milliseconds

  const dimensions = [
    {'Name': 'region', 'Value': 'us-east-1'},
    {'Name': 'az', 'Value': 'az1'},
    {'Name': 'hostname', 'Value': 'host1'}
  ];

  const cpuUtilization = {
    'Dimensions': dimensions,
    'MeasureName': 'cpu_utilization',
    'MeasureValue': '13.5',
    'MeasureValueType': 'DOUBLE',
    'Time': currentTime.toString()
  };

  const memoryUtilization = {
    'Dimensions': dimensions,
    'MeasureName': 'memory_utilization',
    'MeasureValue': '40',
    'MeasureValueType': 'DOUBLE',
    'Time': currentTime.toString()
  };

  const records = [cpuUtilization, memoryUtilization];

  const params = {
    DatabaseName: constants.DATABASE_NAME,
    TableName: constants.TABLE_NAME,
    Records: records
  };

  const request = writeClient.writeRecords(params);
```

```
  await request.promise().then(
    (data) => {
      console.log("Write records successful");
    },
    (err) => {
      console.log("Error writing records:", err);
      if (err.code === 'RejectedRecordsException') {
        const responsePayload =
 JSON.parse(request.response.httpResponse.body.toString());
        console.log("RejectedRecords: ", responsePayload.RejectedRecords);
        console.log("Other records were written successfully. ");
      }
    }
  );
}
```

.NET

```
   public async Task WriteRecords()
   {
     Console.WriteLine("Writing records");

     DateTimeOffset now = DateTimeOffset.UtcNow;
     string currentTimeString = (now.ToUnixTimeMilliseconds()).ToString();

     List<Dimension> dimensions = new List<Dimension>{
       new Dimension { Name = "region", Value = "us-east-1" },
       new Dimension { Name = "az", Value = "az1" },
       new Dimension { Name = "hostname", Value = "host1" }
     };

     var cpuUtilization = new Record
     {
       Dimensions = dimensions,
       MeasureName = "cpu_utilization",
       MeasureValue = "13.6",
       MeasureValueType = MeasureValueType.DOUBLE,
       Time = currentTimeString
     };

     var memoryUtilization = new Record
     {
       Dimensions = dimensions,
```

```
      MeasureName = "memory_utilization",
      MeasureValue = "40",
      MeasureValueType = MeasureValueType.DOUBLE,
      Time = currentTimeString
    };


    List<Record> records = new List<Record> {
      cpuUtilization,
      memoryUtilization
    };

    try
    {
      var writeRecordsRequest = new WriteRecordsRequest
      {
        DatabaseName = Constants.DATABASE_NAME,
        TableName = Constants.TABLE_NAME,
        Records = records
      };
      WriteRecordsResponse response = await
writeClient.WriteRecordsAsync(writeRecordsRequest);
      Console.WriteLine($"Write records status code:
{response.HttpStatusCode.ToString()}");
    }
    catch (RejectedRecordsException e) {
      Console.WriteLine("RejectedRecordsException:" + e.ToString());
      foreach (RejectedRecord rr in e.RejectedRecords) {
        Console.WriteLine("RecordIndex " + rr.RecordIndex + " : " + rr.Reason);
      }
      Console.WriteLine("Other records were written successfully. ");
    }
    catch (Exception e)
    {
      Console.WriteLine("Write records failure:" + e.ToString());
    }
  }
```

Menulis kumpulan catatan dengan atribut umum

Jika data deret waktu Anda memiliki ukuran dan/atau dimensi yang umum di banyak titik data, Anda juga dapat menggunakan versi optimal berikut writeRecords API untuk memasukkan data

ke Timestream untuk LiveAnalytics. Menggunakan atribut umum dengan batching dapat lebih mengoptimalkan biaya penulisan seperti yang dijelaskan dalamMenghitung jumlah penulisan.

> **Note**
>
> Cuplikan kode ini didasarkan pada contoh aplikasi lengkap. GitHub Untuk informasi selengkapnya tentang cara memulai dengan contoh aplikasi, lihatAplikasi sampel.

Java

```
  public void writeRecordsWithCommonAttributes() {
    System.out.println("Writing records with extracting common attributes");
    // Specify repeated values for all records
    List<Record> records = new ArrayList<>();
    final long time = System.currentTimeMillis();

    List<Dimension> dimensions = new ArrayList<>();
    final Dimension region = new Dimension().withName("region").withValue("us-
east-1");
    final Dimension az = new Dimension().withName("az").withValue("az1");
    final Dimension hostname = new
 Dimension().withName("hostname").withValue("host1");

    dimensions.add(region);
    dimensions.add(az);
    dimensions.add(hostname);

    Record commonAttributes = new Record()
        .withDimensions(dimensions)
        .withMeasureValueType(MeasureValueType.DOUBLE)
        .withTime(String.valueOf(time));

    Record cpuUtilization = new Record()
        .withMeasureName("cpu_utilization")
        .withMeasureValue("13.5");
    Record memoryUtilization = new Record()
        .withMeasureName("memory_utilization")
        .withMeasureValue("40");

    records.add(cpuUtilization);
    records.add(memoryUtilization);
```

```
    WriteRecordsRequest writeRecordsRequest = new WriteRecordsRequest()
        .withDatabaseName(DATABASE_NAME)
        .withTableName(TABLE_NAME)
        .withCommonAttributes(commonAttributes);
    writeRecordsRequest.setRecords(records);

    try {
      WriteRecordsResult writeRecordsResult =
 amazonTimestreamWrite.writeRecords(writeRecordsRequest);
      System.out.println("writeRecordsWithCommonAttributes Status: " +
 writeRecordsResult.getSdkHttpMetadata().getHttpStatusCode());
    } catch (RejectedRecordsException e) {
      System.out.println("RejectedRecords: " + e);
      for (RejectedRecord rejectedRecord : e.getRejectedRecords()) {
        System.out.println("Rejected Index " + rejectedRecord.getRecordIndex() + ":
 "
            + rejectedRecord.getReason());
      }
      System.out.println("Other records were written successfully. ");
    } catch (Exception e) {
      System.out.println("Error: " + e);
    }
  }
```

Java v2

```
  public void writeRecordsWithCommonAttributes() {
    System.out.println("Writing records with extracting common attributes");
    // Specify repeated values for all records
    List<Record> records = new ArrayList<>();
    final long time = System.currentTimeMillis();

    List<Dimension> dimensions = new ArrayList<>();
    final Dimension region = Dimension.builder().name("region").value("us-
east-1").build();
    final Dimension az = Dimension.builder().name("az").value("az1").build();
    final Dimension hostname =
 Dimension.builder().name("hostname").value("host1").build();

    dimensions.add(region);
    dimensions.add(az);
    dimensions.add(hostname);
```

```
    Record commonAttributes = Record.builder()
        .dimensions(dimensions)
        .measureValueType(MeasureValueType.DOUBLE)
        .time(String.valueOf(time)).build();

    Record cpuUtilization = Record.builder()
        .measureName("cpu_utilization")
        .measureValue("13.5").build();
    Record memoryUtilization = Record.builder()
        .measureName("memory_utilization")
        .measureValue("40").build();

    records.add(cpuUtilization);
    records.add(memoryUtilization);

    WriteRecordsRequest writeRecordsRequest = WriteRecordsRequest.builder()
        .databaseName(DATABASE_NAME)
        .tableName(TABLE_NAME)
        .commonAttributes(commonAttributes)
        .records(records).build();

    try {
      WriteRecordsResponse writeRecordsResponse =
 timestreamWriteClient.writeRecords(writeRecordsRequest);
      System.out.println("writeRecordsWithCommonAttributes Status: " +
 writeRecordsResponse.sdkHttpResponse().statusCode());
    } catch (RejectedRecordsException e) {
      System.out.println("RejectedRecords: " + e);
      for (RejectedRecord rejectedRecord : e.rejectedRecords()) {
        System.out.println("Rejected Index " + rejectedRecord.recordIndex() + ": "
            + rejectedRecord.reason());
      }
      System.out.println("Other records were written successfully. ");
    } catch (Exception e) {
      System.out.println("Error: " + e);
    }
  }
```

Go

```
now = time.Now()
currentTimeInSeconds = now.Unix()
```

```
writeRecordsCommonAttributesInput := &timestreamwrite.WriteRecordsInput{
 DatabaseName: aws.String(*databaseName),
 TableName:  aws.String(*tableName),
 CommonAttributes: &timestreamwrite.Record{
  Dimensions: []*timestreamwrite.Dimension{
   &timestreamwrite.Dimension{
    Name:  aws.String("region"),
    Value: aws.String("us-east-1"),
   },
   &timestreamwrite.Dimension{
    Name:  aws.String("az"),
    Value: aws.String("az1"),
   },
   &timestreamwrite.Dimension{
    Name:  aws.String("hostname"),
    Value: aws.String("host1"),
   },
  },
  MeasureValueType: aws.String("DOUBLE"),
  Time:       aws.String(strconv.FormatInt(currentTimeInSeconds, 10)),
  TimeUnit:     aws.String("SECONDS"),
 },
 Records: []*timestreamwrite.Record{
  &timestreamwrite.Record{
   MeasureName:  aws.String("cpu_utilization"),
   MeasureValue: aws.String("13.5"),
  },
  &timestreamwrite.Record{
   MeasureName:  aws.String("memory_utilization"),
   MeasureValue: aws.String("40"),
  },
 },
}

_, err = writeSvc.WriteRecords(writeRecordsCommonAttributesInput)

if err != nil {
 fmt.Println("Error:")
 fmt.Println(err)
} else {
 fmt.Println("Ingest records is successful")
}
```

Python

```
  def write_records_with_common_attributes(self):
    print("Writing records extracting common attributes")
    current_time = self._current_milli_time()

    dimensions = [
      {'Name': 'region', 'Value': 'us-east-1'},
      {'Name': 'az', 'Value': 'az1'},
      {'Name': 'hostname', 'Value': 'host1'}
    ]

    common_attributes = {
      'Dimensions': dimensions,
      'MeasureValueType': 'DOUBLE',
      'Time': current_time
    }

    cpu_utilization = {
      'MeasureName': 'cpu_utilization',
      'MeasureValue': '13.5'
    }

    memory_utilization = {
      'MeasureName': 'memory_utilization',
      'MeasureValue': '40'
    }

    records = [cpu_utilization, memory_utilization]

    try:
      result = self.client.write_records(DatabaseName=Constant.DATABASE_NAME,
 TableName=Constant.TABLE_NAME,
                         Records=records, CommonAttributes=common_attributes)
      print("WriteRecords Status: [%s]" % result['ResponseMetadata']
['HTTPStatusCode'])
    except self.client.exceptions.RejectedRecordsException as err:
      self._print_rejected_records_exceptions(err)
    except Exception as err:
      print("Error:", err)

  @staticmethod
  def _print_rejected_records_exceptions(err):
    print("RejectedRecords: ", err)
```

```
    for rr in err.response["RejectedRecords"]:
      print("Rejected Index " + str(rr["RecordIndex"]) + ": " + rr["Reason"])
      if "ExistingVersion" in rr:
        print("Rejected record existing version: ", rr["ExistingVersion"])

  @staticmethod
  def _current_milli_time():
    return str(int(round(time.time() * 1000)))
```

Node.js

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada aplikasi sampel Node.js Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
async function writeRecordsWithCommonAttributes() {
  console.log("Writing records with common attributes");
  const currentTime = Date.now().toString(); // Unix time in milliseconds

  const dimensions = [
    {'Name': 'region', 'Value': 'us-east-1'},
    {'Name': 'az', 'Value': 'az1'},
    {'Name': 'hostname', 'Value': 'host1'}
  ];

  const commonAttributes = {
    'Dimensions': dimensions,
    'MeasureValueType': 'DOUBLE',
    'Time': currentTime.toString()
  };

  const cpuUtilization = {
    'MeasureName': 'cpu_utilization',
    'MeasureValue': '13.5'
  };

  const memoryUtilization = {
    'MeasureName': 'memory_utilization',
    'MeasureValue': '40'
  };

  const records = [cpuUtilization, memoryUtilization];

  const params = {
```

```
    DatabaseName: constants.DATABASE_NAME,
    TableName: constants.TABLE_NAME,
    Records: records,
    CommonAttributes: commonAttributes
  };

  const request = writeClient.writeRecords(params);

  await request.promise().then(
    (data) => {
      console.log("Write records successful");
    },
    (err) => {
      console.log("Error writing records:", err);
      if (err.code === 'RejectedRecordsException') {
        const responsePayload =
 JSON.parse(request.response.httpResponse.body.toString());
        console.log("RejectedRecords: ", responsePayload.RejectedRecords);
        console.log("Other records were written successfully. ");
      }
    }
  );
}
```

.NET

```
  public async Task WriteRecordsWithCommonAttributes()
  {
    Console.WriteLine("Writing records with common attributes");

    DateTimeOffset now = DateTimeOffset.UtcNow;
    string currentTimeString = (now.ToUnixTimeMilliseconds()).ToString();

    List<Dimension> dimensions = new List<Dimension>{
      new Dimension { Name = "region", Value = "us-east-1" },
      new Dimension { Name = "az", Value = "az1" },
      new Dimension { Name = "hostname", Value = "host1" }
    };

    var commonAttributes = new Record
    {
      Dimensions = dimensions,
      MeasureValueType = MeasureValueType.DOUBLE,
```

```
      Time = currentTimeString
    };

    var cpuUtilization = new Record
    {
      MeasureName = "cpu_utilization",
      MeasureValue = "13.6"
    };

    var memoryUtilization = new Record
    {
      MeasureName = "memory_utilization",
      MeasureValue = "40"
    };


    List<Record> records = new List<Record>();
    records.Add(cpuUtilization);
    records.Add(memoryUtilization);

    try
    {
      var writeRecordsRequest = new WriteRecordsRequest
      {
        DatabaseName = Constants.DATABASE_NAME,
        TableName = Constants.TABLE_NAME,
        Records = records,
        CommonAttributes = commonAttributes
      };
      WriteRecordsResponse response = await
writeClient.WriteRecordsAsync(writeRecordsRequest);
      Console.WriteLine($"Write records status code:
{response.HttpStatusCode.ToString()}");
    }
    catch (RejectedRecordsException e) {
      Console.WriteLine("RejectedRecordsException:" + e.ToString());
      foreach (RejectedRecord rr in e.RejectedRecords) {
        Console.WriteLine("RecordIndex " + rr.RecordIndex + " : " + rr.Reason);
      }
      Console.WriteLine("Other records were written successfully. ");
    }
    catch (Exception e)
    {
      Console.WriteLine("Write records failure:" + e.ToString());
```

```
    }
  }
```

Menambah catatan

Sementara penulisan default di Amazon Timestream mengikuti penulis pertama memenangkan semantik, di mana data disimpan sebagai append saja dan catatan duplikat ditolak, ada aplikasi yang memerlukan kemampuan untuk menulis data ke Amazon Timestream menggunakan penulis terakhir memenangkan semantik, di mana catatan dengan versi tertinggi disimpan dalam sistem. Ada juga aplikasi yang membutuhkan kemampuan untuk memperbarui catatan yang ada. Untuk mengatasi skenario ini, Amazon Timestream menyediakan kemampuan untuk meningkatkan data. Upsert adalah operasi yang menyisipkan catatan ke dalam sistem ketika catatan tidak ada atau memperbarui catatan, ketika ada.

Anda dapat meningkatkan catatan dengan menyertakan definisi catatan `Version` dalam saat mengirim `WriteRecords` permintaan. Amazon Timestream akan menyimpan rekor dengan rekor tertinggi. `Version` Contoh kode di bawah ini menunjukkan bagaimana Anda dapat meningkatkan data:

> **Note**
>
> Cuplikan kode ini didasarkan pada contoh aplikasi lengkap. GitHub Untuk informasi selengkapnya tentang cara memulai dengan contoh aplikasi, lihatAplikasi sampel.

Java

```
  public void writeRecordsWithUpsert() {
    System.out.println("Writing records with upsert");
    // Specify repeated values for all records
    List<Record> records = new ArrayList<>();
    final long time = System.currentTimeMillis();
    // To achieve upsert (last writer wins) semantic, one example is to use current
 time as the version if you are writing directly from the data source
    long version = System.currentTimeMillis();

    List<Dimension> dimensions = new ArrayList<>();
    final Dimension region = new Dimension().withName("region").withValue("us-
east-1");
    final Dimension az = new Dimension().withName("az").withValue("az1");
```

```
   final Dimension hostname = new
Dimension().withName("hostname").withValue("host1");

   dimensions.add(region);
   dimensions.add(az);
   dimensions.add(hostname);

   Record commonAttributes = new Record()
       .withDimensions(dimensions)
       .withMeasureValueType(MeasureValueType.DOUBLE)
       .withTime(String.valueOf(time))
       .withVersion(version);

   Record cpuUtilization = new Record()
       .withMeasureName("cpu_utilization")
       .withMeasureValue("13.5");
   Record memoryUtilization = new Record()
       .withMeasureName("memory_utilization")
       .withMeasureValue("40");

   records.add(cpuUtilization);
   records.add(memoryUtilization);

   WriteRecordsRequest writeRecordsRequest = new WriteRecordsRequest()
       .withDatabaseName(DATABASE_NAME)
       .withTableName(TABLE_NAME)
       .withCommonAttributes(commonAttributes);
   writeRecordsRequest.setRecords(records);

   // write records for first time
   try {
     WriteRecordsResult writeRecordsResult =
amazonTimestreamWrite.writeRecords(writeRecordsRequest);
     System.out.println("WriteRecords Status for first time: " +
writeRecordsResult.getSdkHttpMetadata().getHttpStatusCode());
   } catch (RejectedRecordsException e) {
     printRejectedRecordsException(e);
   } catch (Exception e) {
     System.out.println("Error: " + e);
   }

   // Successfully retry same writeRecordsRequest with same records and versions,
because writeRecords API is idempotent.
   try {
```

```
     WriteRecordsResult writeRecordsResult =
amazonTimestreamWrite.writeRecords(writeRecordsRequest);
     System.out.println("WriteRecords Status for retry: " +
writeRecordsResult.getSdkHttpMetadata().getHttpStatusCode());
   } catch (RejectedRecordsException e) {
     printRejectedRecordsException(e);
   } catch (Exception e) {
     System.out.println("Error: " + e);
   }

   // upsert with lower version, this would fail because a higher version is
required to update the measure value.
   version -= 1;
   commonAttributes.setVersion(version);

   cpuUtilization.setMeasureValue("14.5");
   memoryUtilization.setMeasureValue("50");

   List<Record> upsertedRecords = new ArrayList<>();
   upsertedRecords.add(cpuUtilization);
   upsertedRecords.add(memoryUtilization);

   WriteRecordsRequest writeRecordsUpsertRequest = new WriteRecordsRequest()
       .withDatabaseName(DATABASE_NAME)
       .withTableName(TABLE_NAME)
       .withCommonAttributes(commonAttributes);
   writeRecordsUpsertRequest.setRecords(upsertedRecords);

   try {
     WriteRecordsResult writeRecordsUpsertResult =
amazonTimestreamWrite.writeRecords(writeRecordsUpsertRequest);
     System.out.println("WriteRecords Status for upsert with lower version: " +
writeRecordsUpsertResult.getSdkHttpMetadata().getHttpStatusCode());
   } catch (RejectedRecordsException e) {
     System.out.println("WriteRecords Status for upsert with lower version: ");
     printRejectedRecordsException(e);
   } catch (Exception e) {
     System.out.println("Error: " + e);
   }

   // upsert with higher version as new data in generated
   version = System.currentTimeMillis();
   commonAttributes.setVersion(version);
```

```
    writeRecordsUpsertRequest = new WriteRecordsRequest()
        .withDatabaseName(DATABASE_NAME)
        .withTableName(TABLE_NAME)
        .withCommonAttributes(commonAttributes);
    writeRecordsUpsertRequest.setRecords(upsertedRecords);

    try {
      WriteRecordsResult writeRecordsUpsertResult =
 amazonTimestreamWrite.writeRecords(writeRecordsUpsertRequest);
      System.out.println("WriteRecords Status for upsert with higher version: " +
 writeRecordsUpsertResult.getSdkHttpMetadata().getHttpStatusCode());
    } catch (RejectedRecordsException e) {
      printRejectedRecordsException(e);
    } catch (Exception e) {
      System.out.println("Error: " + e);
    }
  }
```

Java v2

```
  public void writeRecordsWithUpsert() {
    System.out.println("Writing records with upsert");
    // Specify repeated values for all records
    List<Record> records = new ArrayList<>();
    final long time = System.currentTimeMillis();
    // To achieve upsert (last writer wins) semantic, one example is to use current
 time as the version if you are writing directly from the data source
    long version = System.currentTimeMillis();

    List<Dimension> dimensions = new ArrayList<>();
    final Dimension region = Dimension.builder().name("region").value("us-
east-1").build();
    final Dimension az = Dimension.builder().name("az").value("az1").build();
    final Dimension hostname =
 Dimension.builder().name("hostname").value("host1").build();

    dimensions.add(region);
    dimensions.add(az);
    dimensions.add(hostname);

    Record commonAttributes = Record.builder()
        .dimensions(dimensions)
        .measureValueType(MeasureValueType.DOUBLE)
```

```
        .time(String.valueOf(time))
        .version(version)
        .build();

    Record cpuUtilization = Record.builder()
        .measureName("cpu_utilization")
        .measureValue("13.5").build();
    Record memoryUtilization = Record.builder()
        .measureName("memory_utilization")
        .measureValue("40").build();

    records.add(cpuUtilization);
    records.add(memoryUtilization);

    WriteRecordsRequest writeRecordsRequest = WriteRecordsRequest.builder()
        .databaseName(DATABASE_NAME)
        .tableName(TABLE_NAME)
        .commonAttributes(commonAttributes)
        .records(records).build();

    // write records for first time
    try {
      WriteRecordsResponse writeRecordsResponse =
timestreamWriteClient.writeRecords(writeRecordsRequest);
      System.out.println("WriteRecords Status for first time: " +
writeRecordsResponse.sdkHttpResponse().statusCode());
    } catch (RejectedRecordsException e) {
      printRejectedRecordsException(e);
    } catch (Exception e) {
      System.out.println("Error: " + e);
    }

    // Successfully retry same writeRecordsRequest with same records and versions,
because writeRecords API is idempotent.
    try {
      WriteRecordsResponse writeRecordsResponse =
timestreamWriteClient.writeRecords(writeRecordsRequest);
      System.out.println("WriteRecords Status for retry: " +
writeRecordsResponse.sdkHttpResponse().statusCode());
    } catch (RejectedRecordsException e) {
      printRejectedRecordsException(e);
    } catch (Exception e) {
      System.out.println("Error: " + e);
    }
```

```
   // upsert with lower version, this would fail because a higher version is
required to update the measure value.
   version -= 1;
   commonAttributes = Record.builder()
       .dimensions(dimensions)
       .measureValueType(MeasureValueType.DOUBLE)
       .time(String.valueOf(time))
       .version(version)
       .build();

   cpuUtilization = Record.builder()
       .measureName("cpu_utilization")
       .measureValue("14.5").build();
   memoryUtilization = Record.builder()
       .measureName("memory_utilization")
       .measureValue("50").build();

   List<Record> upsertedRecords = new ArrayList<>();
   upsertedRecords.add(cpuUtilization);
   upsertedRecords.add(memoryUtilization);

   WriteRecordsRequest writeRecordsUpsertRequest = WriteRecordsRequest.builder()
       .databaseName(DATABASE_NAME)
       .tableName(TABLE_NAME)
       .commonAttributes(commonAttributes)
       .records(upsertedRecords).build();

   try {
     WriteRecordsResponse writeRecordsResponse =
timestreamWriteClient.writeRecords(writeRecordsUpsertRequest);
     System.out.println("WriteRecords Status for upsert with lower version: " +
writeRecordsResponse.sdkHttpResponse().statusCode());
   } catch (RejectedRecordsException e) {
     System.out.println("WriteRecords Status for upsert with lower version: ");
     printRejectedRecordsException(e);
   } catch (Exception e) {
     System.out.println("Error: " + e);
   }

   // upsert with higher version as new data in generated
   version = System.currentTimeMillis();
   commonAttributes = Record.builder()
       .dimensions(dimensions)
```

```
        .measureValueType(MeasureValueType.DOUBLE)
        .time(String.valueOf(time))
        .version(version)
        .build();

    writeRecordsUpsertRequest = WriteRecordsRequest.builder()
        .databaseName(DATABASE_NAME)
        .tableName(TABLE_NAME)
        .commonAttributes(commonAttributes)
        .records(upsertedRecords).build();

    try {
      WriteRecordsResponse writeRecordsUpsertResponse =
 timestreamWriteClient.writeRecords(writeRecordsUpsertRequest);
      System.out.println("WriteRecords Status for upsert with higher version: " +
 writeRecordsUpsertResponse.sdkHttpResponse().statusCode());
    } catch (RejectedRecordsException e) {
      printRejectedRecordsException(e);
    } catch (Exception e) {
      System.out.println("Error: " + e);
    }
  }
```

Go

```
// Below code will ingest and upsert cpu_utilization and memory_utilization metric
 for a host on
// region=us-east-1, az=az1, and hostname=host1
fmt.Println("Ingesting records and set version as currentTimeInMills, hit enter to
 continue")
reader.ReadString('\n')

// Get current time in seconds.
now = time.Now()
currentTimeInSeconds = now.Unix()
// To achieve upsert (last writer wins) semantic, one example is to use current time
 as the version if you are writing directly from the data source
version := time.Now().Round(time.Millisecond).UnixNano() / 1e6   // set version as
 currentTimeInMills

writeRecordsCommonAttributesUpsertInput := &timestreamwrite.WriteRecordsInput{
 DatabaseName: aws.String(*databaseName),
 TableName:  aws.String(*tableName),
```

```
 CommonAttributes: &timestreamwrite.Record{
  Dimensions: []*timestreamwrite.Dimension{
   &timestreamwrite.Dimension{
    Name:  aws.String("region"),
    Value: aws.String("us-east-1"),
   },
   &timestreamwrite.Dimension{
    Name:  aws.String("az"),
    Value: aws.String("az1"),
   },
   &timestreamwrite.Dimension{
    Name:  aws.String("hostname"),
    Value: aws.String("host1"),
   },
  },
  MeasureValueType: aws.String("DOUBLE"),
  Time:       aws.String(strconv.FormatInt(currentTimeInSeconds, 10)),
  TimeUnit:  aws.String("SECONDS"),
  Version:      &version,
 },
 Records: []*timestreamwrite.Record{
  &timestreamwrite.Record{
   MeasureName:  aws.String("cpu_utilization"),
   MeasureValue: aws.String("13.5"),
  },
  &timestreamwrite.Record{
   MeasureName:  aws.String("memory_utilization"),
   MeasureValue: aws.String("40"),
  },
 },
}

// write records for first time
_, err = writeSvc.WriteRecords(writeRecordsCommonAttributesUpsertInput)

if err != nil {
 fmt.Println("Error:")
 fmt.Println(err)
} else {
 fmt.Println("Frist-time write records is successful")
}

fmt.Println("Retry same writeRecordsRequest with same records and versions. Because
 writeRecords API is idempotent, this will success. hit enter to continue")
```

```
reader.ReadString('\n')

_, err = writeSvc.WriteRecords(writeRecordsCommonAttributesUpsertInput)

if err != nil {
 fmt.Println("Error:")
 fmt.Println(err)
} else {
 fmt.Println("Retry write records for same request is successful")
}

fmt.Println("Upsert with lower version, this would fail because a higher version is
 required to update the measure value. hit enter to continue")
reader.ReadString('\n')
version -= 1
writeRecordsCommonAttributesUpsertInput.CommonAttributes.Version = &version

updated_cpu_utilization := &timestreamwrite.Record{
 MeasureName:    aws.String("cpu_utilization"),
 MeasureValue:   aws.String("14.5"),
}
updated_memory_utilization := &timestreamwrite.Record{
 MeasureName:    aws.String("memory_utilization"),
 MeasureValue:   aws.String("50"),
}


writeRecordsCommonAttributesUpsertInput.Records = []*timestreamwrite.Record{
 updated_cpu_utilization,
 updated_memory_utilization,
}

_, err = writeSvc.WriteRecords(writeRecordsCommonAttributesUpsertInput)

if err != nil {
 fmt.Println("Error:")
 fmt.Println(err)
} else {
 fmt.Println("Write records with lower version is successful")
}

fmt.Println("Upsert with higher version as new data in generated, this would
 success. hit enter to continue")
reader.ReadString('\n')
```

```
version = time.Now().Round(time.Millisecond).UnixNano() / 1e6  // set version as
 currentTimeInMills
writeRecordsCommonAttributesUpsertInput.CommonAttributes.Version = &version

_, err = writeSvc.WriteRecords(writeRecordsCommonAttributesUpsertInput)

if err != nil {
 fmt.Println("Error:")
 fmt.Println(err)
} else {
 fmt.Println("Write records with higher version is successful")
}
```

Python

```
  def write_records_with_upsert(self):
    print("Writing records with upsert")
    current_time = self._current_milli_time()
    # To achieve upsert (last writer wins) semantic, one example is to use current
 time as the version if you are writing directly from the data source
    version = int(self._current_milli_time())

    dimensions = [
          {'Name': 'region', 'Value': 'us-east-1'},
          {'Name': 'az', 'Value': 'az1'},
          {'Name': 'hostname', 'Value': 'host1'}
        ]

    common_attributes = {
      'Dimensions': dimensions,
      'MeasureValueType': 'DOUBLE',
      'Time': current_time,
      'Version': version
    }

    cpu_utilization = {
      'MeasureName': 'cpu_utilization',
      'MeasureValue': '13.5'
    }

    memory_utilization = {
      'MeasureName': 'memory_utilization',
```

```
      'MeasureValue': '40'
    }

    records = [cpu_utilization, memory_utilization]

    # write records for first time
    try:
      result = self.client.write_records(DatabaseName=Constant.DATABASE_NAME,
 TableName=Constant.TABLE_NAME,
                         Records=records, CommonAttributes=common_attributes)
      print("WriteRecords Status for first time: [%s]" % result['ResponseMetadata']
['HTTPStatusCode'])
    except self.client.exceptions.RejectedRecordsException as err:
      self._print_rejected_records_exceptions(err)
    except Exception as err:
      print("Error:", err)

    # Successfully retry same writeRecordsRequest with same records and versions,
 because writeRecords API is idempotent.
    try:
      result = self.client.write_records(DatabaseName=Constant.DATABASE_NAME,
 TableName=Constant.TABLE_NAME,
                         Records=records, CommonAttributes=common_attributes)
      print("WriteRecords Status for retry: [%s]" % result['ResponseMetadata']
['HTTPStatusCode'])
    except self.client.exceptions.RejectedRecordsException as err:
      self._print_rejected_records_exceptions(err)
    except Exception as err:
      print("Error:", err)

    # upsert with lower version, this would fail because a higher version is
 required to update the measure value.
    version -= 1
    common_attributes["Version"] = version

    cpu_utilization["MeasureValue"] = '14.5'
    memory_utilization["MeasureValue"] = '50'

    upsertedRecords = [cpu_utilization, memory_utilization]

    try:
      upsertedResult =
 self.client.write_records(DatabaseName=Constant.DATABASE_NAME,
 TableName=Constant.TABLE_NAME,
```

```
                            Records=upsertedRecords,
 CommonAttributes=common_attributes)
      print("WriteRecords Status for upsert with lower version: [%s]" %
 upsertedResult['ResponseMetadata']['HTTPStatusCode'])
    except self.client.exceptions.RejectedRecordsException as err:
      self._print_rejected_records_exceptions(err)
    except Exception as err:
      print("Error:", err)


    # upsert with higher version as new data is generated
    version = int(self._current_milli_time())
    common_attributes["Version"] = version

    try:
      upsertedResult =
 self.client.write_records(DatabaseName=Constant.DATABASE_NAME,
 TableName=Constant.TABLE_NAME,
                            Records=upsertedRecords,
 CommonAttributes=common_attributes)
      print("WriteRecords Upsert Status: [%s]" % upsertedResult['ResponseMetadata']
['HTTPStatusCode'])
    except self.client.exceptions.RejectedRecordsException as err:
      self._print_rejected_records_exceptions(err)
    except Exception as err:
      print("Error:", err)

  @staticmethod
  def _current_milli_time():
    return str(int(round(time.time() * 1000)))
```

Node.js

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada aplikasi sampel Node.js Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
async function writeRecordsWithUpsert() {
  console.log("Writing records with upsert");
  const currentTime = Date.now().toString(); // Unix time in milliseconds
  // To achieve upsert (last writer wins) semantic, one example is to use current
 time as the version if you are writing directly from the data source
  let version = Date.now();
```

```
const dimensions = [
  {'Name': 'region', 'Value': 'us-east-1'},
  {'Name': 'az', 'Value': 'az1'},
  {'Name': 'hostname', 'Value': 'host1'}
];

const commonAttributes = {
  'Dimensions': dimensions,
  'MeasureValueType': 'DOUBLE',
  'Time': currentTime.toString(),
  'Version': version
};

const cpuUtilization = {
  'MeasureName': 'cpu_utilization',
  'MeasureValue': '13.5'
};

const memoryUtilization = {
  'MeasureName': 'memory_utilization',
  'MeasureValue': '40'
};

const records = [cpuUtilization, memoryUtilization];

const params = {
  DatabaseName: constants.DATABASE_NAME,
  TableName: constants.TABLE_NAME,
  Records: records,
  CommonAttributes: commonAttributes
};

const request = writeClient.writeRecords(params);

// write records for first time
await request.promise().then(
  (data) => {
    console.log("Write records successful for first time.");
  },
  (err) => {
    console.log("Error writing records:", err);
    if (err.code === 'RejectedRecordsException') {
      printRejectedRecordsException(request);
    }
```

```
    }
  );

  // Successfully retry same writeRecordsRequest with same records and versions,
because writeRecords API is idempotent.
  await request.promise().then(
    (data) => {
      console.log("Write records successful for retry.");
    },
    (err) => {
      console.log("Error writing records:", err);
      if (err.code === 'RejectedRecordsException') {
        printRejectedRecordsException(request);
      }
    }
  );

  // upsert with lower version, this would fail because a higher version is required
to update the measure value.
  version--;

  const commonAttributesWithLowerVersion = {
    'Dimensions': dimensions,
    'MeasureValueType': 'DOUBLE',
    'Time': currentTime.toString(),
    'Version': version
  };

  const updatedCpuUtilization = {
    'MeasureName': 'cpu_utilization',
    'MeasureValue': '14.5'
  };

  const updatedMemoryUtilization = {
    'MeasureName': 'memory_utilization',
    'MeasureValue': '50'
  };

  const upsertedRecords = [updatedCpuUtilization, updatedMemoryUtilization];

  const upsertedParamsWithLowerVersion = {
    DatabaseName: constants.DATABASE_NAME,
    TableName: constants.TABLE_NAME,
    Records: upsertedRecords,
```

```
    CommonAttributes: commonAttributesWithLowerVersion
  };

  const upsertRequestWithLowerVersion =
writeClient.writeRecords(upsertedParamsWithLowerVersion);

  await upsertRequestWithLowerVersion.promise().then(
    (data) => {
      console.log("Write records for upsert with lower version successful");
    },
    (err) => {
      console.log("Error writing records:", err);
      if (err.code === 'RejectedRecordsException') {
        printRejectedRecordsException(upsertRequestWithLowerVersion);
      }
    }
  );

  // upsert with higher version as new data in generated
  version = Date.now();

  const commonAttributesWithHigherVersion = {
    'Dimensions': dimensions,
    'MeasureValueType': 'DOUBLE',
    'Time': currentTime.toString(),
    'Version': version
  };

  const upsertedParamsWithHigherVerion = {
    DatabaseName: constants.DATABASE_NAME,
    TableName: constants.TABLE_NAME,
    Records: upsertedRecords,
    CommonAttributes: commonAttributesWithHigherVersion
  };

  const upsertRequestWithHigherVersion =
writeClient.writeRecords(upsertedParamsWithHigherVerion);

  await upsertRequestWithHigherVersion.promise().then(
    (data) => {
      console.log("Write records upsert successful with higher version");
    },
    (err) => {
      console.log("Error writing records:", err);
```

```
      if (err.code === 'RejectedRecordsException') {
        printRejectedRecordsException(upsertedParamsWithHigherVerion);
      }
    }
  );

}
```

.NET

```
  public async Task WriteRecordsWithUpsert()
  {
    Console.WriteLine("Writing records with upsert");

    DateTimeOffset now = DateTimeOffset.UtcNow;
    string currentTimeString = (now.ToUnixTimeMilliseconds()).ToString();
    // To achieve upsert (last writer wins) semantic, one example is to use current
 time as the version if you are writing directly from the data source
    long version = now.ToUnixTimeMilliseconds();

    List<Dimension> dimensions = new List<Dimension>{
      new Dimension { Name = "region", Value = "us-east-1" },
      new Dimension { Name = "az", Value = "az1" },
      new Dimension { Name = "hostname", Value = "host1" }
    };

    var commonAttributes = new Record
    {
      Dimensions = dimensions,
      MeasureValueType = MeasureValueType.DOUBLE,
      Time = currentTimeString,
      Version = version
    };

    var cpuUtilization = new Record
    {
      MeasureName = "cpu_utilization",
      MeasureValue = "13.6"
    };

    var memoryUtilization = new Record
    {
      MeasureName = "memory_utilization",
```

```
      MeasureValue = "40"
    };


    List<Record> records = new List<Record>();
    records.Add(cpuUtilization);
    records.Add(memoryUtilization);

    // write records for first time
    try
    {
      var writeRecordsRequest = new WriteRecordsRequest
      {
        DatabaseName = Constants.DATABASE_NAME,
        TableName = Constants.TABLE_NAME,
        Records = records,
        CommonAttributes = commonAttributes
      };
      WriteRecordsResponse response = await
writeClient.WriteRecordsAsync(writeRecordsRequest);
      Console.WriteLine($"WriteRecords Status for first time:
{response.HttpStatusCode.ToString()}");
    }
    catch (RejectedRecordsException e) {
      PrintRejectedRecordsException(e);
    }
    catch (Exception e)
    {
      Console.WriteLine("Write records failure:" + e.ToString());
    }

    // Successfully retry same writeRecordsRequest with same records and versions,
because writeRecords API is idempotent.
    try
    {
      var writeRecordsRequest = new WriteRecordsRequest
      {
        DatabaseName = Constants.DATABASE_NAME,
        TableName = Constants.TABLE_NAME,
        Records = records,
        CommonAttributes = commonAttributes
      };
      WriteRecordsResponse response = await
writeClient.WriteRecordsAsync(writeRecordsRequest);
```

```
     Console.WriteLine($"WriteRecords Status for retry:
{response.HttpStatusCode.ToString()}");
   }
   catch (RejectedRecordsException e) {
     PrintRejectedRecordsException(e);
   }
   catch (Exception e)
   {
     Console.WriteLine("Write records failure:" + e.ToString());
   }

   // upsert with lower version, this would fail because a higher version is
required to update the measure value.
   version--;
   Type recordType = typeof(Record);
   recordType.GetProperty("Version").SetValue(commonAttributes, version);
   recordType.GetProperty("MeasureValue").SetValue(cpuUtilization, "14.6");
   recordType.GetProperty("MeasureValue").SetValue(memoryUtilization, "50");

   List<Record> upsertedRecords = new List<Record> {
     cpuUtilization,
     memoryUtilization
   };

   try
   {
     var writeRecordsUpsertRequest = new WriteRecordsRequest
     {
       DatabaseName = Constants.DATABASE_NAME,
       TableName = Constants.TABLE_NAME,
       Records = upsertedRecords,
       CommonAttributes = commonAttributes
     };
     WriteRecordsResponse upsertResponse = await
writeClient.WriteRecordsAsync(writeRecordsUpsertRequest);
     Console.WriteLine($"WriteRecords Status for upsert with lower version:
{upsertResponse.HttpStatusCode.ToString()}");
   }
   catch (RejectedRecordsException e) {
     PrintRejectedRecordsException(e);
   }
   catch (Exception e)
   {
     Console.WriteLine("Write records failure:" + e.ToString());
```

```
    }

    // upsert with higher version as new data in generated
    now = DateTimeOffset.UtcNow;
    version = now.ToUnixTimeMilliseconds();
    recordType.GetProperty("Version").SetValue(commonAttributes, version);

    try
    {
      var writeRecordsUpsertRequest = new WriteRecordsRequest
      {
        DatabaseName = Constants.DATABASE_NAME,
        TableName = Constants.TABLE_NAME,
        Records = upsertedRecords,
        CommonAttributes = commonAttributes
      };
      WriteRecordsResponse upsertResponse = await
writeClient.WriteRecordsAsync(writeRecordsUpsertRequest);
      Console.WriteLine($"WriteRecords Status for upsert with higher version:
{upsertResponse.HttpStatusCode.ToString()}");
    }
    catch (RejectedRecordsException e) {
      PrintRejectedRecordsException(e);
    }
    catch (Exception e)
    {
      Console.WriteLine("Write records failure:" + e.ToString());
    }
  }
```

Contoh atribut multi-ukuran

Contoh ini menggambarkan penulisan atribut multi-mearure. Atribut multi-ukuran berguna ketika perangkat atau aplikasi yang Anda lacak memancarkan beberapa metrik atau peristiwa pada stempel waktu yang sama..

> **Note**
>
> Cuplikan kode ini didasarkan pada contoh aplikasi lengkap. GitHub Untuk informasi selengkapnya tentang cara memulai dengan contoh aplikasi, lihatAplikasi sampel.

Java

```
package com.amazonaws.services.timestream;

import static com.amazonaws.services.timestream.Main.DATABASE_NAME;
import static com.amazonaws.services.timestream.Main.REGION;
import static com.amazonaws.services.timestream.Main.TABLE_NAME;

import java.util.ArrayList;
import java.util.List;

import com.amazonaws.services.timestreamwrite.AmazonTimestreamWrite;
import com.amazonaws.services.timestreamwrite.model.Dimension;
import com.amazonaws.services.timestreamwrite.model.MeasureValue;
import com.amazonaws.services.timestreamwrite.model.MeasureValueType;
import com.amazonaws.services.timestreamwrite.model.Record;
import com.amazonaws.services.timestreamwrite.model.RejectedRecordsException;
import com.amazonaws.services.timestreamwrite.model.WriteRecordsRequest;
import com.amazonaws.services.timestreamwrite.model.WriteRecordsResult;


public class multimeasureAttributeExample {
  AmazonTimestreamWrite timestreamWriteClient;

  public multimeasureAttributeExample(AmazonTimestreamWrite client) {
    this.timestreamWriteClient = client;
  }

  public void writeRecordsMultiMeasureValueSingleRecord() {
    System.out.println("Writing records with multi value attributes");

    List<Record> records = new ArrayList<>();
    final long time = System.currentTimeMillis();
    long version = System.currentTimeMillis();

    List<Dimension> dimensions = new ArrayList<>();
    final Dimension region = new Dimension().withName("region").withValue(REGION);
    final Dimension az = new Dimension().withName("az").withValue("az1");
    final Dimension hostname = new
 Dimension().withName("hostname").withValue("host1");

    dimensions.add(region);
    dimensions.add(az);
    dimensions.add(hostname);
```

```
    Record commonAttributes = new Record()
        .withDimensions(dimensions)
        .withTime(String.valueOf(time))
        .withVersion(version);

    MeasureValue cpuUtilization = new MeasureValue()
        .withName("cpu_utilization")
        .withType(MeasureValueType.DOUBLE)
        .withValue("13.5");
    MeasureValue memoryUtilization = new MeasureValue()
        .withName("memory_utilization")
        .withType(MeasureValueType.DOUBLE)
        .withValue("40");
    Record computationalResources = new Record()
        .withMeasureName("cpu_memory")
        .withMeasureValues(cpuUtilization, memoryUtilization)
        .withMeasureValueType(MeasureValueType.MULTI);

    records.add(computationalResources);

    WriteRecordsRequest writeRecordsRequest = new WriteRecordsRequest()
        .withDatabaseName(DATABASE_NAME)
        .withTableName(TABLE_NAME)
        .withCommonAttributes(commonAttributes)
        .withRecords(records);

    // write records for first time
    try {
      WriteRecordsResult writeRecordResult =
timestreamWriteClient.writeRecords(writeRecordsRequest);
      System.out.println(
          "WriteRecords Status for multi value attributes: " + writeRecordResult
              .getSdkHttpMetadata().getHttpStatusCode());
    } catch (RejectedRecordsException e) {
      printRejectedRecordsException(e);
    } catch (Exception e) {
      System.out.println("Error: " + e);
    }
  }

  public void writeRecordsMultiMeasureValueMultipleRecords() {
    System.out.println(
        "Writing records with multi value attributes mixture type");
```

```
    List<Record> records = new ArrayList<>();
    final long time = System.currentTimeMillis();
    long version = System.currentTimeMillis();

    List<Dimension> dimensions = new ArrayList<>();
    final Dimension region = new Dimension().withName("region").withValue(REGION);
    final Dimension az = new Dimension().withName("az").withValue("az1");
    final Dimension hostname = new
Dimension().withName("hostname").withValue("host1");

    dimensions.add(region);
    dimensions.add(az);
    dimensions.add(hostname);

    Record commonAttributes = new Record()
        .withDimensions(dimensions)
        .withTime(String.valueOf(time))
        .withVersion(version);

    MeasureValue cpuUtilization = new MeasureValue()
        .withName("cpu_utilization")
        .withType(MeasureValueType.DOUBLE)
        .withValue("13");
    MeasureValue memoryUtilization =new MeasureValue()
        .withName("memory_utilization")
        .withType(MeasureValueType.DOUBLE)
        .withValue("40");
    MeasureValue activeCores = new MeasureValue()
        .withName("active_cores")
        .withType(MeasureValueType.BIGINT)
        .withValue("4");


    Record computationalResources = new Record()
        .withMeasureName("computational_utilization")
        .withMeasureValues(cpuUtilization, memoryUtilization, activeCores)
        .withMeasureValueType(MeasureValueType.MULTI);

    records.add(computationalResources);

    WriteRecordsRequest writeRecordsRequest = new WriteRecordsRequest()
        .withDatabaseName(DATABASE_NAME)
        .withTableName(TABLE_NAME)
```

```
        .withCommonAttributes(commonAttributes)
        .withRecords(records);

    // write records for first time
    try {
      WriteRecordsResult writeRecordResult =
 timestreamWriteClient.writeRecords(writeRecordsRequest);
      System.out.println(
          "WriteRecords Status for multi value attributes: " + writeRecordResult
              .getSdkHttpMetadata().getHttpStatusCode());
    } catch (RejectedRecordsException e) {
      printRejectedRecordsException(e);
    } catch (Exception e) {
      System.out.println("Error: " + e);
    }
  }

  private void printRejectedRecordsException(RejectedRecordsException e) {
    System.out.println("RejectedRecords: " + e);
    e.getRejectedRecords().forEach(System.out::println);
  }
}
```

Java v2

```
package com.amazonaws.services.timestream;

import java.util.ArrayList;
import java.util.List;

import software.amazon.awssdk.services.timestreamwrite.TimestreamWriteClient;
import software.amazon.awssdk.services.timestreamwrite.model.Dimension;
import software.amazon.awssdk.services.timestreamwrite.model.MeasureValue;
import software.amazon.awssdk.services.timestreamwrite.model.MeasureValueType;
import software.amazon.awssdk.services.timestreamwrite.model.Record;
import
 software.amazon.awssdk.services.timestreamwrite.model.RejectedRecordsException;
import software.amazon.awssdk.services.timestreamwrite.model.WriteRecordsRequest;
import software.amazon.awssdk.services.timestreamwrite.model.WriteRecordsResponse;

import static com.amazonaws.services.timestream.Main.DATABASE_NAME;
import static com.amazonaws.services.timestream.Main.TABLE_NAME;
```

```
public class multimeasureAttributeExample {

  TimestreamWriteClient timestreamWriteClient;

  public multimeasureAttributeExample(TimestreamWriteClient client) {
    this.timestreamWriteClient = client;
  }

  public void writeRecordsMultiMeasureValueSingleRecord() {
    System.out.println("Writing records with multi value attributes");

    List<Record> records = new ArrayList<>();
    final long time = System.currentTimeMillis();
    long version = System.currentTimeMillis();

    List<Dimension> dimensions = new ArrayList<>();
    final Dimension region =
        Dimension.builder().name("region").value("us-east-1").build();
    final Dimension az = Dimension.builder().name("az").value("az1").build();
    final Dimension hostname =
        Dimension.builder().name("hostname").value("host1").build();

    dimensions.add(region);
    dimensions.add(az);
    dimensions.add(hostname);

    Record commonAttributes = Record.builder()
        .dimensions(dimensions)
        .time(String.valueOf(time))
        .version(version)
        .build();

    MeasureValue cpuUtilization = MeasureValue.builder()
        .name("cpu_utilization")
        .type(MeasureValueType.DOUBLE)
        .value("13.5").build();
    MeasureValue memoryUtilization = MeasureValue.builder()
        .name("memory_utilization")
        .type(MeasureValueType.DOUBLE)
        .value("40").build();
    Record computationalResources = Record
        .builder()
        .measureName("cpu_memory")
```

```
        .measureValues(cpuUtilization, memoryUtilization)
        .measureValueType(MeasureValueType.MULTI)
        .build();

    records.add(computationalResources);

    WriteRecordsRequest writeRecordsRequest = WriteRecordsRequest.builder()
        .databaseName(DATABASE_NAME)
        .tableName(TABLE_NAME)
        .commonAttributes(commonAttributes)
        .records(records).build();

    // write records for first time
    try {
      WriteRecordsResponse writeRecordsResponse =
timestreamWriteClient.writeRecords(writeRecordsRequest);
      System.out.println(
          "WriteRecords Status for multi value attributes: " + writeRecordsResponse
              .sdkHttpResponse()
              .statusCode());
    } catch (RejectedRecordsException e) {
      printRejectedRecordsException(e);
    } catch (Exception e) {
      System.out.println("Error: " + e);
    }
  }

  public void writeRecordsMultiMeasureValueMultipleRecords() {
    System.out.println(
        "Writing records with multi value attributes mixture type");

    List<Record> records = new ArrayList<>();
    final long time = System.currentTimeMillis();
    long version = System.currentTimeMillis();

    List<Dimension> dimensions = new ArrayList<>();
    final Dimension region =
        Dimension.builder().name("region").value("us-east-1").build();
    final Dimension az = Dimension.builder().name("az").value("az1").build();
    final Dimension hostname =
        Dimension.builder().name("hostname").value("host1").build();

    dimensions.add(region);
    dimensions.add(az);
```

```
    dimensions.add(hostname);

    Record commonAttributes = Record.builder()
        .dimensions(dimensions)
        .time(String.valueOf(time))
        .version(version)
        .build();

    MeasureValue cpuUtilization = MeasureValue.builder()
        .name("cpu_utilization")
        .type(MeasureValueType.DOUBLE)
        .value("13.5").build();
    MeasureValue memoryUtilization = MeasureValue.builder()
        .name("memory_utilization")
        .type(MeasureValueType.DOUBLE)
        .value("40").build();
    MeasureValue activeCores = MeasureValue.builder()
        .name("active_cores")
        .type(MeasureValueType.BIGINT)
        .value("4").build();


    Record computationalResources = Record
        .builder()
        .measureName("computational_utilization")
        .measureValues(cpuUtilization, memoryUtilization, activeCores)
        .measureValueType(MeasureValueType.MULTI)
        .build();

    records.add(computationalResources);

    WriteRecordsRequest writeRecordsRequest = WriteRecordsRequest.builder()
        .databaseName(DATABASE_NAME)
        .tableName(TABLE_NAME)
        .commonAttributes(commonAttributes)
        .records(records).build();

    // write records for first time
    try {
      WriteRecordsResponse writeRecordsResponse =
timestreamWriteClient.writeRecords(writeRecordsRequest);
     System.out.println(
         "WriteRecords Status for multi value attributes: " + writeRecordsResponse
             .sdkHttpResponse()
```

```
              .statusCode());
    } catch (RejectedRecordsException e) {
      printRejectedRecordsException(e);
    } catch (Exception e) {
      System.out.println("Error: " + e);
    }
  }

  private void printRejectedRecordsException(RejectedRecordsException e) {
    System.out.println("RejectedRecords: " + e);
    e.rejectedRecords().forEach(System.out::println);
  }
}
```

Go

```
  now := time.Now()
  currentTimeInSeconds := now.Unix()
  writeRecordsInput := &timestreamwrite.WriteRecordsInput{
    DatabaseName: aws.String(*databaseName),
    TableName:  aws.String(*tableName),
    Records: []*timestreamwrite.Record{
    &timestreamwrite.Record{
      Dimensions: []*timestreamwrite.Dimension{
      &timestreamwrite.Dimension{
        Name:  aws.String("region"),
        Value: aws.String("us-east-1"),
      },
      &timestreamwrite.Dimension{
        Name:  aws.String("az"),
        Value: aws.String("az1"),
      },
      &timestreamwrite.Dimension{
        Name:  aws.String("hostname"),
        Value: aws.String("host1"),
      },
      },
      MeasureName:  aws.String("metrics"),
      MeasureValueType: aws.String("MULTI"),
      Time:     aws.String(strconv.FormatInt(currentTimeInSeconds, 10)),
      TimeUnit:  aws.String("SECONDS"),
      MeasureValues: []*timestreamwrite.MeasureValue{
      &timestreamwrite.MeasureValue{
```

```
        Name:  aws.String("cpu_utilization"),
        Value: aws.String("13.5"),
        Type:  aws.String("DOUBLE"),
      },
      &timestreamwrite.MeasureValue{
        Name:  aws.String("memory_utilization"),
        Value: aws.String("40"),
        Type:  aws.String("DOUBLE"),
      },
      },
    },
    },
  }

  _, err = writeSvc.WriteRecords(writeRecordsInput)

  if err != nil {
    fmt.Println("Error:")
    fmt.Println(err)
  } else {
    fmt.Println("Write records is successful")
  }
```

Python

```
import time
import boto3
import psutil
import os

from botocore.config import Config

DATABASE_NAME = os.environ['DATABASE_NAME']
TABLE_NAME = os.environ['TABLE_NAME']

COUNTRY = "UK"
CITY = "London"
HOSTNAME = "MyHostname" # You can make it dynamic using socket.gethostname()

INTERVAL = 1 # Seconds

def prepare_common_attributes():
  common_attributes = {
```

```
    'Dimensions': [
      {'Name': 'country', 'Value': COUNTRY},
      {'Name': 'city', 'Value': CITY},
      {'Name': 'hostname', 'Value': HOSTNAME}
    ],
    'MeasureName': 'utilization',
    'MeasureValueType': 'MULTI'
  }
  return common_attributes


def prepare_record(current_time):
  record = {
    'Time': str(current_time),
    'MeasureValues': []
  }
  return record


def prepare_measure(measure_name, measure_value):
  measure = {
    'Name': measure_name,
    'Value': str(measure_value),
    'Type': 'DOUBLE'
  }
  return measure


def write_records(records, common_attributes):
  try:
    result = write_client.write_records(DatabaseName=DATABASE_NAME,
                                        TableName=TABLE_NAME,
                                        CommonAttributes=common_attributes,
                                        Records=records)
    status = result['ResponseMetadata']['HTTPStatusCode']
    print("Processed %d records. WriteRecords HTTPStatusCode: %s" %
        (len(records), status))
  except Exception as err:
    print("Error:", err)


if __name__ == '__main__':

  print("writing data to database {} table {}".format(
```

```
  DATABASE_NAME, TABLE_NAME))

session = boto3.Session()
write_client = session.client('timestream-write', config=Config(
  read_timeout=20, max_pool_connections=5000, retries={'max_attempts': 10}))
query_client = session.client('timestream-query') # Not used

common_attributes = prepare_common_attributes()

records = []

while True:

  current_time = int(time.time() * 1000)
  cpu_utilization = psutil.cpu_percent()
  memory_utilization = psutil.virtual_memory().percent
  swap_utilization = psutil.swap_memory().percent
  disk_utilization = psutil.disk_usage('/').percent

  record = prepare_record(current_time)
  record['MeasureValues'].append(prepare_measure('cpu', cpu_utilization))
  record['MeasureValues'].append(prepare_measure('memory', memory_utilization))
  record['MeasureValues'].append(prepare_measure('swap', swap_utilization))
  record['MeasureValues'].append(prepare_measure('disk', disk_utilization))

  records.append(record)

  print("records {} - cpu {} - memory {} - swap {} - disk {}".format(
    len(records), cpu_utilization, memory_utilization,
    swap_utilization, disk_utilization))

  if len(records) == 100:
    write_records(records, common_attributes)
    records = []

  time.sleep(INTERVAL)
```

Node.js

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada aplikasi sampel Node.js Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
async function writeRecords() {
```

```
    console.log("Writing records");
    const currentTime = Date.now().toString(); // Unix time in milliseconds

    const dimensions = [
    {'Name': 'region', 'Value': 'us-east-1'},
    {'Name': 'az', 'Value': 'az1'},
    {'Name': 'hostname', 'Value': 'host1'}
    ];

    const record = {
    'Dimensions': dimensions,
    'MeasureName': 'metrics',
    'MeasureValues': [
      {
        'Name': 'cpu_utilization',
        'Value': '40',
        'Type': 'DOUBLE',
      },
      {
        'Name': 'memory_utilization',
        'Value': '13.5',
        'Type': 'DOUBLE',
      },
      ],
      'MeasureValueType': 'MULTI',
      'Time': currentTime.toString()
    }

    const records = [record];

    const params = {
    DatabaseName: 'DatabaseName',
    TableName: 'TableName',
    Records: records
    };

    const response = await writeClient.writeRecords(params);

    console.log(response);
  }
```

.NET

```
using System;
using System.IO;
using System.Collections.Generic;
using Amazon.TimestreamWrite;
using Amazon.TimestreamWrite.Model;
using System.Threading.Tasks;

namespace TimestreamDotNetSample
{
  static class MultiMeasureValueConstants
  {
    public const string MultiMeasureValueSampleDb = "multiMeasureValueSampleDb";
    public const string MultiMeasureValueSampleTable =
 "multiMeasureValueSampleTable";
  }

  public class MultiValueAttributesExample
  {
    private readonly AmazonTimestreamWriteClient writeClient;

    public MultiValueAttributesExample(AmazonTimestreamWriteClient writeClient)
    {
      this.writeClient = writeClient;
    }

    public async Task WriteRecordsMultiMeasureValueSingleRecord()
    {
      Console.WriteLine("Writing records with multi value attributes");

      DateTimeOffset now = DateTimeOffset.UtcNow;
      string currentTimeString = (now.ToUnixTimeMilliseconds()).ToString();

      List<Dimension> dimensions = new List<Dimension>{
        new Dimension { Name = "region", Value = "us-east-1" },
        new Dimension { Name = "az", Value = "az1" },
        new Dimension { Name = "hostname", Value = "host1" }
      };

      var commonAttributes = new Record
      {
        Dimensions = dimensions,
        Time = currentTimeString
```

```
      };

      var cpuUtilization = new MeasureValue
      {
        Name = "cpu_utilization",
        Value = "13.6",
        Type = "DOUBLE"
      };

      var memoryUtilization = new MeasureValue
      {
        Name = "memory_utilization",
        Value = "40",
        Type = "DOUBLE"
      };

      var computationalRecord = new Record
      {
        MeasureName = "cpu_memory",
        MeasureValues = new List<MeasureValue> {cpuUtilization, memoryUtilization},
        MeasureValueType = "MULTI"
      };


      List<Record> records = new List<Record>();
      records.Add(computationalRecord);

      try
      {
        var writeRecordsRequest = new WriteRecordsRequest
        {
          DatabaseName = MultiMeasureValueConstants.MultiMeasureValueSampleDb,
          TableName = MultiMeasureValueConstants.MultiMeasureValueSampleTable,
          Records = records,
          CommonAttributes = commonAttributes
        };
        WriteRecordsResponse response = await
writeClient.WriteRecordsAsync(writeRecordsRequest);
        Console.WriteLine($"Write records status code:
{response.HttpStatusCode.ToString()}");
      }
      catch (Exception e)
      {
        Console.WriteLine("Write records failure:" + e.ToString());
```

```
    }
}

public async Task WriteRecordsMultiMeasureValueMultipleRecords()
{
    Console.WriteLine("Writing records with multi value attributes mixture type");

    DateTimeOffset now = DateTimeOffset.UtcNow;
    string currentTimeString = (now.ToUnixTimeMilliseconds()).ToString();

    List<Dimension> dimensions = new List<Dimension>{
        new Dimension { Name = "region", Value = "us-east-1" },
        new Dimension { Name = "az", Value = "az1" },
        new Dimension { Name = "hostname", Value = "host1" }
    };

    var commonAttributes = new Record
    {
        Dimensions = dimensions,
        Time = currentTimeString
    };

    var cpuUtilization = new MeasureValue
    {
        Name = "cpu_utilization",
        Value = "13.6",
        Type = "DOUBLE"
    };

    var memoryUtilization = new MeasureValue
    {
        Name = "memory_utilization",
        Value = "40",
        Type = "DOUBLE"
    };

    var activeCores = new MeasureValue
    {
        Name = "active_cores",
        Value = "4",
        Type = "BIGINT"
    };

    var computationalRecord = new Record
```

```
      {
        MeasureName = "computational_utilization",
        MeasureValues = new List<MeasureValue> {cpuUtilization, memoryUtilization,
 activeCores},
        MeasureValueType = "MULTI"
      };

      var aliveRecord = new Record
      {
        MeasureName = "is_healthy",
        MeasureValue = "true",
        MeasureValueType = "BOOLEAN"
      };

      List<Record> records = new List<Record>();
      records.Add(computationalRecord);
      records.Add(aliveRecord);

      try
      {
        var writeRecordsRequest = new WriteRecordsRequest
        {
          DatabaseName = MultiMeasureValueConstants.MultiMeasureValueSampleDb,
          TableName = MultiMeasureValueConstants.MultiMeasureValueSampleTable,
          Records = records,
          CommonAttributes = commonAttributes
        };
        WriteRecordsResponse response = await
 writeClient.WriteRecordsAsync(writeRecordsRequest);
        Console.WriteLine($"Write records status code:
 {response.HttpStatusCode.ToString()}");
      }
      catch (Exception e)
      {
        Console.WriteLine("Write records failure:" + e.ToString());
      }
    }
  }
}
```

Menangani kegagalan menulis

Menulis di Amazon Timestream dapat gagal karena satu atau beberapa alasan berikut:

- Ada catatan dengan stempel waktu yang berada di luar durasi retensi penyimpanan memori.
- Ada catatan yang berisi dimensi dan/atau ukuran yang melebihi batas yang ditentukan Timestream.
- Amazon Timestream telah mendeteksi catatan duplikat. Catatan ditandai sebagai duplikat, ketika ada beberapa catatan dengan dimensi, stempel waktu, dan nama ukuran yang sama tetapi:
  - Nilai ukur berbeda.
  - Versi tidak hadir dalam permintaan atau nilai versi dalam catatan baru sama dengan atau lebih rendah dari nilai yang ada. Jika Amazon Timestream menolak data karena alasan ini, `ExistingVersion` bidang di dalamnya `RejectedRecords` akan berisi versi rekaman saat ini yang disimpan di Amazon Timestream. Untuk memaksa pembaruan, Anda dapat mengirim ulang permintaan dengan versi untuk catatan yang ditetapkan ke nilai yang lebih besar dari`ExistingVersion`.

Untuk informasi selengkapnya tentang kesalahan dan catatan yang ditolak, lihat Kesalahan dan RejectedRecord.

Jika aplikasi Anda menerima `RejectedRecordsException` saat mencoba menulis catatan ke Timestream, Anda dapat mengurai catatan yang ditolak untuk mempelajari lebih lanjut tentang kegagalan penulisan seperti yang ditunjukkan di bawah ini.

> **Note**
>
> Cuplikan kode ini didasarkan pada contoh aplikasi lengkap. GitHub Untuk informasi selengkapnya tentang cara memulai dengan contoh aplikasi, lihatAplikasi sampel.

Java

```
 try {
   WriteRecordsResult writeRecordsResult =
amazonTimestreamWrite.writeRecords(writeRecordsRequest);
   System.out.println("WriteRecords Status: " +
writeRecordsResult.getSdkHttpMetadata().getHttpStatusCode());
 } catch (RejectedRecordsException e) {
   System.out.println("RejectedRecords: " + e);
```

```
    for (RejectedRecord rejectedRecord : e.getRejectedRecords()) {
      System.out.println("Rejected Index " + rejectedRecord.getRecordIndex() + ": "
          + rejectedRecord.getReason());
    }
    System.out.println("Other records were written successfully. ");
  } catch (Exception e) {
    System.out.println("Error: " + e);
  }
```

Java v2

```
    try {
      WriteRecordsResponse writeRecordsResponse =
 timestreamWriteClient.writeRecords(writeRecordsRequest);
      System.out.println("writeRecordsWithCommonAttributes Status: " +
 writeRecordsResponse.sdkHttpResponse().statusCode());
    } catch (RejectedRecordsException e) {
      System.out.println("RejectedRecords: " + e);
      for (RejectedRecord rejectedRecord : e.rejectedRecords()) {
        System.out.println("Rejected Index " + rejectedRecord.recordIndex() + ": "
            + rejectedRecord.reason());
      }
      System.out.println("Other records were written successfully. ");
    } catch (Exception e) {
      System.out.println("Error: " + e);
    }
```

Go

```
_, err = writeSvc.WriteRecords(writeRecordsInput)

if err != nil {
  fmt.Println("Error:")
  fmt.Println(err)
} else {
  fmt.Println("Write records is successful")
}
```

Python

```
try:
  result = self.client.write_records(DatabaseName=Constant.DATABASE_NAME,
 TableName=Constant.TABLE_NAME, Records=records, CommonAttributes=common_attributes)
```

```
  print("WriteRecords Status: [%s]" % result['ResponseMetadata']['HTTPStatusCode'])
except self.client.exceptions.RejectedRecordsException as err:
  print("RejectedRecords: ", err)
  for rr in err.response["RejectedRecords"]:
    print("Rejected Index " + str(rr["RecordIndex"]) + ": " + rr["Reason"])
  print("Other records were written successfully. ")
except Exception as err:
  print("Error:", err)
```

Node.js

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada aplikasi sampel Node.js Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
await request.promise().then(
    (data) => {
      console.log("Write records successful");
    },
    (err) => {
      console.log("Error writing records:", err);
      if (err.code === 'RejectedRecordsException') {
        const responsePayload =
 JSON.parse(request.response.httpResponse.body.toString());
        console.log("RejectedRecords: ", responsePayload.RejectedRecords);
        console.log("Other records were written successfully. ");
      }
    }
  );
```

.NET

```
  try
  {
    var writeRecordsRequest = new WriteRecordsRequest
    {
      DatabaseName = Constants.DATABASE_NAME,
      TableName = Constants.TABLE_NAME,
      Records = records,
      CommonAttributes = commonAttributes
    };
    WriteRecordsResponse response = await
 writeClient.WriteRecordsAsync(writeRecordsRequest);
```

```
    Console.WriteLine($"Write records status code:
{response.HttpStatusCode.ToString()}");
 }
 catch (RejectedRecordsException e) {
    Console.WriteLine("RejectedRecordsException:" + e.ToString());
    foreach (RejectedRecord rr in e.RejectedRecords) {
      Console.WriteLine("RecordIndex " + rr.RecordIndex + " : " + rr.Reason);
    }
    Console.WriteLine("Other records were written successfully. ");
 }
 catch (Exception e)
 {
    Console.WriteLine("Write records failure:" + e.ToString());
 }
```

Jalankan kueri

Topik

- Hasil paginating
- Mengurai set hasil
- Mengakses status kueri

Hasil paginating

Saat Anda menjalankan kueri, Timestream mengembalikan hasil yang disetel dengan cara paginasi untuk mengoptimalkan respons aplikasi Anda. Cuplikan kode di bawah ini menunjukkan bagaimana Anda dapat melakukan paginasi melalui kumpulan hasil. Anda harus mengulang semua halaman set hasil sampai Anda menemukan nilai null. Token pagination kedaluwarsa 3 jam setelah dikeluarkan oleh Timestream untuk. LiveAnalytics

> **Note**
>
> Cuplikan kode ini didasarkan pada contoh aplikasi lengkap. GitHub Untuk informasi selengkapnya tentang cara memulai dengan contoh aplikasi, lihatAplikasi sampel.

Java

```
   private void runQuery(String queryString) {
       try {
           QueryRequest queryRequest = new QueryRequest();
           queryRequest.setQueryString(queryString);
           QueryResult queryResult = queryClient.query(queryRequest);
           while (true) {
               parseQueryResult(queryResult);
               if (queryResult.getNextToken() == null) {
                   break;
               }
               queryRequest.setNextToken(queryResult.getNextToken());
               queryResult = queryClient.query(queryRequest);
           }
       } catch (Exception e) {
           // Some queries might fail with 500 if the result of a sequence function
has more than 10000 entries
           e.printStackTrace();
       }
   }
```

Java v2

```
   private void runQuery(String queryString) {
       try {
           QueryRequest queryRequest =
QueryRequest.builder().queryString(queryString).build();
           final QueryIterable queryResponseIterator =
timestreamQueryClient.queryPaginator(queryRequest);
           for(QueryResponse queryResponse : queryResponseIterator) {
               parseQueryResult(queryResponse);
           }
       } catch (Exception e) {
           // Some queries might fail with 500 if the result of a sequence function
has more than 10000 entries
           e.printStackTrace();
       }
   }
```

Go

```
func runQuery(queryPtr *string, querySvc *timestreamquery.TimestreamQuery, f
 *os.File) {
    queryInput := &timestreamquery.QueryInput{
        QueryString: aws.String(*queryPtr),
    }
    fmt.Println("QueryInput:")
    fmt.Println(queryInput)
    // execute the query
    err := querySvc.QueryPages(queryInput,
        func(page *timestreamquery.QueryOutput, lastPage bool) bool {
            // process query response
            queryStatus := page.QueryStatus
            fmt.Println("Current query status:", queryStatus)
            // query response metadata
            // includes column names and types
            metadata := page.ColumnInfo
            // fmt.Println("Metadata:")
            fmt.Println(metadata)
            header := ""
            for i := 0; i < len(metadata); i++ {
                header += *metadata[i].Name
                if i != len(metadata)-1 {
                    header += ", "
                }
            }
            write(f, header)

            // query response data
            fmt.Println("Data:")
            // process rows
            rows := page.Rows
            for i := 0; i < len(rows); i++ {
                data := rows[i].Data
                value := processRowType(data, metadata)
                fmt.Println(value)
                write(f, value)
            }
            fmt.Println("Number of rows:", len(page.Rows))
            return true
        })
    if err != nil {
        fmt.Println("Error:")
```

```
        fmt.Println(err)
    }
}
```

Python

```
    def run_query(self, query_string):
        try:
            page_iterator = self.paginator.paginate(QueryString=query_string)
            for page in page_iterator:
                self._parse_query_result(page)
        except Exception as err:
            print("Exception while running query:", err)
```

Node.js

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada aplikasi sampel Node.js Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
async function getAllRows(query, nextToken) {
    const params = {
        QueryString: query
    };

    if (nextToken) {
        params.NextToken = nextToken;
    }

    await queryClient.query(params).promise()
        .then(
            (response) => {
                parseQueryResult(response);
                if (response.NextToken) {
                    getAllRows(query, response.NextToken);
                }
            },
            (err) => {
                console.error("Error while querying:", err);
            });
}
```

.NET

```
        private async Task RunQueryAsync(string queryString)
        {
            try
            {
                QueryRequest queryRequest = new QueryRequest();
                queryRequest.QueryString = queryString;
                QueryResponse queryResponse = await
queryClient.QueryAsync(queryRequest);
                while (true)
                {
                    ParseQueryResult(queryResponse);
                    if (queryResponse.NextToken == null)
                    {
                        break;
                    }
                    queryRequest.NextToken = queryResponse.NextToken;
                    queryResponse = await queryClient.QueryAsync(queryRequest);
                }
            } catch(Exception e)
            {
                // Some queries might fail with 500 if the result of a sequence
function has more than 10000 entries
                Console.WriteLine(e.ToString());
            }
        }
```

Mengurai set hasil

Anda dapat menggunakan cuplikan kode berikut untuk mengekstrak data dari kumpulan hasil. Hasil kueri dapat diakses hingga 24 jam setelah kueri selesai.

> **Note**
>
> Cuplikan kode ini didasarkan pada contoh aplikasi lengkap. GitHub Untuk informasi selengkapnya tentang cara memulai dengan contoh aplikasi, lihatAplikasi sampel.

Java

```
   private static final DateTimeFormatter TIMESTAMP_FORMATTER =
DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss.SSSSSSSSS");
   private static final DateTimeFormatter DATE_FORMATTER =
DateTimeFormatter.ofPattern("yyyy-MM-dd");
   private static final DateTimeFormatter TIME_FORMATTER =
DateTimeFormatter.ofPattern("HH:mm:ss.SSSSSSSSS");

   private static final long ONE_GB_IN_BYTES = 1073741824L;

   private void parseQueryResult(QueryResult response) {
       final QueryStatus currentStatusOfQuery = queryResult.getQueryStatus();

       System.out.println("Query progress so far: " +
currentStatusOfQuery.getProgressPercentage() + "%");

       double bytesScannedSoFar = ((double)
currentStatusOfQuery.getCumulativeBytesScanned() / ONE_GB_IN_BYTES);
       System.out.println("Bytes scanned so far: " + bytesScannedSoFar + " GB");

       double bytesMeteredSoFar = ((double)
currentStatusOfQuery.getCumulativeBytesMetered() / ONE_GB_IN_BYTES);
       System.out.println("Bytes metered so far: " + bytesMeteredSoFar + " GB");

       List<ColumnInfo> columnInfo = response.getColumnInfo();
       List<Row> rows = response.getRows();

       System.out.println("Metadata: " + columnInfo);
       System.out.println("Data: ");

       // iterate every row
       for (Row row : rows) {
           System.out.println(parseRow(columnInfo, row));
       }
   }

   private String parseRow(List<ColumnInfo> columnInfo, Row row) {
       List<Datum> data = row.getData();
       List<String> rowOutput = new ArrayList<>();
       // iterate every column per row
       for (int j = 0; j < data.size(); j++) {
           ColumnInfo info = columnInfo.get(j);
           Datum datum = data.get(j);
```

```
            rowOutput.add(parseDatum(info, datum));
        }
        return String.format("{%s}",
rowOutput.stream().map(Object::toString).collect(Collectors.joining(",")));
    }

    private String parseDatum(ColumnInfo info, Datum datum) {
        if (datum.isNullValue() != null && datum.isNullValue()) {
            return info.getName() + "=" + "NULL";
        }
        Type columnType = info.getType();
        // If the column is of TimeSeries Type
        if (columnType.getTimeSeriesMeasureValueColumnInfo() != null) {
            return parseTimeSeries(info, datum);
        }
        // If the column is of Array Type
        else if (columnType.getArrayColumnInfo() != null) {
            List<Datum> arrayValues = datum.getArrayValue();
            return info.getName() + "=" +
parseArray(info.getType().getArrayColumnInfo(), arrayValues);
        }
        // If the column is of Row Type
        else if (columnType.getRowColumnInfo() != null) {
            List<ColumnInfo> rowColumnInfo = info.getType().getRowColumnInfo();
            Row rowValues = datum.getRowValue();
            return parseRow(rowColumnInfo, rowValues);
        }
        // If the column is of Scalar Type
        else {
            return parseScalarType(info, datum);
        }
    }

    private String parseTimeSeries(ColumnInfo info, Datum datum) {
        List<String> timeSeriesOutput = new ArrayList<>();
        for (TimeSeriesDataPoint dataPoint : datum.getTimeSeriesValue()) {
            timeSeriesOutput.add("{time=" + dataPoint.getTime() + ", value=" +
                    parseDatum(info.getType().getTimeSeriesMeasureValueColumnInfo(),
dataPoint.getValue()) + "}");
        }
        return String.format("[%s]",
timeSeriesOutput.stream().map(Object::toString).collect(Collectors.joining(",")));
    }
```

```
    private String parseScalarType(ColumnInfo info, Datum datum) {
        switch (ScalarType.fromValue(info.getType().getScalarType())) {
            case VARCHAR:
                return parseColumnName(info) + datum.getScalarValue();
            case BIGINT:
                Long longValue = Long.valueOf(datum.getScalarValue());
                return parseColumnName(info) + longValue;
            case INTEGER:
                Integer intValue = Integer.valueOf(datum.getScalarValue());
                return parseColumnName(info) + intValue;
            case BOOLEAN:
                Boolean booleanValue = Boolean.valueOf(datum.getScalarValue());
                return parseColumnName(info) + booleanValue;
            case DOUBLE:
                Double doubleValue = Double.valueOf(datum.getScalarValue());
                return parseColumnName(info) + doubleValue;
            case TIMESTAMP:
                return parseColumnName(info) +
LocalDateTime.parse(datum.getScalarValue(), TIMESTAMP_FORMATTER);
            case DATE:
                return parseColumnName(info) +
LocalDate.parse(datum.getScalarValue(), DATE_FORMATTER);
            case TIME:
                return parseColumnName(info) +
LocalTime.parse(datum.getScalarValue(), TIME_FORMATTER);
            case INTERVAL_DAY_TO_SECOND:
            case INTERVAL_YEAR_TO_MONTH:
                return parseColumnName(info) + datum.getScalarValue();
            case UNKNOWN:
                return parseColumnName(info) + datum.getScalarValue();
            default:
                throw new IllegalArgumentException("Given type is not valid: " +
info.getType().getScalarType());
        }
    }

    private String parseColumnName(ColumnInfo info) {
        return info.getName() == null ? "" : info.getName() + "=";
    }

    private String parseArray(ColumnInfo arrayColumnInfo, List<Datum> arrayValues) {
        List<String> arrayOutput = new ArrayList<>();
        for (Datum datum : arrayValues) {
            arrayOutput.add(parseDatum(arrayColumnInfo, datum));
```

```
        }
        return String.format("[%s]",
arrayOutput.stream().map(Object::toString).collect(Collectors.joining(",")));
    }
```

Java v2

```
    private static final long ONE_GB_IN_BYTES = 1073741824L;

    private void parseQueryResult(QueryResponse response) {
        final QueryStatus currentStatusOfQuery = response.queryStatus();

        System.out.println("Query progress so far: " +
currentStatusOfQuery.progressPercentage() + "%");

        double bytesScannedSoFar = ((double)
currentStatusOfQuery.cumulativeBytesScanned() / ONE_GB_IN_BYTES);
        System.out.println("Bytes scanned so far: " + bytesScannedSoFar + " GB");

        double bytesMeteredSoFar = ((double)
currentStatusOfQuery.cumulativeBytesMetered() / ONE_GB_IN_BYTES);
        System.out.println("Bytes metered so far: " + bytesMeteredSoFar + " GB");

        List<ColumnInfo> columnInfo = response.columnInfo();
        List<Row> rows = response.rows();

        System.out.println("Metadata: " + columnInfo);
        System.out.println("Data: ");

        // iterate every row
        for (Row row : rows) {
            System.out.println(parseRow(columnInfo, row));
        }
    }

    private String parseRow(List<ColumnInfo> columnInfo, Row row) {
        List<Datum> data = row.data();
        List<String> rowOutput = new ArrayList<>();
        // iterate every column per row
        for (int j = 0; j < data.size(); j++) {
            ColumnInfo info = columnInfo.get(j);
            Datum datum = data.get(j);
            rowOutput.add(parseDatum(info, datum));
```

```
        }
        return String.format("{%s}",
rowOutput.stream().map(Object::toString).collect(Collectors.joining(",")));
    }

    private String parseDatum(ColumnInfo info, Datum datum) {
        if (datum.nullValue() != null && datum.nullValue()) {
            return info.name() + "=" + "NULL";
        }
        Type columnType = info.type();
        // If the column is of TimeSeries Type
        if (columnType.timeSeriesMeasureValueColumnInfo() != null) {
            return parseTimeSeries(info, datum);
        }
        // If the column is of Array Type
        else if (columnType.arrayColumnInfo() != null) {
            List<Datum> arrayValues = datum.arrayValue();
            return info.name() + "=" + parseArray(info.type().arrayColumnInfo(),
arrayValues);
        }
        // If the column is of Row Type
        else if (columnType.rowColumnInfo() != null &&
columnType.rowColumnInfo().size() > 0) {
            List<ColumnInfo> rowColumnInfo = info.type().rowColumnInfo();
            Row rowValues = datum.rowValue();
            return parseRow(rowColumnInfo, rowValues);
        }
        // If the column is of Scalar Type
        else {
            return parseScalarType(info, datum);
        }
    }

    private String parseTimeSeries(ColumnInfo info, Datum datum) {
        List<String> timeSeriesOutput = new ArrayList<>();
        for (TimeSeriesDataPoint dataPoint : datum.timeSeriesValue()) {
            timeSeriesOutput.add("{time=" + dataPoint.time() + ", value=" +
                    parseDatum(info.type().timeSeriesMeasureValueColumnInfo(),
dataPoint.value()) + "}");
        }
        return String.format("[%s]",
timeSeriesOutput.stream().map(Object::toString).collect(Collectors.joining(",")));
    }
```

```
    private String parseScalarType(ColumnInfo info, Datum datum) {
        return parseColumnName(info) + datum.scalarValue();
    }

    private String parseColumnName(ColumnInfo info) {
        return info.name() == null ? "" : info.name() + "=";
    }

    private String parseArray(ColumnInfo arrayColumnInfo, List<Datum> arrayValues) {
        List<String> arrayOutput = new ArrayList<>();
        for (Datum datum : arrayValues) {
            arrayOutput.add(parseDatum(arrayColumnInfo, datum));
        }
        return String.format("[%s]",
 arrayOutput.stream().map(Object::toString).collect(Collectors.joining(",")));
    }
```

Go

```
func processScalarType(data *timestreamquery.Datum) string {
    return *data.ScalarValue
}

func processTimeSeriesType(data []*timestreamquery.TimeSeriesDataPoint, columnInfo
 *timestreamquery.ColumnInfo) string {
    value := ""
    for k := 0; k < len(data); k++ {
        time := data[k].Time
        value += *time + ":"
        if columnInfo.Type.ScalarType != nil {
            value += processScalarType(data[k].Value)
        } else if columnInfo.Type.ArrayColumnInfo != nil {
            value += processArrayType(data[k].Value.ArrayValue,
 columnInfo.Type.ArrayColumnInfo)
        } else if columnInfo.Type.RowColumnInfo != nil {
            value += processRowType(data[k].Value.RowValue.Data,
 columnInfo.Type.RowColumnInfo)
        } else {
            fail("Bad data type")
        }
        if k != len(data)-1 {
            value += ", "
        }
```

```
    }
    return value
}

func processArrayType(datumList []*timestreamquery.Datum, columnInfo
 *timestreamquery.ColumnInfo) string {
    value := ""
    for k := 0; k < len(datumList); k++ {
        if columnInfo.Type.ScalarType != nil {
            value += processScalarType(datumList[k])
        } else if columnInfo.Type.TimeSeriesMeasureValueColumnInfo != nil {
            value += processTimeSeriesType(datumList[k].TimeSeriesValue,
 columnInfo.Type.TimeSeriesMeasureValueColumnInfo)
        } else if columnInfo.Type.ArrayColumnInfo != nil {
            value += "["
            value += processArrayType(datumList[k].ArrayValue,
 columnInfo.Type.ArrayColumnInfo)
            value += "]"
        } else if columnInfo.Type.RowColumnInfo != nil {
            value += "["
            value += processRowType(datumList[k].RowValue.Data,
 columnInfo.Type.RowColumnInfo)
            value += "]"
        } else {
            fail("Bad column type")
        }

        if k != len(datumList)-1 {
            value += ", "
        }
    }
    return value
}

func processRowType(data []*timestreamquery.Datum, metadata
 []*timestreamquery.ColumnInfo) string {
    value := ""
    for j := 0; j < len(data); j++ {
        if metadata[j].Type.ScalarType != nil {
            // process simple data types
            value += processScalarType(data[j])
        } else if metadata[j].Type.TimeSeriesMeasureValueColumnInfo != nil {
            // fmt.Println("Timeseries measure value column info")
            // fmt.Println(metadata[j].Type.TimeSeriesMeasureValueColumnInfo.Type)
```

```
            datapointList := data[j].TimeSeriesValue
            value += "["
            value += processTimeSeriesType(datapointList,
 metadata[j].Type.TimeSeriesMeasureValueColumnInfo)
            value += "]"
        } else if metadata[j].Type.ArrayColumnInfo != nil {
            columnInfo := metadata[j].Type.ArrayColumnInfo
            // fmt.Println("Array column info")
            // fmt.Println(columnInfo)
            datumList := data[j].ArrayValue
            value += "["
            value += processArrayType(datumList, columnInfo)
            value += "]"
        } else if metadata[j].Type.RowColumnInfo != nil {
            columnInfo := metadata[j].Type.RowColumnInfo
            datumList := data[j].RowValue.Data
            value += "["
            value += processRowType(datumList, columnInfo)
            value += "]"
        } else {
            panic("Bad column type")
        }
        // comma seperated column values
        if j != len(data)-1 {
            value += ", "
        }
    }
    return value
}
```

Python

```
    def _parse_query_result(self, query_result):
        query_status = query_result["QueryStatus"]

        progress_percentage = query_status["ProgressPercentage"]
        print(f"Query progress so far: {progress_percentage}%")

        bytes_scanned = float(query_status["CumulativeBytesScanned"]) /
 ONE_GB_IN_BYTES
        print(f"Data scanned so far: {bytes_scanned} GB")
```

```
        bytes_metered = float(query_status["CumulativeBytesMetered"]) /
 ONE_GB_IN_BYTES
        print(f"Data metered so far: {bytes_metered} GB")

        column_info = query_result['ColumnInfo']

        print("Metadata: %s" % column_info)
        print("Data: ")
        for row in query_result['Rows']:
            print(self._parse_row(column_info, row))

    def _parse_row(self, column_info, row):
        data = row['Data']
        row_output = []
        for j in range(len(data)):
            info = column_info[j]
            datum = data[j]
            row_output.append(self._parse_datum(info, datum))

        return "{%s}" % str(row_output)

    def _parse_datum(self, info, datum):
        if datum.get('NullValue', False):
            return "%s=NULL" % info['Name'],

        column_type = info['Type']

        # If the column is of TimeSeries Type
        if 'TimeSeriesMeasureValueColumnInfo' in column_type:
            return self._parse_time_series(info, datum)

        # If the column is of Array Type
        elif 'ArrayColumnInfo' in column_type:
            array_values = datum['ArrayValue']
            return "%s=%s" % (info['Name'], self._parse_array(info['Type']
['ArrayColumnInfo'], array_values))

        # If the column is of Row Type
        elif 'RowColumnInfo' in column_type:
            row_column_info = info['Type']['RowColumnInfo']
            row_values = datum['RowValue']
            return self._parse_row(row_column_info, row_values)

        # If the column is of Scalar Type
```

```
        else:
            return self._parse_column_name(info) + datum['ScalarValue']

    def _parse_time_series(self, info, datum):
        time_series_output = []
        for data_point in datum['TimeSeriesValue']:
            time_series_output.append("{time=%s, value=%s}"
                                      % (data_point['Time'],
                                         self._parse_datum(info['Type']
['TimeSeriesMeasureValueColumnInfo'],
                                                           data_point['Value'])))
        return "[%s]" % str(time_series_output)

    def _parse_array(self, array_column_info, array_values):
        array_output = []
        for datum in array_values:
            array_output.append(self._parse_datum(array_column_info, datum))

        return "[%s]" % str(array_output)

    @staticmethod
    def _parse_column_name(info):
        if 'Name' in info:
            return info['Name'] + "="
        else:
            return ""
```

Node.js

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada aplikasi sampel Node.js Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
function parseQueryResult(response) {
    const queryStatus = response.QueryStatus;
    console.log("Current query status: " + JSON.stringify(queryStatus));

    const columnInfo = response.ColumnInfo;
    const rows = response.Rows;

    console.log("Metadata: " + JSON.stringify(columnInfo));
    console.log("Data: ");

    rows.forEach(function (row) {
```

```
        console.log(parseRow(columnInfo, row));
    });
}

function parseRow(columnInfo, row) {
    const data = row.Data;
    const rowOutput = [];

    var i;
    for ( i = 0; i < data.length; i++ ) {
        info = columnInfo[i];
        datum = data[i];
        rowOutput.push(parseDatum(info, datum));
    }

    return `{${rowOutput.join(", ")}}`
}

function parseDatum(info, datum) {
    if (datum.NullValue != null && datum.NullValue === true) {
        return `${info.Name}=NULL`;
    }

    const columnType = info.Type;

    // If the column is of TimeSeries Type
    if (columnType.TimeSeriesMeasureValueColumnInfo != null) {
        return parseTimeSeries(info, datum);
    }
    // If the column is of Array Type
    else if (columnType.ArrayColumnInfo != null) {
        const arrayValues = datum.ArrayValue;
        return `${info.Name}=${parseArray(info.Type.ArrayColumnInfo, arrayValues)}`;
    }
    // If the column is of Row Type
    else if (columnType.RowColumnInfo != null) {
        const rowColumnInfo = info.Type.RowColumnInfo;
        const rowValues = datum.RowValue;
        return parseRow(rowColumnInfo, rowValues);
    }
    // If the column is of Scalar Type
    else {
        return parseScalarType(info, datum);
    }
```

```
}

function parseTimeSeries(info, datum) {
    const timeSeriesOutput = [];
    datum.TimeSeriesValue.forEach(function (dataPoint) {
        timeSeriesOutput.push(`{time=${dataPoint.Time}, value=
${parseDatum(info.Type.TimeSeriesMeasureValueColumnInfo, dataPoint.Value)}}`)
    });

    return `[${timeSeriesOutput.join(", ")}]`
}

function parseScalarType(info, datum) {
    return parseColumnName(info) + datum.ScalarValue;
}

function parseColumnName(info) {
    return info.Name == null ? "" : `${info.Name}=`;
}

function parseArray(arrayColumnInfo, arrayValues) {
    const arrayOutput = [];
    arrayValues.forEach(function (datum) {
        arrayOutput.push(parseDatum(arrayColumnInfo, datum));
    });
    return `[${arrayOutput.join(", ")}]`
}
```

.NET

```
        private void ParseQueryResult(QueryResponse response)
        {
            List<ColumnInfo> columnInfo = response.ColumnInfo;
            var options = new JsonSerializerOptions
            {
                IgnoreNullValues = true
            };
            List<String> columnInfoStrings = columnInfo.ConvertAll(x =>
 JsonSerializer.Serialize(x, options));
            List<Row> rows = response.Rows;

            QueryStatus queryStatus = response.QueryStatus;
```

```
            Console.WriteLine("Current Query status:" +
JsonSerializer.Serialize(queryStatus, options));

            Console.WriteLine("Metadata:" + string.Join(",", columnInfoStrings));
            Console.WriteLine("Data:");

            foreach (Row row in rows)
            {
                Console.WriteLine(ParseRow(columnInfo, row));
            }
        }

        private string ParseRow(List<ColumnInfo> columnInfo, Row row)
        {
            List<Datum> data = row.Data;
            List<string> rowOutput = new List<string>();
            for (int j = 0; j < data.Count; j++)
            {
                ColumnInfo info = columnInfo[j];
                Datum datum = data[j];
                rowOutput.Add(ParseDatum(info, datum));
            }
            return $"{{{string.Join(",", rowOutput)}}}";
        }

        private string ParseDatum(ColumnInfo info, Datum datum)
        {
            if (datum.NullValue)
            {
                return $"{info.Name}=NULL";
            }

            Amazon.TimestreamQuery.Model.Type columnType = info.Type;
            if (columnType.TimeSeriesMeasureValueColumnInfo != null)
            {
                return ParseTimeSeries(info, datum);
            }
            else if (columnType.ArrayColumnInfo != null)
            {
                List<Datum> arrayValues = datum.ArrayValue;
                return $"{info.Name}={ParseArray(info.Type.ArrayColumnInfo,
arrayValues)}";
            }
```

```
            else if (columnType.RowColumnInfo != null &&
columnType.RowColumnInfo.Count > 0)
            {
                List<ColumnInfo> rowColumnInfo = info.Type.RowColumnInfo;
                Row rowValue = datum.RowValue;
                return ParseRow(rowColumnInfo, rowValue);
            }
            else
            {
                return ParseScalarType(info, datum);
            }
        }

        private string ParseTimeSeries(ColumnInfo info, Datum datum)
        {
            var timeseriesString = datum.TimeSeriesValue
                .Select(value => $"{{time={value.Time},
value={ParseDatum(info.Type.TimeSeriesMeasureValueColumnInfo, value.Value)}}}")
                .Aggregate((current, next) => current + "," + next);

            return $"[{timeseriesString}]";
        }

        private string ParseScalarType(ColumnInfo info, Datum datum)
        {
            return ParseColumnName(info) + datum.ScalarValue;
        }

        private string ParseColumnName(ColumnInfo info)
        {
            return info.Name == null ? "" : (info.Name + "=");
        }

        private string ParseArray(ColumnInfo arrayColumnInfo, List<Datum>
arrayValues)
        {
            return $"[{arrayValues.Select(value => ParseDatum(arrayColumnInfo,
value)).Aggregate((current, next) => current + "," + next)}]";
        }
```

Mengakses status kueri

Anda dapat mengakses status kueri melaluiQueryResponse, yang berisi informasi tentang kemajuan kueri, byte yang dipindai oleh kueri, dan byte yang diukur oleh kueri. bytesScannedNilai bytesMetered dan bersifat kumulatif dan terus diperbarui saat melakukan paging hasil kueri. Anda dapat menggunakan informasi ini untuk memahami byte yang dipindai oleh kueri individual dan juga menggunakannya untuk membuat keputusan tertentu. Misalnya, dengan asumsi bahwa harga kueri adalah $0,01 per GB yang dipindai, Anda mungkin ingin membatalkan kueri yang melebihi $25 per kueri, atau GB. X Cuplikan kode di bawah ini menunjukkan bagaimana hal ini dapat dilakukan.

> **Note**
>
> Cuplikan kode ini didasarkan pada contoh aplikasi lengkap. GitHub Untuk informasi selengkapnya tentang cara memulai dengan contoh aplikasi, lihatAplikasi sampel.

Java

```
   private static final long ONE_GB_IN_BYTES = 1073741824L;
   private static final double QUERY_COST_PER_GB_IN_DOLLARS = 0.01; // Assuming the
price of query is $0.01 per GB

   public void cancelQueryBasedOnQueryStatus() {
       System.out.println("Starting query: " + SELECT_ALL_QUERY);
       QueryRequest queryRequest = new QueryRequest();
       queryRequest.setQueryString(SELECT_ALL_QUERY);
       QueryResult queryResult = queryClient.query(queryRequest);

       while (true) {
           final QueryStatus currentStatusOfQuery = queryResult.getQueryStatus();
           System.out.println("Query progress so far: " +
currentStatusOfQuery.getProgressPercentage() + "%");
           double bytesMeteredSoFar = ((double)
currentStatusOfQuery.getCumulativeBytesMetered() / ONE_GB_IN_BYTES);
           System.out.println("Bytes metered so far: " + bytesMeteredSoFar + "
GB");
           // Cancel query if its costing more than 1 cent
           if (bytesMeteredSoFar * QUERY_COST_PER_GB_IN_DOLLARS > 0.01) {
               cancelQuery(queryResult);
               break;
           }
```

```
            if (queryResult.getNextToken() == null) {
                break;
            }
            queryRequest.setNextToken(queryResult.getNextToken());
            queryResult = queryClient.query(queryRequest);
        }
    }
```

Java v2

```
    private static final long ONE_GB_IN_BYTES = 1073741824L;
    private static final double QUERY_COST_PER_GB_IN_DOLLARS = 0.01; // Assuming the
price of query is $0.01 per GB

    public void cancelQueryBasedOnQueryStatus() {
        System.out.println("Starting query: " + SELECT_ALL_QUERY);
        QueryRequest queryRequest =
QueryRequest.builder().queryString(SELECT_ALL_QUERY).build();

        final QueryIterable queryResponseIterator =
timestreamQueryClient.queryPaginator(queryRequest);
        for(QueryResponse queryResponse : queryResponseIterator) {
            final QueryStatus currentStatusOfQuery = queryResponse.queryStatus();
            System.out.println("Query progress so far: " +
currentStatusOfQuery.progressPercentage() + "%");
            double bytesMeteredSoFar = ((double)
currentStatusOfQuery.cumulativeBytesMetered() / ONE_GB_IN_BYTES);
            System.out.println("Bytes metered so far: " + bytesMeteredSoFar + "GB");
            // Cancel query if its costing more than 1 cent
            if (bytesMeteredSoFar * QUERY_COST_PER_GB_IN_DOLLARS > 0.01) {
                cancelQuery(queryResponse);
                break;
            }
        }
    }
```

Go

```
const OneGbInBytes = 1073741824
// Assuming the price of query is $0.01 per GB
const QueryCostPerGbInDollars = 0.01
```

```
func cancelQueryBasedOnQueryStatus(queryPtr *string, querySvc
 *timestreamquery.TimestreamQuery, f *os.File) {
    queryInput := &timestreamquery.QueryInput{
        QueryString: aws.String(*queryPtr),
    }
    fmt.Println("QueryInput:")
    fmt.Println(queryInput)
    // execute the query
    err := querySvc.QueryPages(queryInput,
        func(page *timestreamquery.QueryOutput, lastPage bool) bool {
            // process query response
            queryStatus := page.QueryStatus
            fmt.Println("Current query status:", queryStatus)
            bytes_metered := float64(*queryStatus.CumulativeBytesMetered) /
 float64(ONE_GB_IN_BYTES)
            if bytes_metered * QUERY_COST_PER_GB_IN_DOLLARS > 0.01 {
                cancelQuery(page, querySvc)
                return true
            }
            // query response metadata
            // includes column names and types
            metadata := page.ColumnInfo
            // fmt.Println("Metadata:")
            fmt.Println(metadata)
            header := ""
            for i := 0; i < len(metadata); i++ {
                header += *metadata[i].Name
                if i != len(metadata)-1 {
                    header += ", "
                }
            }
            write(f, header)

            // query response data
            fmt.Println("Data:")
            // process rows
            rows := page.Rows
            for i := 0; i < len(rows); i++ {
                data := rows[i].Data
                value := processRowType(data, metadata)
                fmt.Println(value)
                write(f, value)
            }
            fmt.Println("Number of rows:", len(page.Rows))
```

```
            return true
        })
    if err != nil {
        fmt.Println("Error:")
        fmt.Println(err)
    }
}
```

Python

```
ONE_GB_IN_BYTES = 1073741824
# Assuming the price of query is $0.01 per GB
QUERY_COST_PER_GB_IN_DOLLARS = 0.01

    def cancel_query_based_on_query_status(self):
        try:
            print("Starting query: " + self.SELECT_ALL)
            page_iterator = self.paginator.paginate(QueryString=self.SELECT_ALL)
            for page in page_iterator:
                query_status = page["QueryStatus"]
                progress_percentage = query_status["ProgressPercentage"]
                print("Query progress so far: " + str(progress_percentage) + "%")
                bytes_metered = query_status["CumulativeBytesMetered"] /
 self.ONE_GB_IN_BYTES
                print("Bytes Metered so far: " + str(bytes_metered) + " GB")
                if bytes_metered * self.QUERY_COST_PER_GB_IN_DOLLARS > 0.01:
                    self.cancel_query_for(page)
                    break
        except Exception as err:
            print("Exception while running query:", err)
            traceback.print_exc(file=sys.stderr)
```

Node.js

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada aplikasi sampel Node.js Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
function parseQueryResult(response) {
    const queryStatus = response.QueryStatus;
    console.log("Current query status: " + JSON.stringify(queryStatus));

    const columnInfo = response.ColumnInfo;
    const rows = response.Rows;
```

```
    console.log("Metadata: " + JSON.stringify(columnInfo));
    console.log("Data: ");

    rows.forEach(function (row) {
        console.log(parseRow(columnInfo, row));
    });
}

function parseRow(columnInfo, row) {
    const data = row.Data;
    const rowOutput = [];

    var i;
    for ( i = 0; i < data.length; i++ ) {
        info = columnInfo[i];
        datum = data[i];
        rowOutput.push(parseDatum(info, datum));
    }

    return `{${rowOutput.join(", ")}}`
}

function parseDatum(info, datum) {
    if (datum.NullValue != null && datum.NullValue === true) {
        return `${info.Name}=NULL`;
    }

    const columnType = info.Type;

    // If the column is of TimeSeries Type
    if (columnType.TimeSeriesMeasureValueColumnInfo != null) {
        return parseTimeSeries(info, datum);
    }
    // If the column is of Array Type
    else if (columnType.ArrayColumnInfo != null) {
        const arrayValues = datum.ArrayValue;
        return `${info.Name}=${parseArray(info.Type.ArrayColumnInfo, arrayValues)}`;
    }
    // If the column is of Row Type
    else if (columnType.RowColumnInfo != null) {
        const rowColumnInfo = info.Type.RowColumnInfo;
        const rowValues = datum.RowValue;
        return parseRow(rowColumnInfo, rowValues);
```

```
    }
    // If the column is of Scalar Type
    else {
        return parseScalarType(info, datum);
    }
}

function parseTimeSeries(info, datum) {
    const timeSeriesOutput = [];
    datum.TimeSeriesValue.forEach(function (dataPoint) {
        timeSeriesOutput.push(`{time=${dataPoint.Time}, value=
${parseDatum(info.Type.TimeSeriesMeasureValueColumnInfo, dataPoint.Value)}}`)
    });

    return `[${timeSeriesOutput.join(", ")}]`
}

function parseScalarType(info, datum) {
    return parseColumnName(info) + datum.ScalarValue;
}

function parseColumnName(info) {
    return info.Name == null ? "" : `${info.Name}=`;
}

function parseArray(arrayColumnInfo, arrayValues) {
    const arrayOutput = [];
    arrayValues.forEach(function (datum) {
        arrayOutput.push(parseDatum(arrayColumnInfo, datum));
    });
    return `[${arrayOutput.join(", ")}]`
}
```

.NET

```
private static readonly long ONE_GB_IN_BYTES = 1073741824L;
private static readonly double QUERY_COST_PER_GB_IN_DOLLARS = 0.01; // Assuming the
 price of query is $0.01 per GB

private async Task CancelQueryBasedOnQueryStatus(string queryString)
{
    try
    {
```

```
        QueryRequest queryRequest = new QueryRequest();
        queryRequest.QueryString = queryString;
        QueryResponse queryResponse = await queryClient.QueryAsync(queryRequest);
        while (true)
        {
            QueryStatus queryStatus = queryResponse.QueryStatus;
            double bytesMeteredSoFar = ((double)
 queryStatus.CumulativeBytesMetered / ONE_GB_IN_BYTES);
            // Cancel query if its costing more than 1 cent
            if (bytesMeteredSoFar * QUERY_COST_PER_GB_IN_DOLLARS > 0.01)
            {
                await CancelQuery(queryResponse);
                break;
            }

            ParseQueryResult(queryResponse);
            if (queryResponse.NextToken == null)
            {
                break;
            }
            queryRequest.NextToken = queryResponse.NextToken;
            queryResponse = await queryClient.QueryAsync(queryRequest);
       }
    } catch(Exception e)
    {
        // Some queries might fail with 500 if the result of a sequence function has
 more than 10000 entries
        Console.WriteLine(e.ToString());
    }
}
```

Untuk detail tambahan tentang cara membatalkan kueri, lihatBatalkan kueri.

Jalankan UNLOAD kueri

Contoh kode berikut memanggil UNLOAD kueri. Untuk informasi tentang UNLOAD, lihat Menggunakan UNLOAD untuk mengekspor hasil kueri ke S3 dari Timestream untuk LiveAnalytics. Untuk contoh UNLOAD kueri, lihatContoh kasus penggunaan untuk UNLOAD dari Timestream untuk LiveAnalytics.

Topik

- Membangun dan menjalankan UNLOAD kueri

- Mengurai UNLOAD respons, dan dapatkan jumlah baris, tautan manifes, dan tautan metadata
- Membaca dan mengurai konten manifes
- Membaca dan mengurai konten metadata

Membangun dan menjalankan UNLOAD kueri

Java

```
// When you have a SELECT like below

String QUERY_1 = "SELECT user_id, ip_address, event, session_id, measure_name, time,
 query, quantity, product_id, channel FROM "
        + DATABASE_NAME + "." + UNLOAD_TABLE_NAME
        + " WHERE time BETWEEN ago(2d) AND now()";

// You can construct UNLOAD query as follows
UnloadQuery unloadQuery = UnloadQuery.builder()
        .selectQuery(QUERY_1)
        .bucketName("timestream-sample-<region>-<accountId>")
        .resultsPrefix("without_partition")
        .format(CSV)
        .compression(UnloadQuery.Compression.GZIP)
        .build();
QueryResult unloadResult = runQuery(unloadQuery.getUnloadQuery());

// Run UNLOAD query (Similar to how you run SELECT query)
// https://docs.aws.amazon.com/timestream/latest/developerguide/code-samples.run-
query.html#code-samples.run-query.pagination
    private QueryResult runQuery(String queryString) {
        QueryResult queryResult = null;
        try {
            QueryRequest queryRequest = new QueryRequest();
            queryRequest.setQueryString(queryString);
            queryResult = queryClient.query(queryRequest);
            while (true) {
                parseQueryResult(queryResult);
                if (queryResult.getNextToken() == null) {
                    break;
                }
                queryRequest.setNextToken(queryResult.getNextToken());
                queryResult = queryClient.query(queryRequest);
            }
```

```
        } catch (Exception e) {
            // Some queries might fail with 500 if the result of a sequence function
 has more than 10000 entries
            e.printStackTrace();
        }
        return queryResult;
    }

// Utility that helps to construct UNLOAD query

@Builder
static class UnloadQuery {
    private String selectQuery;
    private String bucketName;
    private String resultsPrefix;
    private Format format;
    private Compression compression;
    private EncryptionType encryptionType;
    private List<String> partitionColumns;
    private String kmsKey;
    private Character csvFieldDelimiter;
    private Character csvEscapeCharacter;

    public String getUnloadQuery() {
        String destination = constructDestination();
        String withClause = constructOptionalParameters();
        return String.format("UNLOAD (%s) TO '%s' %s", selectQuery, destination,
 withClause);
    }

    private String constructDestination() {
        return "s3://" + this.bucketName + "/" + this.resultsPrefix + "/";
    }

    private String constructOptionalParameters() {
        boolean isOptionalParametersPresent = Objects.nonNull(format)
                || Objects.nonNull(compression)
                || Objects.nonNull(encryptionType)
                || Objects.nonNull(partitionColumns)
                || Objects.nonNull(kmsKey)
                || Objects.nonNull(csvFieldDelimiter)
                || Objects.nonNull(csvEscapeCharacter);

        String withClause = "";
```

```
        if (isOptionalParametersPresent) {
            StringJoiner optionalParameters = new StringJoiner(",");
            if (Objects.nonNull(format)) {
                optionalParameters.add("format = '" + format + "'");
            }
            if (Objects.nonNull(compression)) {
                optionalParameters.add("compression = '" + compression + "'");
            }
            if (Objects.nonNull(encryptionType)) {
                optionalParameters.add("encryption = '" + encryptionType + "'");
            }
            if (Objects.nonNull(kmsKey)) {
                optionalParameters.add("kms_key = '" + kmsKey + "'");
            }
            if (Objects.nonNull(csvFieldDelimiter)) {
                optionalParameters.add("field_delimiter = '" + csvFieldDelimiter +
"'");
            }
            if (Objects.nonNull(csvEscapeCharacter)) {
                optionalParameters.add("escaped_by = '" + csvEscapeCharacter + "'");
            }
            if (Objects.nonNull(partitionColumns) && !partitionColumns.isEmpty()) {
                final StringJoiner partitionedByList = new StringJoiner(",");
                partitionColumns.forEach(column -> partitionedByList.add("'" +
column + "'"));
                optionalParameters.add(String.format("partitioned_by = ARRAY[%s]",
partitionedByList));
            }
            withClause = String.format("WITH (%s)", optionalParameters);
        }
        return withClause;
    }

    public enum Format {
        CSV, PARQUET
    }

    public enum Compression {
        GZIP, NONE
    }

    public enum EncryptionType {
        SSE_S3, SSE_KMS
    }
```

```
    @Override
    public String toString() {
        return getUnloadQuery();
    }
}
```

Java v2

```
// When you have a SELECT like below

String QUERY_1 = "SELECT user_id, ip_address, event, session_id, measure_name, time,
 query, quantity, product_id, channel FROM "
        + DATABASE_NAME + "." + UNLOAD_TABLE_NAME
        + " WHERE time BETWEEN ago(2d) AND now()";

//You can construct UNLOAD query as follows
UnloadQuery unloadQuery = UnloadQuery.builder()
        .selectQuery(QUERY_1)
        .bucketName("timestream-sample-<region>-<accountId>")
        .resultsPrefix("without_partition")
        .format(CSV)
        .compression(UnloadQuery.Compression.GZIP)
        .build();

QueryResponse unloadResponse = runQuery(unloadQuery.getUnloadQuery());


// Run UNLOAD query (Similar to how you run SELECT query)
// https://docs.aws.amazon.com/timestream/latest/developerguide/code-samples.run-
query.html#code-samples.run-query.pagination
private QueryResponse runQuery(String queryString) {
   QueryResponse finalResponse = null;
    try {
        QueryRequest queryRequest =
 QueryRequest.builder().queryString(queryString).build();
        final QueryIterable queryResponseIterator =
 timestreamQueryClient.queryPaginator(queryRequest);
        for(QueryResponse queryResponse : queryResponseIterator) {
            parseQueryResult(queryResponse);
           finalResponse = queryResponse;
        }
    } catch (Exception e) {
```

```
        // Some queries might fail with 500 if the result of a sequence function has
 more than 10000 entries
        e.printStackTrace();
    }
   return finalResponse;
}

// Utility that helps to construct UNLOAD query
@Builder
static class UnloadQuery {
    private String selectQuery;
    private String bucketName;
    private String resultsPrefix;
    private Format format;
    private Compression compression;
    private EncryptionType encryptionType;
    private List<String> partitionColumns;
    private String kmsKey;
    private Character csvFieldDelimiter;
    private Character csvEscapeCharacter;

    public String getUnloadQuery() {
        String destination = constructDestination();
        String withClause = constructOptionalParameters();
        return String.format("UNLOAD (%s) TO '%s' %s", selectQuery, destination,
 withClause);
    }

    private String constructDestination() {
        return "s3://" + this.bucketName + "/" + this.resultsPrefix + "/";
    }

    private String constructOptionalParameters() {
        boolean isOptionalParametersPresent = Objects.nonNull(format)
                || Objects.nonNull(compression)
                || Objects.nonNull(encryptionType)
                || Objects.nonNull(partitionColumns)
                || Objects.nonNull(kmsKey)
                || Objects.nonNull(csvFieldDelimiter)
                || Objects.nonNull(csvEscapeCharacter);

        String withClause = "";
        if (isOptionalParametersPresent) {
            StringJoiner optionalParameters = new StringJoiner(",");
```

```
            if (Objects.nonNull(format)) {
                optionalParameters.add("format = '" + format + "'");
            }
            if (Objects.nonNull(compression)) {
                optionalParameters.add("compression = '" + compression + "'");
            }
            if (Objects.nonNull(encryptionType)) {
                optionalParameters.add("encryption = '" + encryptionType + "'");
            }
            if (Objects.nonNull(kmsKey)) {
                optionalParameters.add("kms_key = '" + kmsKey + "'");
            }
            if (Objects.nonNull(csvFieldDelimiter)) {
                optionalParameters.add("field_delimiter = '" + csvFieldDelimiter +
"'");
            }
            if (Objects.nonNull(csvEscapeCharacter)) {
                optionalParameters.add("escaped_by = '" + csvEscapeCharacter + "'");
            }
            if (Objects.nonNull(partitionColumns) && !partitionColumns.isEmpty()) {
                final StringJoiner partitionedByList = new StringJoiner(",");
                partitionColumns.forEach(column -> partitionedByList.add("'" +
column + "'"));
                optionalParameters.add(String.format("partitioned_by = ARRAY[%s]",
partitionedByList));
            }
            withClause = String.format("WITH (%s)", optionalParameters);
        }
        return withClause;
    }

    public enum Format {
        CSV, PARQUET
    }

    public enum Compression {
        GZIP, NONE
    }

    public enum EncryptionType {
        SSE_S3, SSE_KMS
    }

    @Override
```

```
    public String toString() {
        return getUnloadQuery();
    }
}
```

Go

```
// When you have a SELECT like below
var Query = "SELECT user_id, ip_address, event, session_id, measure_name, time,
 query, quantity, product_id, channel FROM "
+ *databaseName + "." + *tableName + " WHERE time BETWEEN ago(2d) AND now()"

// You can construct UNLOAD query as follows
var unloadQuery = UnloadQuery{
    Query: "SELECT user_id, ip_address, session_id, measure_name, time, query,
 quantity, product_id, channel, event FROM " + *databaseName + "." + *tableName +
    " WHERE time BETWEEN ago(2d) AND now()",
    Partitioned_by: []string{},
    Compression: "GZIP",
    Format: "CSV",
    S3Location: bucketName,
    ResultPrefix: "without_partition",
}


// Run UNLOAD query (Similar to how you run SELECT query)
// https://docs.aws.amazon.com/timestream/latest/developerguide/code-samples.run-
query.html#code-samples.run-query.pagination

queryInput := &timestreamquery.QueryInput{
    QueryString: build_query(unloadQuery),
}

err := querySvc.QueryPages(queryInput,
    func(page *timestreamquery.QueryOutput, lastPage bool) bool {
        if (lastPage) {
            var response = parseQueryResult(page)
            var unloadFiles = getManifestAndMetadataFiles(s3Svc, response)
            displayColumns(unloadFiles, unloadQuery.Partitioned_by)
            displayResults(s3Svc, unloadFiles)
        }
        return true
    })
```

```
if err != nil {
    fmt.Println("Error:")
    fmt.Println(err)
}

// Utility that helps to construct UNLOAD query
type UnloadQuery struct {
    Query string
    Partitioned_by []string
    Format string
    S3Location string
    ResultPrefix string
    Compression string
}

func build_query(unload_query UnloadQuery) *string {
    var query_results_s3_path = "'s3://" + unload_query.S3Location + "/" +
 unload_query.ResultPrefix + "/'"
    var query = "UNLOAD(" + unload_query.Query + ") TO " + query_results_s3_path + "
 WITH ( "
    if (len(unload_query.Partitioned_by) > 0) {
        query = query + "partitioned_by=ARRAY["
        for i, column := range unload_query.Partitioned_by {
            if i == 0 {
                query = query + "'" + column + "'"
            } else {
                query = query + ",'" + column + "'"
            }
        }
        query = query + "],"
    }
    query = query + " format='" + unload_query.Format + "', "
    query = query + "  compression='" + unload_query.Compression + "')"
    fmt.Println(query)
    return aws.String(query)
}
```

Python

```
# When you have a SELECT like below
QUERY_1 = "SELECT user_id, ip_address, event, session_id, measure_name, time, query,
 quantity, product_id, channel FROM "
```

```
        + database_name + "." + table_name + " WHERE time BETWEEN ago(2d) AND now()"
# You can construct UNLOAD query as follows
UNLOAD_QUERY_1 = UnloadQuery(QUERY_1, "timestream-sample-<region>-<accountId>",
 "without_partition", "CSV", "GZIP", "")

# Run UNLOAD query (Similar to how you run SELECT query)
# https://docs.aws.amazon.com/timestream/latest/developerguide/code-samples.run-
query.html#code-samples.run-query.pagination
def run_query(self, query_string):
    try:
        page_iterator = self.paginator.paginate(QueryString=UNLOAD_QUERY_1)
    except Exception as err:
        print("Exception while running query:", err)

# Utility that helps to construct UNLOAD query
class UnloadQuery:
    def __init__(self, query, s3_bucket_location, results_prefix, format,
 compression , partition_by):
        self.query = query
        self.s3_bucket_location = s3_bucket_location
        self.results_prefix = results_prefix
        self.format = format
        self.compression = compression
        self.partition_by = partition_by

    def build_query(self):
        query_results_s3_path = "'s3://" + self.s3_bucket_location + "/" +
 self.results_prefix + "/'"
        unload_query = "UNLOAD("
        unload_query = unload_query + self.query
        unload_query = unload_query + ") "
        unload_query = unload_query + " TO " + query_results_s3_path
        unload_query = unload_query + " WITH ( "

        if(len(self.partition_by) > 0) :
            unload_query = unload_query + " partitioned_by = ARRAY" +
 str(self.partition_by) + ","

        unload_query = unload_query + " format='" + self.format  + "', "
        unload_query = unload_query + "  compression='" + self.compression + "')"

        return unload_query
```

Node.js

```
// When you have a SELECT like below
QUERY_1 = "SELECT user_id, ip_address, event, session_id, measure_name, time, query,
 quantity, product_id, channel FROM "
        + database_name + "." + table_name + " WHERE time BETWEEN ago(2d) AND now()"
// You can construct UNLOAD query as follows
UNLOAD_QUERY_1 = new UnloadQuery(QUERY_1, "timestream-sample-<region>-<accountId>",
 "without_partition", "CSV", "GZIP", "")


// Run UNLOAD query (Similar to how you run SELECT query)
// https://docs.aws.amazon.com/timestream/latest/developerguide/code-samples.run-
query.html#code-samples.run-query.pagination

async runQuery(query = UNLOAD_QUERY_1, nextToken) {
    const params = new QueryCommand({
        QueryString: query
    });

    if (nextToken) {
        params.NextToken = nextToken;
    }

    await queryClient.send(params).then(
            (response) => {
                if (response.NextToken) {
                    runQuery(queryClient, query, response.NextToken);
                } else {
                    await parseAndDisplayResults(response);
                }
            },
            (err) => {
                console.error("Error while querying:", err);
            });
}


class UnloadQuery {
    constructor(query, s3_bucket_location, results_prefix, format, compression ,
 partition_by) {
        this.query = query;
        this.s3_bucket_location = s3_bucket_location
        this.results_prefix = results_prefix
```

```
        this.format = format
        this.compression = compression
        this.partition_by = partition_by
    }

    buildQuery() {
        const query_results_s3_path = "'s3://" + this.s3_bucket_location + "/" +
 this.results_prefix + "/'"
        let unload_query = "UNLOAD("
        unload_query = unload_query + this.query
        unload_query = unload_query + ") "
        unload_query = unload_query + " TO " + query_results_s3_path
        unload_query = unload_query + " WITH ( "

        if(this.partition_by.length > 0) {
            let partitionBy = ""
            this.partition_by.forEach((str, i) => {
                partitionBy = partitionBy + (i ? ",'" : "'") + str + "'"
            })
            unload_query = unload_query + " partitioned_by = ARRAY[" + partitionBy +
 "],"
        }
        unload_query = unload_query + " format='" + this.format  + "', "
        unload_query = unload_query + "  compression='" + this.compression + "')"

        return unload_query
    }
}
```

Mengurai UNLOAD respons, dan dapatkan jumlah baris, tautan manifes, dan tautan metadata

Java

```
// Parsing UNLOAD query response is similar to how you parse SELECT query response:
// https://docs.aws.amazon.com/timestream/latest/developerguide/code-samples.run-
query.html#code-samples.run-query.parsing

// But unlike SELECT, UNLOAD only has 1 row * 3 columns outputed
// (rows, metadataFile, manifestFile) => (BIGINT, VARCHAR, VARCHAR)

public UnloadResponse parseResult(QueryResult queryResult) {
```

```
    Map<String, String> outputMap = new HashMap<>();
    for (int i = 0; i < queryResult.getColumnInfo().size(); i++) {
        outputMap.put(queryResult.getColumnInfo().get(i).getName(),
                queryResult.getRows().get(0).getData().get(i).getScalarValue());

    }
    return new UnloadResponse(outputMap);
}

@Getter
class UnloadResponse {
    private final String metadataFile;
    private final String manifestFile;
    private final int rows;

    public UnloadResponse(Map<String, String> unloadResponse) {
        this.metadataFile = unloadResponse.get("metadataFile");
        this.manifestFile = unloadResponse.get("manifestFile");
        this.rows = Integer.parseInt(unloadResponse.get("rows"));
    }
}
```

Java v2

```
// Parsing UNLOAD query response is similar to how you parse SELECT query response:
// https://docs.aws.amazon.com/timestream/latest/developerguide/code-samples.run-
query.html#code-samples.run-query.parsing

// But unlike SELECT, UNLOAD only has 1 row * 3 columns outputed
// (rows, metadataFile, manifestFile) => (BIGINT, VARCHAR, VARCHAR)

public UnloadResponse parseResult(QueryResponse queryResponse) {
    Map<String, String> outputMap = new HashMap<>();
    for (int i = 0; i < queryResponse.columnInfo().size(); i++) {
        outputMap.put(queryResponse.columnInfo().get(i).name(),
                queryResponse.rows().get(0).data().get(i).scalarValue());

    }
    return new UnloadResponse(outputMap);
}

@Getter
class UnloadResponse {
```

```
    private final String metadataFile;
    private final String manifestFile;
    private final int rows;

    public UnloadResponse(Map<String, String> unloadResponse) {
        this.metadataFile = unloadResponse.get("metadataFile");
        this.manifestFile = unloadResponse.get("manifestFile");
        this.rows = Integer.parseInt(unloadResponse.get("rows"));
    }
}
```

Go

```
// Parsing UNLOAD query response is similar to how you parse SELECT query response:
// https://docs.aws.amazon.com/timestream/latest/developerguide/code-samples.run-
query.html#code-samples.run-query.parsing

// But unlike SELECT, UNLOAD only has 1 row * 3 columns outputed
// (rows, metadataFile, manifestFile) => (BIGINT, VARCHAR, VARCHAR)

func parseQueryResult(queryOutput *timestreamquery.QueryOutput) map[string]string {
    var columnInfo = queryOutput.ColumnInfo;
    fmt.Println("ColumnInfo", columnInfo)
    fmt.Println("QueryId", queryOutput.QueryId)
    fmt.Println("QueryStatus", queryOutput.QueryStatus)
    return parseResponse(columnInfo, queryOutput.Rows[0])
}

func parseResponse(columnInfo []*timestreamquery.ColumnInfo, row
 *timestreamquery.Row) map[string]string {
    var datum = row.Data
    response := make(map[string]string)
    for i, column := range columnInfo {
        response[*column.Name] = *datum[i].ScalarValue
    }
    return response
}
```

Python

```
# Parsing UNLOAD query response is similar to how you parse SELECT query response:
# https://docs.aws.amazon.com/timestream/latest/developerguide/code-samples.run-
query.html#code-samples.run-query.parsing
```

```
# But unlike SELECT, UNLOAD only has 1 row * 3 columns outputed
# (rows, metadataFile, manifestFile) => (BIGINT, VARCHAR, VARCHAR)

for page in page_iterator:
  last_page = page
response = self._parse_unload_query_result(last_page)

def _parse_unload_query_result(self, query_result):
    column_info = query_result['ColumnInfo']

    print("ColumnInfo: %s" % column_info)
    print("QueryId: %s" % query_result['QueryId'])
    print("QueryStatus:%s" % query_result['QueryStatus'])
    return self.parse_unload_response(column_info, query_result['Rows'][0])

def parse_unload_response(self, column_info, row):
    response = {}
    data = row['Data']
    for i, column in enumerate(column_info):
        response[column['Name']] = data[i]['ScalarValue']
   print("Rows: %s" % response['rows'])
   print("Metadata File location: %s" % response['metadataFile'])
   print("Manifest File location: %s" % response['manifestFile'])
   return response
```

Node.js

```
# Parsing UNLOAD query response is similar to how you parse SELECT query response:
# https://docs.aws.amazon.com/timestream/latest/developerguide/code-samples.run-
query.html#code-samples.run-query.parsing

# But unlike SELECT, UNLOAD only has 1 row * 3 columns outputed
# (rows, metadataFile, manifestFile) => (BIGINT, VARCHAR, VARCHAR)

async parseAndDisplayResults(data, query) {
    const columnInfo = data['ColumnInfo'];
    console.log("ColumnInfo:", columnInfo)
    console.log("QueryId: %s", data['QueryId'])
    console.log("QueryStatus:", data['QueryStatus'])
    await this.parseResponse(columnInfo, data['Rows'][0], query)
}
```

```
async parseResponse(columnInfo, row, query) {
    let response = {}
    const data = row['Data']
    columnInfo.forEach((column, i) => {
        response[column['Name']] = data[i]['ScalarValue']
    })

   console.log("Manifest file", response['manifestFile']);
   console.log("Metadata file", response['metadataFile']);

   return response
}
```

Membaca dan mengurai konten manifes

Java

```
// Read and parse manifest content
public UnloadManifest getUnloadManifest(UnloadResponse unloadResponse) throws
 IOException {
   AmazonS3URI s3URI = new AmazonS3URI(unloadResponse.getManifestFile());
    S3Object s3Object = s3Client.getObject(s3URI.getBucket(), s3URI.getKey());
    String manifestFileContent = new
 String(IOUtils.toByteArray(s3Object.getObjectContent()), StandardCharsets.UTF_8);
    return new Gson().fromJson(manifestFileContent, UnloadManifest.class);
}

class UnloadManifest {
    @Getter
    public class FileMetadata {
        long content_length_in_bytes;
        long row_count;
    }

    @Getter
    public class ResultFile {
        String url;
        FileMetadata file_metadata;
    }

    @Getter
    public class QueryMetadata {
```

```
        long total_content_length_in_bytes;
        long total_row_count;
        String result_format;
        String result_version;
    }

    @Getter
    public class Author {
        String name;
        String manifest_file_version;
    }

    @Getter
    private List<ResultFile> result_files;
    @Getter
    private QueryMetadata query_metadata;
    @Getter
    private Author author;
}
```

Java v2

```
// Read and parse manifest content
public UnloadManifest getUnloadManifest(UnloadResponse unloadResponse) throws
 URISyntaxException {
    // Space needs to encoded to use S3 parseUri function
    S3Uri s3Uri =
 s3Utilities.parseUri(URI.create(unloadResponse.getManifestFile().replace(" ",
 "%20")));
    ResponseBytes<GetObjectResponse> objectBytes =
 s3Client.getObjectAsBytes(GetObjectRequest.builder()
            .bucket(s3Uri.bucket().orElseThrow(() -> new
 URISyntaxException(unloadResponse.getManifestFile(), "Invalid S3 URI")))
            .key(s3Uri.key().orElseThrow(() -> new
 URISyntaxException(unloadResponse.getManifestFile(), "Invalid S3 URI")))
            .build());
    String manifestFileContent = new String(objectBytes.asByteArray(),
 StandardCharsets.UTF_8);
    return new Gson().fromJson(manifestFileContent, UnloadManifest.class);
}

class UnloadManifest {
    @Getter
```

```
    public class FileMetadata {
        long content_length_in_bytes;
        long row_count;
    }

    @Getter
    public class ResultFile {
        String url;
        FileMetadata file_metadata;
    }

    @Getter
    public class QueryMetadata {
        long total_content_length_in_bytes;
        long total_row_count;
        String result_format;
        String result_version;
    }

    @Getter
    public class Author {
        String name;
        String manifest_file_version;
    }

    @Getter
    private List<ResultFile> result_files;
    @Getter
    private QueryMetadata query_metadata;
    @Getter
    private Author author;
}
```

Go

```
// Read and parse manifest content

func getManifestFile(s3Svc *s3.S3, response map[string]string) Manifest {
    var manifestBuf = getObject(s3Svc, response["manifestFile"])
    var manifest Manifest
    json.Unmarshal(manifestBuf.Bytes(), &manifest)
    return manifest
}
```

```
func getObject(s3Svc *s3.S3, s3Uri string) *bytes.Buffer {
    u,_ := url.Parse(s3Uri)
    getObjectInput := &s3.GetObjectInput{
        Key:    aws.String(u.Path),
        Bucket: aws.String(u.Host),
    }
    getObjectOutput, err := s3Svc.GetObject(getObjectInput)
    if err != nil {
        fmt.Println("Error: %s\n", err.Error())
    }
    buf := new(bytes.Buffer)
    buf.ReadFrom(getObjectOutput.Body)
    return buf
}

// Unload's Manifest structure

type Manifest struct {
    Author interface{}
    Query_metadata map[string]any
    Result_files  []struct {
        File_metadata interface{}
        Url string
    }
}}
```

Python

```
def __get_manifest_file(self, response):
    manifest = self.get_object(response['manifestFile']).read().decode('utf-8')
    parsed_manifest = json.loads(manifest)
    print("Manifest contents: \n%s" % parsed_manifest)

def get_object(self, uri):
    try:
        bucket, key = uri.replace("s3://", "").split("/", 1)
        s3_client = boto3.client('s3', region_name=<region>)
        response = s3_client.get_object(Bucket=bucket, Key=key)
        return response['Body']
    except Exception as err:
        print("Failed to get the object for URI:", uri)
        raise err
```

Node.js

```
// Read and parse manifest content

async getManifestFile(response) {
    let manifest;
    await this.getS3Object(response['manifestFile']).then(
        (data) => {
            manifest = JSON.parse(data);
        }
    );
    return manifest;
}

async getS3Object(uri) {
    const {bucketName, key} = this.getBucketAndKey(uri);
    const params = new GetObjectCommand({
        Bucket: bucketName,
        Key: key
    })
    const response = await this.s3Client.send(params);
    return await response.Body.transformToString();
}

getBucketAndKey(uri) {
    const [bucketName] = uri.replace("s3://", "").split("/", 1);
    const key = uri.replace("s3://", "").split('/').slice(1).join('/');
    return {bucketName, key};
}
```

Membaca dan mengurai konten metadata

Java

```
// Read and parse metadata content
public UnloadMetadata getUnloadMetadata(UnloadResponse unloadResponse) throws
 IOException {
   AmazonS3URI s3URI = new AmazonS3URI(unloadResponse.getMetadataFile());
   S3Object s3Object = s3Client.getObject(s3URI.getBucket(), s3URI.getKey());
    String metadataFileContent = new
 String(IOUtils.toByteArray(s3Object.getObjectContent()), StandardCharsets.UTF_8);
    final Gson gson = new GsonBuilder()
```

```
            .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
            .create();
    return gson.fromJson(metadataFileContent, UnloadMetadata.class);
}

class UnloadMetadata {
    @JsonProperty("ColumnInfo")
    List<ColumnInfo> columnInfo;
    @JsonProperty("Author")
    Author author;

    @Data
    public class Author {
        @JsonProperty("Name")
        String name;
        @JsonProperty("MetadataFileVersion")
        String metadataFileVersion;
    }
}
```

Java v2

```
// Read and parse metadata content

public UnloadMetadata getUnloadMetadata(UnloadResponse unloadResponse) throws
 URISyntaxException {
   // Space needs to encoded to use S3 parseUri function
    S3Uri s3Uri =
 s3Utilities.parseUri(URI.create(unloadResponse.getMetadataFile().replace(" ",
 "%20")));
    ResponseBytes<GetObjectResponse> objectBytes =
 s3Client.getObjectAsBytes(GetObjectRequest.builder()
            .bucket(s3Uri.bucket().orElseThrow(() -> new
 URISyntaxException(unloadResponse.getMetadataFile(), "Invalid S3 URI")))
            .key(s3Uri.key().orElseThrow(() -> new
 URISyntaxException(unloadResponse.getMetadataFile(), "Invalid S3 URI")))
            .build());
    String metadataFileContent = new String(objectBytes.asByteArray(),
 StandardCharsets.UTF_8);
    final Gson gson = new GsonBuilder()
            .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
            .create();
    return gson.fromJson(metadataFileContent, UnloadMetadata.class);
```

```
}

class UnloadMetadata {
    @JsonProperty("ColumnInfo")
    List<ColumnInfo> columnInfo;
    @JsonProperty("Author")
    Author author;

    @Data
    public class Author {
        @JsonProperty("Name")
        String name;
        @JsonProperty("MetadataFileVersion")
        String metadataFileVersion;
    }
}
```

Go

```
// Read and parse metadata content

func getMetadataFile(s3Svc *s3.S3, response map[string]string) Metadata {
    var metadataBuf = getObject(s3Svc, response["metadataFile"])
    var metadata Metadata
    json.Unmarshal(metadataBuf.Bytes(), &metadata)
    return metadata
}

func getObject(s3Svc *s3.S3, s3Uri string) *bytes.Buffer {
    u,_ := url.Parse(s3Uri)
    getObjectInput := &s3.GetObjectInput{
        Key:    aws.String(u.Path),
        Bucket: aws.String(u.Host),
    }
    getObjectOutput, err := s3Svc.GetObject(getObjectInput)
    if err != nil {
        fmt.Println("Error: %s\n", err.Error())
    }
    buf := new(bytes.Buffer)
    buf.ReadFrom(getObjectOutput.Body)
    return buf
}
```

```
// Unload's Metadata structure

type Metadata struct {
    Author interface{}
    ColumnInfo []struct {
        Name string
        Type map[string]string
    }
}
```

Python

```
def __get_metadata_file(self, response):
    metadata = self.get_object(response['metadataFile']).read().decode('utf-8')
    parsed_metadata = json.loads(metadata)
   print("Metadata contents: \n%s" % parsed_metadata)

def get_object(self, uri):
    try:
        bucket, key = uri.replace("s3://", "").split("/", 1)
        s3_client = boto3.client('s3', region_name=<region>)
        response = s3_client.get_object(Bucket=bucket, Key=key)
        return response['Body']
    except Exception as err:
        print("Failed to get the object for URI:", uri)
        raise err
```

Node.js

```
// Read and parse metadata content
async getMetadataFile(response) {
    let metadata;
    await this.getS3Object(response['metadataFile']).then(
        (data) => {
            metadata = JSON.parse(data);
        }
    );
    return metadata;
}

async getS3Object(uri) {
```

```
    const {bucketName, key} = this.getBucketAndKey(uri);
    const params = new GetObjectCommand({
        Bucket: bucketName,
        Key: key
    })
    const response = await this.s3Client.send(params);
    return await response.Body.transformToString();
}

getBucketAndKey(uri) {
    const [bucketName] = uri.replace("s3://", "").split("/", 1);
    const key = uri.replace("s3://", "").split('/').slice(1).join('/');
    return {bucketName, key};
}
```

Batalkan kueri

Anda dapat menggunakan cuplikan kode berikut untuk membatalkan kueri.

> **Note**
>
> Cuplikan kode ini didasarkan pada contoh aplikasi lengkap. GitHub Untuk informasi selengkapnya tentang cara memulai dengan contoh aplikasi, lihatAplikasi sampel.

Java

```
    public void cancelQuery() {
        System.out.println("Starting query: " + SELECT_ALL_QUERY);
        QueryRequest queryRequest = new QueryRequest();
        queryRequest.setQueryString(SELECT_ALL_QUERY);
        QueryResult queryResult = queryClient.query(queryRequest);

        System.out.println("Cancelling the query: " + SELECT_ALL_QUERY);
        final CancelQueryRequest cancelQueryRequest = new CancelQueryRequest();
        cancelQueryRequest.setQueryId(queryResult.getQueryId());
        try {
            queryClient.cancelQuery(cancelQueryRequest);
            System.out.println("Query has been successfully cancelled");
        } catch (Exception e) {
```

```
            System.out.println("Could not cancel the query: " + SELECT_ALL_QUERY + "
 = " + e);
        }
    }
```

Java v2

```
    public void cancelQuery() {
        System.out.println("Starting query: " + SELECT_ALL_QUERY);
        QueryRequest queryRequest =
 QueryRequest.builder().queryString(SELECT_ALL_QUERY).build();
        QueryResponse queryResponse = timestreamQueryClient.query(queryRequest);

        System.out.println("Cancelling the query: " + SELECT_ALL_QUERY);
        final CancelQueryRequest cancelQueryRequest = CancelQueryRequest.builder()
                .queryId(queryResponse.queryId()).build();
        try {
            timestreamQueryClient.cancelQuery(cancelQueryRequest);
            System.out.println("Query has been successfully cancelled");
        } catch (Exception e) {
            System.out.println("Could not cancel the query: " + SELECT_ALL_QUERY + "
 = " + e);
        }
    }
```

Go

```
cancelQueryInput := &timestreamquery.CancelQueryInput{
      QueryId: aws.String(*queryOutput.QueryId),
  }

  fmt.Println("Submitting cancellation for the query")
  fmt.Println(cancelQueryInput)

  // submit the query
  cancelQueryOutput, err := querySvc.CancelQuery(cancelQueryInput)

  if err != nil {
      fmt.Println("Error:")
      fmt.Println(err)
  } else {
    fmt.Println("Query has been cancelled successfully")
    fmt.Println(cancelQueryOutput)
```

```
}
```

Python

```
    def cancel_query(self):
        print("Starting query: " + self.SELECT_ALL)
        result = self.client.query(QueryString=self.SELECT_ALL)
        print("Cancelling query: " + self.SELECT_ALL)
        try:
            self.client.cancel_query(QueryId=result['QueryId'])
            print("Query has been successfully cancelled")
        except Exception as err:
            print("Cancelling query failed:", err)
```

Node.js

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada contoh aplikasi di Node.js sampel Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
async function tryCancelQuery() {
    const params = {
        QueryString: SELECT_ALL_QUERY
    };
    console.log(`Running query: ${SELECT_ALL_QUERY}`);

    await queryClient.query(params).promise()
        .then(
            async (response) => {
                await cancelQuery(response.QueryId);
            },
            (err) => {
                console.error("Error while executing select all query:", err);
            });
}

async function cancelQuery(queryId) {
    const cancelParams = {
        QueryId: queryId
    };
    console.log(`Sending cancellation for query: ${SELECT_ALL_QUERY}`);
    await queryClient.cancelQuery(cancelParams).promise()
        .then(
            (response) => {
```

```
                console.log("Query has been cancelled successfully");
            },
            (err) => {
                console.error("Error while cancelling select all:", err);
            });
}
```

.NET

```
        public async Task CancelQuery()
        {
            Console.WriteLine("Starting query: " + SELECT_ALL_QUERY);
            QueryRequest queryRequest = new QueryRequest();
            queryRequest.QueryString = SELECT_ALL_QUERY;
            QueryResponse queryResponse = await
 queryClient.QueryAsync(queryRequest);

            Console.WriteLine("Cancelling query: " + SELECT_ALL_QUERY);
            CancelQueryRequest cancelQueryRequest = new CancelQueryRequest();
            cancelQueryRequest.QueryId = queryResponse.QueryId;

            try
            {
                await queryClient.CancelQueryAsync(cancelQueryRequest);
                Console.WriteLine("Query has been successfully cancelled.");
            } catch(Exception e)
            {
                Console.WriteLine("Could not cancel the query: " + SELECT_ALL_QUERY
 + " = " + e);
            }
        }
```

Buat tugas pemuatan batch

Anda dapat menggunakan cuplikan kode berikut untuk membuat tugas pemuatan batch.

Java

```
package com.example.tryit;

import java.util.Arrays;
```

```
import
 software.amazon.awssdk.services.timestreamwrite.model.CreateBatchLoadTaskRequest;
import
 software.amazon.awssdk.services.timestreamwrite.model.CreateBatchLoadTaskResponse;
import software.amazon.awssdk.services.timestreamwrite.model.DataModel;
import software.amazon.awssdk.services.timestreamwrite.model.DataModelConfiguration;
import
 software.amazon.awssdk.services.timestreamwrite.model.DataSourceConfiguration;
import
 software.amazon.awssdk.services.timestreamwrite.model.DataSourceS3Configuration;
import software.amazon.awssdk.services.timestreamwrite.model.DimensionMapping;
import
 software.amazon.awssdk.services.timestreamwrite.model.MultiMeasureAttributeMapping;
import software.amazon.awssdk.services.timestreamwrite.model.MultiMeasureMappings;
import software.amazon.awssdk.services.timestreamwrite.model.ReportConfiguration;
import software.amazon.awssdk.services.timestreamwrite.model.ReportS3Configuration;
import software.amazon.awssdk.services.timestreamwrite.model.ScalarMeasureValueType;
import software.amazon.awssdk.services.timestreamwrite.model.TimeUnit;
import software.amazon.awssdk.services.timestreamwrite.TimestreamWriteClient;

public class BatchLoadExample {
    public static final String DATABASE_NAME = <database name>;
    public static final String TABLE_NAME = <table name>;
    public static final String INPUT_BUCKET = <S3 location>;
    public static final String INPUT_OBJECT_KEY_PREFIX = <CSV filename>;
    public static final String REPORT_BUCKET = <S3 location>;
    public static final long HT_TTL_HOURS = 24L;
    public static final long CT_TTL_DAYS = 7L;

    TimestreamWriteClient amazonTimestreamWrite;

    public BatchLoadExample(TimestreamWriteClient client) {
        this.amazonTimestreamWrite = client;
    }

    public String createBatchLoadTask() {
        System.out.println("Creating batch load task");

        CreateBatchLoadTaskRequest request = CreateBatchLoadTaskRequest.builder()
                .dataModelConfiguration(DataModelConfiguration.builder()
                        .dataModel(DataModel.builder()
                                .timeColumn("timestamp")
                                .timeUnit(TimeUnit.SECONDS)
                                .dimensionMappings(Arrays.asList(
```

```
                                        DimensionMapping.builder()
                                                .sourceColumn("vehicle")
                                                .build(),
                                        DimensionMapping.builder()
                                                .sourceColumn("registration")
                                                .destinationColumn("license")
                                                .build()))
                                .multiMeasureMappings(MultiMeasureMappings.builder()
                                        .targetMultiMeasureName("mva_measure_name")

.multiMeasureAttributeMappings(Arrays.asList(

MultiMeasureAttributeMapping.builder()
                                                                .sourceColumn("wgt")

.targetMultiMeasureAttributeName("weight")

.measureValueType(ScalarMeasureValueType.DOUBLE)
                                                                .build(),

MultiMeasureAttributeMapping.builder()
                                                                .sourceColumn("spd")

.targetMultiMeasureAttributeName("speed")

.measureValueType(ScalarMeasureValueType.DOUBLE)
                                                                .build(),

MultiMeasureAttributeMapping.builder()
                                                                .sourceColumn("fuel")

.measureValueType(ScalarMeasureValueType.DOUBLE)
                                                                .build(),

MultiMeasureAttributeMapping.builder()
                                                                .sourceColumn("miles")

.measureValueType(ScalarMeasureValueType.DOUBLE)
                                                                .build()))
                                        .build())
                                .build())
                        .build())
                .dataSourceConfiguration(DataSourceConfiguration.builder()
                        .dataSourceS3Configuration(
```

```
                                DataSourceS3Configuration.builder()
                                        .bucketName(INPUT_BUCKET)
                                        .objectKeyPrefix(INPUT_OBJECT_KEY_PREFIX)
                                        .build())
                                .dataFormat("CSV")
                                .build())
                        .reportConfiguration(ReportConfiguration.builder()
                                .reportS3Configuration(ReportS3Configuration.builder()
                                        .bucketName(REPORT_BUCKET)
                                        .build())
                                .build())
                        .targetDatabaseName(DATABASE_NAME)
                        .targetTableName(TABLE_NAME)
                        .build();
                try {
                    final CreateBatchLoadTaskResponse createBatchLoadTaskResponse =
 amazonTimestreamWrite.createBatchLoadTask(request);
                    String taskId = createBatchLoadTaskResponse.taskId();
                    System.out.println("Successfully created batch load task: " + taskId);
                    return taskId;
                } catch (Exception e) {
                    System.out.println("Failed to create batch load task: " + e);
                    throw e;
                }
            }
}
```

Go

```
package main

import (
 "fmt"
 "context"
 "log"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/timestreamwrite"
 "github.com/aws/aws-sdk-go-v2/service/timestreamwrite/types"
)

func main() {
```

```
 customResolver := aws.EndpointResolverWithOptionsFunc(func(service, region string,
 options ...interface{})(aws.Endpoint, error) {
  if service == timestreamwrite.ServiceID &&  region == "us-west-2" {
      return aws.Endpoint{
          PartitionID:   "aws",
          URL:           <URL>,
          SigningRegion: "us-west-2",
      }, nil
  }
  return aws.Endpoint{}, &  aws.EndpointNotFoundError{}
 })

 cfg, err := config.LoadDefaultConfig(context.TODO(),
 config.WithEndpointResolverWithOptions(customResolver), config.WithRegion("us-
west-2"))

 if err != nil {
    log.Fatalf("failed to load configuration, %v", err)
 }

 client := timestreamwrite.NewFromConfig(cfg)

 response, err := client.CreateBatchLoadTask(context.TODO(), &
 timestreamwrite.CreateBatchLoadTaskInput{
            TargetDatabaseName: aws.String("BatchLoadExampleDatabase"),
            TargetTableName: aws.String("BatchLoadExampleTable"),
            RecordVersion: aws.Int64(1),
            DataModelConfiguration: &  types.DataModelConfiguration{
                DataModel: &  types.DataModel{
                    TimeColumn: aws.String("timestamp"),
                    TimeUnit: types.TimeUnitMilliseconds,
                    DimensionMappings: []types.DimensionMapping{
                        {
                            SourceColumn: aws.String("registration"),
                            DestinationColumn: aws.String("license"),
                        },
                    },
                    MultiMeasureMappings: &  types.MultiMeasureMappings{
                        TargetMultiMeasureName: aws.String("mva_measure_name"),
                        MultiMeasureAttributeMappings:
 []types.MultiMeasureAttributeMapping{
                            {
                                SourceColumn: aws.String("wgt"),
```

```
                                TargetMultiMeasureAttributeName:
 aws.String("weight"),
                                MeasureValueType:
 types.ScalarMeasureValueTypeDouble,
                            },
                            {
                                SourceColumn: aws.String("spd"),
                                TargetMultiMeasureAttributeName:
 aws.String("speed"),
                                MeasureValueType:
 types.ScalarMeasureValueTypeDouble,
                            },
                            {
                                SourceColumn: aws.String("fuel_consumption"),
                                TargetMultiMeasureAttributeName: aws.String("fuel"),
                                MeasureValueType:
 types.ScalarMeasureValueTypeDouble,
                            },
                        },
                    },
                },
            },
           DataSourceConfiguration: &  types.DataSourceConfiguration{
                DataSourceS3Configuration: &  types.DataSourceS3Configuration{
                    BucketName: aws.String("test-batch-load-west-2"),
                    ObjectKeyPrefix: aws.String("sample.csv"),
                },
                DataFormat: types.BatchLoadDataFormatCsv,
            },
           ReportConfiguration: & types.ReportConfiguration{
                ReportS3Configuration: & types.ReportS3Configuration{
                    BucketName: aws.String("test-batch-load-report-west-2"),
                    EncryptionOption: types.S3EncryptionOptionSseS3,
                },
            },
 })

 fmt.Println(aws.ToString(response.TaskId))
}
```

Python

```
import boto3
```

```
from botocore.config import Config

INGEST_ENDPOINT = "<URL>"
REGION = "us-west-2"
HT_TTL_HOURS = 24
CT_TTL_DAYS = 7
DATABASE_NAME = "<database name>"
TABLE_NAME = "<table name>"
INPUT_BUCKET_NAME = "<S3 location>"
INPUT_OBJECT_KEY_PREFIX = "<CSV file name>"
REPORT_BUCKET_NAME = "<S3 location>"


def create_batch_load_task(client, database_name, table_name, input_bucket_name,
 input_object_key_prefix, report_bucket_name):
    try:
        result = client.create_batch_load_task(TargetDatabaseName=database_name,
 TargetTableName=table_name,
                                               DataModelConfiguration={"DataModel":
 {
                                                   "TimeColumn": "timestamp",
                                                   "TimeUnit": "SECONDS",
                                                   "DimensionMappings": [
                                                       {
                                                           "SourceColumn": "vehicle"
                                                       },
                                                       {
                                                           "SourceColumn":
 "registration",
                                                           "DestinationColumn":
 "license"
                                                       }
                                                   ],
                                                   "MultiMeasureMappings": {
                                                       "TargetMultiMeasureName":
 "metrics",

 "MultiMeasureAttributeMappings": [
                                                           {
                                                               "SourceColumn":
 "wgt",
                                                               "MeasureValueType":
 "DOUBLE"
                                                           },
```

```
                                                    {
                                                        "SourceColumn":
"spd",
                                                        "MeasureValueType":
"DOUBLE"
                                                    },
                                                    {
                                                        "SourceColumn":
"fuel_consumption",

"TargetMultiMeasureAttributeName": "fuel",
                                                        "MeasureValueType":
"DOUBLE"
                                                    },
                                                    {
                                                        "SourceColumn":
"miles",
                                                        "MeasureValueType":
"DOUBLE"
                                                    }
                                                ]}
                                        }
                                        },
                                        DataSourceConfiguration={
                                            "DataSourceS3Configuration": {
                                                "BucketName":
input_bucket_name,
                                                "ObjectKeyPrefix":
input_object_key_prefix
                                            },
                                            "DataFormat": "CSV"
                                        },
                                        ReportConfiguration={
                                            "ReportS3Configuration": {
                                                "BucketName":
report_bucket_name,
                                                "EncryptionOption": "SSE_S3"
                                            }
                                        }
                                        )

        task_id = result["TaskId"]
        print("Successfully created batch load task: ", task_id)
        return task_id
```

```
    except Exception as err:
        print("Create batch load task job failed:", err)
        return None


if __name__ == '__main__':
    session = boto3.Session()

    write_client = session.client('timestream-write',
                                  endpoint_url=INGEST_ENDPOINT, region_name=REGION,
                                  config=Config(read_timeout=20,
 max_pool_connections=5000, retries={'max_attempts': 10}))

    task_id = create_batch_load_task(write_client, DATABASE_NAME, TABLE_NAME,
                                     INPUT_BUCKET_NAME, INPUT_OBJECT_KEY_PREFIX,
 REPORT_BUCKET_NAME)
```

Node.js

Cuplikan berikut digunakan AWS SDK untuk JavaScript v3. Untuk informasi selengkapnya tentang cara menginstal klien dan penggunaan, lihat Timestream Write Client - AWS SDK untuk JavaScript v3.

Untuk API detailnya, lihat Kelas CreateBatchLoadCommand dan CreateBatchLoadTask.

```
import { TimestreamWriteClient, CreateBatchLoadTaskCommand } from "@aws-sdk/client-
timestream-write";
const writeClient = new TimestreamWriteClient({ region: "us-west-2", endpoint:
 "https://gamma-ingest-cell3.timestream.us-west-2.amazonaws.com" });

const params = {
    TargetDatabaseName: "BatchLoadExampleDatabase",
    TargetTableName: "BatchLoadExampleTable",
 RecordVersion: 1,
 DataModelConfiguration: {
  DataModel: {
   TimeColumn: "timestamp",
            TimeUnit: "MILLISECONDS",
            DimensionMappings: [
                {
                    SourceColumn: "registration",
                    DestinationColumn: "license"
                }
```

```
            ],
            MultiMeasureMappings: {
                TargetMultiMeasureName: "mva_measure_name",
                MultiMeasureAttributeMappings: [
                    {
                        SourceColumn: "wgt",
                        TargetMultiMeasureAttributeName: "weight",
                        MeasureValueType: "DOUBLE"
                    },
                    {
                        SourceColumn: "spd",
                        TargetMultiMeasureAttributeName: "speed",
                        MeasureValueType: "DOUBLE"
                    },
                    {
                        SourceColumn: "fuel_consumption",
                        TargetMultiMeasureAttributeName: "fuel",
                        MeasureValueType: "DOUBLE"
                    }
                ]
            }
        }
    },
 DataSourceConfiguration: {
        DataSourceS3Configuration: {
            BucketName: "test-batch-load-west-2",
            ObjectKeyPrefix: "sample.csv"
        },
        DataFormat: "CSV"
    },
    ReportConfiguration: {
        ReportS3Configuration: {
            BucketName: "test-batch-load-report-west-2",
            EncryptionOption: "SSE_S3"
        }
    }
};

const command = new CreateBatchLoadTaskCommand(params);

try {
    const data = await writeClient.send(command);
    console.log(`Created batch load task ` + data.TaskId);
} catch (error) {
```

```
 console.log("Error creating table. ", error);
 throw error;
}
```

.NET

```
using System;
using System.IO;
using System.Collections.Generic;
using Amazon.TimestreamWrite;
using Amazon.TimestreamWrite.Model;
using System.Threading.Tasks;

namespace TimestreamDotNetSample
{
    public class CreateBatchLoadTaskExample
    {
        public const string DATABASE_NAME = "<database name>";
        public const string TABLE_NAME = "<table name>";
        public const string INPUT_BUCKET = "<input bucket name>";
        public const string INPUT_OBJECT_KEY_PREFIX = "<CSV file name>";
        public const string REPORT_BUCKET = "<report bucket name>";
        public const long HT_TTL_HOURS = 24L;
        public const long CT_TTL_DAYS = 7L;
        private readonly AmazonTimestreamWriteClient writeClient;

        public CreateBatchLoadTaskExample(AmazonTimestreamWriteClient writeClient)
        {
            this.writeClient = writeClient;
        }

        public async Task CreateBatchLoadTask()
        {
            try
            {
                var createBatchLoadTaskRequest = new CreateBatchLoadTaskRequest
                {
                    DataModelConfiguration = new DataModelConfiguration
                    {
                        DataModel = new DataModel
                        {
                            TimeColumn = "timestamp",
                            TimeUnit = TimeUnit.SECONDS,
```

```
                            DimensionMappings = new List<DimensionMapping>()
                            {
                                new()
                                {
                                        SourceColumn = "vehicle"
                                },
                                new()
                                {
                                        SourceColumn = "registration",
                                        DestinationColumn = "license"
                                }
                            },
                            MultiMeasureMappings = new MultiMeasureMappings
                            {
                                TargetMultiMeasureName = "mva_measure_name",
                                MultiMeasureAttributeMappings = new
List<MultiMeasureAttributeMapping>()
                                {
                                        new()
                                        {
                                                SourceColumn = "wgt",
                                                TargetMultiMeasureAttributeName =
"weight",
                                                MeasureValueType =
ScalarMeasureValueType.DOUBLE
                                        },
                                        new()
                                        {
                                                SourceColumn = "spd",
                                                TargetMultiMeasureAttributeName =
"speed",
                                                MeasureValueType =
ScalarMeasureValueType.DOUBLE
                                        },
                                        new()
                                        {
                                                SourceColumn = "fuel",
                                                TargetMultiMeasureAttributeName =
"fuel",
                                                MeasureValueType =
ScalarMeasureValueType.DOUBLE
                                        },
                                        new()
                                        {
```

```
                                                SourceColumn = "miles",
                                                TargetMultiMeasureAttributeName =
 "miles",
                                                MeasureValueType =
 ScalarMeasureValueType.DOUBLE
                                            }
                                        }
                                    }
                                }
                            },
                            DataSourceConfiguration = new DataSourceConfiguration
                            {
                                DataSourceS3Configuration = new DataSourceS3Configuration
                                {
                                    BucketName = INPUT_BUCKET,
                                    ObjectKeyPrefix = INPUT_OBJECT_KEY_PREFIX
                                },
                                DataFormat = "CSV"
                            },
                            ReportConfiguration = new ReportConfiguration
                            {
                                ReportS3Configuration = new ReportS3Configuration
                                {
                                    BucketName = REPORT_BUCKET
                                }
                            },
                            TargetDatabaseName = DATABASE_NAME,
                            TargetTableName = TABLE_NAME
                        };

                        CreateBatchLoadTaskResponse response = await
 writeClient.CreateBatchLoadTaskAsync(createBatchLoadTaskRequest);
                        Console.WriteLine($"Task created: " + response.TaskId);
                    }
                    catch (Exception e)
                    {
                        Console.WriteLine("Create batch load task failed:" + e.ToString());
                    }
                }
            }
        }
```

```
using Amazon.TimestreamWrite;
```

```
using Amazon.TimestreamWrite.Model;
using Amazon;
using Amazon.TimestreamQuery;
using System.Threading.Tasks;
using System;
using CommandLine;
static class Constants
{

}
namespace TimestreamDotNetSample
{
    class MainClass
    {
        public class Options
        {

        }
        public static void Main(string[] args)
        {
            Parser.Default.ParseArguments<Options>(args)
                .WithParsed<Options>(o => {
                    MainAsync().GetAwaiter().GetResult();
                });
        }

        static async Task MainAsync()
        {
            var writeClientConfig = new AmazonTimestreamWriteConfig
            {
                ServiceURL =  "<service URL>",
                Timeout = TimeSpan.FromSeconds(20),
                MaxErrorRetry = 10
            };

            var writeClient = new AmazonTimestreamWriteClient(writeClientConfig);
            var example = new CreateBatchLoadTaskExample(writeClient);
            await example.CreateBatchLoadTask();
        }
    }
}
```

Jelaskan tugas pemuatan batch

Anda dapat menggunakan cuplikan kode berikut untuk menjelaskan tugas pemuatan batch.

Java

```
    public void describeBatchLoadTask(String taskId) {
            final DescribeBatchLoadTaskResponse batchLoadTaskResponse =
 amazonTimestreamWrite

 .describeBatchLoadTask(DescribeBatchLoadTaskRequest.builder()
                                             .taskId(taskId)
                                             .build());

            System.out.println("Task id: " +
 batchLoadTaskResponse.batchLoadTaskDescription().taskId());
            System.out.println("Status: " +
 batchLoadTaskResponse.batchLoadTaskDescription().taskStatusAsString());
            System.out.println("Records processed: "
                               +
 batchLoadTaskResponse.batchLoadTaskDescription().progressReport().recordsProcessed());
    }
```

Go

```
package main

import (
 "fmt"
 "context"
 "log"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/timestreamwrite"
)

func main() {
 customResolver := aws.EndpointResolverWithOptionsFunc(func(service, region string,
 options ...interface{}) (aws.Endpoint, error) {
  if service == timestreamwrite.ServiceID && region == "us-west-2" {
      return aws.Endpoint{
          PartitionID:   "aws",
          URL:           <URL>,
```

```
          SigningRegion: "us-west-2",
      }, nil
  }
  return aws.Endpoint{}, &aws.EndpointNotFoundError{}
 })

 cfg, err := config.LoadDefaultConfig(context.TODO(),
 config.WithEndpointResolverWithOptions(customResolver), config.WithRegion("us-
west-2"))

 if err != nil {
    log.Fatalf("failed to load configuration, %v", err)
 }

 client := timestreamwrite.NewFromConfig(cfg)

 response, err := client.DescribeBatchLoadTask(context.TODO(),
 &timestreamwrite.DescribeBatchLoadTaskInput{
  TaskId: aws.String("<TaskId>"),
 })

 fmt.Println(aws.ToString(response.BatchLoadTaskDescription.TaskId))
}
```

Python

```
import boto3
from botocore.config import Config

INGEST_ENDPOINT="<url>"
REGION="us-west-2"
HT_TTL_HOURS = 24
CT_TTL_DAYS = 7
TASK_ID = "<task id>"

def describe_batch_load_task(client, task_id):
    try:
        result = client.describe_batch_load_task(TaskId=task_id)
        print("Successfully described batch load task: ", result)
    except Exception as err:
        print("Describe batch load task job failed:", err)
```

```
if __name__ == '__main__':
    session = boto3.Session()

    write_client = session.client('timestream-write', \
        endpoint_url=INGEST_ENDPOINT, region_name=REGION, \
        config=Config(read_timeout=20, max_pool_connections = 5000,
 retries={'max_attempts': 10}))

    describe_batch_load_task(write_client, TASK_ID)
```

Node.js

Cuplikan berikut digunakan AWS SDK untuk JavaScript v3. Untuk informasi selengkapnya tentang cara menginstal klien dan penggunaan, lihat Timestream Write Client - AWS SDK untuk JavaScript v3.

Untuk API detailnya, lihat Kelas DescribeBatchLoadCommand dan DescribeBatchLoadTask.

```
import { TimestreamWriteClient, DescribeBatchLoadTaskCommand } from "@aws-sdk/
client-timestream-write";
const writeClient = new TimestreamWriteClient({ region: "<region>", endpoint:
 "<endpoint>" });

const params = {
    TaskId: "<TaskId>"
};

const command = new DescribeBatchLoadTaskCommand(params);

try {
    const data = await writeClient.send(command);
    console.log(`Batch load task has id ` + data.BatchLoadTaskDescription.TaskId);
} catch (error) {
    if (error.code === 'ResourceNotFoundException') {
        console.log("Batch load task doesn't exist.");
    } else {
        console.log("Describe batch load task failed.", error);
        throw error;
    }
}
```

.NET

```
using System;
using System.IO;
using System.Collections.Generic;
using Amazon.TimestreamWrite;
using Amazon.TimestreamWrite.Model;
using System.Threading.Tasks;

namespace TimestreamDotNetSample
{
    public class DescribeBatchLoadTaskExample
    {
        private readonly AmazonTimestreamWriteClient writeClient;

        public DescribeBatchLoadTaskExample(AmazonTimestreamWriteClient writeClient)
        {
            this.writeClient = writeClient;
        }

        public async Task DescribeBatchLoadTask(String taskId)
        {
            try
            {
                var describeBatchLoadTaskRequest = new DescribeBatchLoadTaskRequest
                {
                    TaskId = taskId
                };
                DescribeBatchLoadTaskResponse response = await
 writeClient.DescribeBatchLoadTaskAsync(describeBatchLoadTaskRequest);
                Console.WriteLine($"Task has id:
{response.BatchLoadTaskDescription.TaskId}");
            }
            catch (ResourceNotFoundException)
            {
                Console.WriteLine("Batch load task does not exist.");
            }
            catch (Exception e)
            {
                Console.WriteLine("Describe batch load task failed:" +
 e.ToString());
            }
        }
    }
```

```
}
```

```
using Amazon.TimestreamWrite;
using Amazon.TimestreamWrite.Model;
using Amazon;
using Amazon.TimestreamQuery;
using System.Threading.Tasks;
using System;
using CommandLine;
static class Constants
{

}
namespace TimestreamDotNetSample
{
    class MainClass
    {
        public class Options
        {

        }
        public static void Main(string[] args)
        {
            Parser.Default.ParseArguments<Options>(args)
                .WithParsed<Options>(o => {
                    MainAsync().GetAwaiter().GetResult();
                });
        }

        static async Task MainAsync()
        {
            var writeClientConfig = new AmazonTimestreamWriteConfig
            {
                ServiceURL =  "<service URL>",
                Timeout = TimeSpan.FromSeconds(20),
                MaxErrorRetry = 10
            };

            var writeClient = new AmazonTimestreamWriteClient(writeClientConfig);
            var example = new DescribeBatchLoadTaskExample(writeClient);
            await example.DescribeBatchLoadTask("<batch load task id>");
        }
    }
```

```
}
```

Buat daftar tugas pemuatan batch

Anda dapat menggunakan cuplikan kode berikut untuk membuat daftar tugas pemuatan batch.

Java

```
    public void listBatchLoadTasks() {
            final ListBatchLoadTasksResponse listBatchLoadTasksResponse =
 amazonTimestreamWrite
                            .listBatchLoadTasks(ListBatchLoadTasksRequest.builder()
                                    .maxResults(15)
                                    .build());

            for (BatchLoadTask batchLoadTask :
 listBatchLoadTasksResponse.batchLoadTasks()) {
                    System.out.println(batchLoadTask.taskId());
            }
    }
```

Go

```
package main

import (
 "fmt"
 "context"
 "log"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/timestreamwrite"
)

func main() {
 customResolver := aws.EndpointResolverWithOptionsFunc(func(service, region string,
 options ...interface{}) (aws.Endpoint, error) {
  if service == timestreamwrite.ServiceID && region == "us-west-2" {
      return aws.Endpoint{
          PartitionID:   "aws",
          URL:           <URL>,
          SigningRegion: "us-west-2",
```

```
      }, nil
  }
  return aws.Endpoint{}, &aws.EndpointNotFoundError{}
 })

 cfg, err := config.LoadDefaultConfig(context.TODO(),
 config.WithEndpointResolverWithOptions(customResolver), config.WithRegion("us-
west-2"))

 if err != nil {
    log.Fatalf("failed to load configuration, %v", err)
 }

 client := timestreamwrite.NewFromConfig(cfg)
 listBatchLoadTasksMaxResult := int32(15)

 response, err := client.ListBatchLoadTasks(context.TODO(),
 &timestreamwrite.ListBatchLoadTasksInput{
  MaxResults: &listBatchLoadTasksMaxResult,
 })

 for i, task := range response.BatchLoadTasks {
  fmt.Println(i, aws.ToString(task.TaskId))
 }
}
```

Python

```
import boto3
from botocore.config import Config

INGEST_ENDPOINT = "<url>"
REGION = "us-west-2"
HT_TTL_HOURS = 24
CT_TTL_DAYS = 7


def print_batch_load_tasks(batch_load_tasks):
    for batch_load_task in batch_load_tasks:
        print(batch_load_task['TaskId'])


def list_batch_load_tasks(client):
```

```
    print("\nListing batch load tasks")
    try:
        response = client.list_batch_load_tasks(MaxResults=10)
        print_batch_load_tasks(response['BatchLoadTasks'])
        next_token = response.get('NextToken', None)
        while next_token:
            response = client.list_batch_load_tasks(
                NextToken=next_token, MaxResults=10)
            print_batch_load_tasks(response['BatchLoadTasks'])
            next_token = response.get('NextToken', None)
    except Exception as err:
        print("List batch load tasks failed:", err)
        raise err


if __name__ == '__main__':
    session = boto3.Session()

    write_client = session.client('timestream-write',
                                  endpoint_url=INGEST_ENDPOINT, region_name=REGION,
                                  config=Config(read_timeout=20,
 max_pool_connections=5000, retries={'max_attempts': 10}))

    list_batch_load_tasks(write_client)
```

Node.js

Cuplikan berikut digunakan AWS SDK untuk JavaScript v3. Untuk informasi selengkapnya tentang cara menginstal klien dan penggunaan, lihat Timestream Write Client - AWS SDK untuk JavaScript v3.

Untuk API detailnya, lihat Kelas DescribeBatchLoadCommand dan DescribeBatchLoadTask.

```
import { TimestreamWriteClient, ListBatchLoadTasksCommand } from "@aws-sdk/client-
timestream-write";
const writeClient = new TimestreamWriteClient({ region: "<region>", endpoint:
 "<endpoint>" });

const params = {
    MaxResults: <15>
};

const command = new ListBatchLoadTasksCommand(params);
```

```
getBatchLoadTasksList(null);

async function getBatchLoadTasksList(nextToken) {
    if (nextToken) {
        params.NextToken = nextToken;
    }

    try {
        const data = await writeClient.send(command);

        data.BatchLoadTasks.forEach(function (task) {
            console.log(task.TaskId);
        });

        if (data.NextToken) {
            return getBatchLoadTasksList(data.NextToken);
        }
    } catch (error) {
        console.log("Error while listing batch load tasks", error);
    }
}
```

.NET

```
using System;
using System.IO;
using System.Collections.Generic;
using Amazon.TimestreamWrite;
using Amazon.TimestreamWrite.Model;
using System.Threading.Tasks;

namespace TimestreamDotNetSample
{
    public class ListBatchLoadTasksExample
    {
        private readonly AmazonTimestreamWriteClient writeClient;

        public ListBatchLoadTasksExample(AmazonTimestreamWriteClient writeClient)
        {
            this.writeClient = writeClient;
        }
```

```
        public async Task ListBatchLoadTasks()
        {
            Console.WriteLine("Listing batch load tasks");

            try
            {
                var listBatchLoadTasksRequest = new ListBatchLoadTasksRequest
                {
                    MaxResults = 15
                };

                ListBatchLoadTasksResponse response = await
 writeClient.ListBatchLoadTasksAsync(listBatchLoadTasksRequest);

                PrintBatchLoadTasks(response.BatchLoadTasks);
                var nextToken = response.NextToken;

                while (nextToken != null)
                {
                    listBatchLoadTasksRequest.NextToken = nextToken;
                    response = await
 writeClient.ListBatchLoadTasksAsync(listBatchLoadTasksRequest);
                    PrintBatchLoadTasks(response.BatchLoadTasks);
                    nextToken = response.NextToken;
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("List batch load tasks failed:" + e.ToString());
            }
        }

        private void PrintBatchLoadTasks(List<BatchLoadTask> tasks)
        {
            foreach (BatchLoadTask task in tasks)
                Console.WriteLine($"Task:{task.TaskId}");
        }
    }
}
```

```
using Amazon.TimestreamWrite;
using Amazon.TimestreamWrite.Model;
using Amazon;
```

```
using Amazon.TimestreamQuery;
using System.Threading.Tasks;
using System;
using CommandLine;
static class Constants
{

}
namespace TimestreamDotNetSample
{
    class MainClass
    {
        public class Options
        {

        }
        public static void Main(string[] args)
        {
            Parser.Default.ParseArguments<Options>(args)
                .WithParsed<Options>(o => {
                    MainAsync().GetAwaiter().GetResult();
                });
        }

        static async Task MainAsync()
        {
            var writeClientConfig = new AmazonTimestreamWriteConfig
            {
                ServiceURL =  "<service URL>",
                Timeout = TimeSpan.FromSeconds(20),
                MaxErrorRetry = 10
            };

            var writeClient = new AmazonTimestreamWriteClient(writeClientConfig);
            var example = new ListBatchLoadTasksExample(writeClient);
            await example.ListBatchLoadTasks();
        }
    }
}
```

Lanjutkan tugas pemuatan batch

Anda dapat menggunakan cuplikan kode berikut untuk melanjutkan tugas pemuatan batch.

Java

```
    public void resumeBatchLoadTask(String taskId) {
            try {
                    amazonTimestreamWrite

 .resumeBatchLoadTask(ResumeBatchLoadTaskRequest.builder()
                                                    .taskId(taskId)
                                                    .build());

                    System.out.println("Successfully resumed batch load task.");
            } catch (ValidationException validationException) {
                    System.out.println(validationException.getMessage());
            }
    }
```

Go

```
package main

import (
 "fmt"
 "context"
 "log"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/timestreamwrite"
)

func main() {
 customResolver := aws.EndpointResolverWithOptionsFunc(func(service, region string,
 options ...interface{}) (aws.Endpoint, error) {
  if service == timestreamwrite.ServiceID && region == "us-west-2" {
      return aws.Endpoint{
          PartitionID:   "aws",
          URL:           <URL>,
          SigningRegion: "us-west-2",
      }, nil
  }
```

```
  return aws.Endpoint{}, &aws.EndpointNotFoundError{}
 })

 cfg, err := config.LoadDefaultConfig(context.TODO(),
 config.WithEndpointResolverWithOptions(customResolver), config.WithRegion("us-
west-2"))

 if err != nil {
    log.Fatalf("failed to load configuration, %v", err)
 }

 client := timestreamwrite.NewFromConfig(cfg)

 response, err := client.ResumeBatchLoadTask(context.TODO(),
 &timestreamwrite.ResumeBatchLoadTaskInput{
  TaskId: aws.String("TaskId"),
 })

 if err != nil {
  fmt.Println("Error:")
  fmt.Println(err)
 } else {
  fmt.Println("Resume batch load task is successful")
  fmt.Println(response)
 }
}
```

Python

```
import boto3
from botocore.config import Config

INGEST_ENDPOINT="<url>"
REGION="us-west-2"
HT_TTL_HOURS = 24
CT_TTL_DAYS = 7
TASK_ID = "<TaskId>"

def resume_batch_load_task(client, task_id):
    try:
        result = client.resume_batch_load_task(TaskId=task_id)
        print("Successfully resumed batch load task: ", result)
    except Exception as err:
```

```
        print("Resume batch load task failed:", err)


if __name__ == '__main__':
    session = boto3.Session()

    write_client = session.client('timestream-write', \
        endpoint_url=INGEST_ENDPOINT, region_name=REGION, \
        config=Config(read_timeout=20, max_pool_connections = 5000,
 retries={'max_attempts': 10}))

    resume_batch_load_task(write_client, TASK_ID)
```

Node.js

Cuplikan berikut digunakan AWS SDK untuk JavaScript v3. Untuk informasi selengkapnya tentang cara menginstal klien dan penggunaan, lihat Timestream Write Client - AWS SDK untuk JavaScript v3.

Untuk API detailnya, lihat Kelas CreateBatchLoadCommand dan CreateBatchLoadTask.

```
import { TimestreamWriteClient, ResumeBatchLoadTaskCommand } from "@aws-sdk/client-
timestream-write";
const writeClient = new TimestreamWriteClient({ region: "<region>", endpoint:
 "<endpoint>" });

const params = {
    TaskId: "<TaskId>"
};

const command = new ResumeBatchLoadTaskCommand(params);

try {
    const data = await writeClient.send(command);
    console.log("Resumed batch load task");
} catch (error) {
    console.log("Resume batch load task failed.", error);
 throw error;
}
```

.NET

```
using System;
```

```
using System.IO;
using System.Collections.Generic;
using Amazon.TimestreamWrite;
using Amazon.TimestreamWrite.Model;
using System.Threading.Tasks;

namespace TimestreamDotNetSample
{
    public class ResumeBatchLoadTaskExample
    {
        private readonly AmazonTimestreamWriteClient writeClient;

        public ResumeBatchLoadTaskExample(AmazonTimestreamWriteClient writeClient)
        {
            this.writeClient = writeClient;
        }

        public async Task ResumeBatchLoadTask(String taskId)
        {
            try
            {
                var resumeBatchLoadTaskRequest = new ResumeBatchLoadTaskRequest
                {
                    TaskId = taskId
                };
                ResumeBatchLoadTaskResponse response = await
 writeClient.ResumeBatchLoadTaskAsync(resumeBatchLoadTaskRequest);
                Console.WriteLine("Successfully resumed batch load task.");
            }
            catch (ResourceNotFoundException)
            {
                Console.WriteLine("Batch load task does not exist.");
            }
            catch (Exception e)
            {
                Console.WriteLine("Resume batch load task failed: " + e.ToString());
            }
        }
    }
}
```

Buat kueri terjadwal

Anda dapat menggunakan cuplikan kode berikut untuk membuat kueri terjadwal dengan pemetaan multi-ukuran.

Java

```
public static String DATABASE_NAME = "devops_sample_application";
public static String TABLE_NAME = "host_metrics_sample_application";
public static String HOSTNAME = "host-24Gju";
public static String SQ_NAME = "daily-sample";
public static String SCHEDULE_EXPRESSION = "cron(0/2 * * * ? *)";

// Find the average, p90, p95, and p99 CPU utilization for a specific EC2 host over
 the past 2 hours.
public static String QUERY = "SELECT region, az, hostname, BIN(time, 15s) AS
 binned_timestamp, " +
"ROUND(AVG(cpu_utilization), 2) AS avg_cpu_utilization, " +
"ROUND(APPROX_PERCENTILE(cpu_utilization, 0.9), 2) AS p90_cpu_utilization, " +
"ROUND(APPROX_PERCENTILE(cpu_utilization, 0.95), 2) AS p95_cpu_utilization, " +
"ROUND(APPROX_PERCENTILE(cpu_utilization, 0.99), 2) AS p99_cpu_utilization " +
"FROM " +  DATABASE_NAME + "." +  TABLE_NAME + " " +
"WHERE measure_name = 'metrics' " +
"AND hostname = '" + HOSTNAME + "' " +
"AND time > ago(2h) " +
"GROUP BY region, hostname, az, BIN(time, 15s) " +
"ORDER BY binned_timestamp ASC " +
"LIMIT 5";


public String createScheduledQuery(String topic_arn,
    String role_arn,
    String database_name,
    String table_name) {
    System.out.println("Creating Scheduled Query");

    List<Pair<String, MeasureValueType>> sourceColToMeasureValueTypes =
 Arrays.asList(
        Pair.of("avg_cpu_utilization", DOUBLE),
        Pair.of("p90_cpu_utilization", DOUBLE),
        Pair.of("p95_cpu_utilization", DOUBLE),
        Pair.of("p99_cpu_utilization", DOUBLE));
```

```
   CreateScheduledQueryRequest createScheduledQueryRequest = new
CreateScheduledQueryRequest()
           .withName(SQ_NAME)
           .withQueryString(QUERY)
           .withScheduleConfiguration(new ScheduleConfiguration()
                   .withScheduleExpression(SCHEDULE_EXPRESSION))
           .withNotificationConfiguration(new NotificationConfiguration()
                   .withSnsConfiguration(new SnsConfiguration()
                           .withTopicArn(topic_arn)))
           .withTargetConfiguration(new
TargetConfiguration().withTimestreamConfiguration(new TimestreamConfiguration()
                   .withDatabaseName(database_name)
                   .withTableName(table_name)
                   .withTimeColumn("binned_timestamp")
                   .withDimensionMappings(Arrays.asList(
                           new DimensionMapping()
                                   .withName("region")
                                   .withDimensionValueType("VARCHAR"),
                           new DimensionMapping()
                                   .withName("az")
                                   .withDimensionValueType("VARCHAR"),
                           new DimensionMapping()
                                   .withName("hostname")
                                   .withDimensionValueType("VARCHAR")
                   ))
                   .withMultiMeasureMappings(new MultiMeasureMappings()
                       .withTargetMultiMeasureName("multi-metrics")
                       .withMultiMeasureAttributeMappings(
                           sourceColToMeasureValueTypes.stream()
                           .map(pair -> new MultiMeasureAttributeMapping()
                               .withMeasureValueType(pair.getValue().name())
                               .withSourceColumn(pair.getKey()))
                           .collect(Collectors.toList()))))
           .withErrorReportConfiguration(new ErrorReportConfiguration()
                   .withS3Configuration(new S3Configuration()

.withBucketName(timestreamDependencyHelper.getS3ErrorReportBucketName())))
           .withScheduledQueryExecutionRoleArn(role_arn);

   try {
       final CreateScheduledQueryResult createScheduledQueryResult =
queryClient.createScheduledQuery(createScheduledQueryRequest);
       final String scheduledQueryArn = createScheduledQueryResult.getArn();
```

```
        System.out.println("Successfully created scheduled query : " +
 scheduledQueryArn);
        return scheduledQueryArn;
    }
    catch (Exception e) {
        System.out.println("Scheduled Query creation failed: " + e);
        throw e;
    }
}
```

Java v2

```
public static String DATABASE_NAME = "testJavaV2DB";
public static String TABLE_NAME = "testJavaV2Table";
public static String HOSTNAME = "host-24Gju";
public static String SQ_NAME = "daily-sample";
public static String SCHEDULE_EXPRESSION = "cron(0/2 * * * ? *)";

// Find the average, p90, p95, and p99 CPU utilization for a specific EC2 host over
 the past 2 hours.
public static String VALID_QUERY = "SELECT region, az, hostname, BIN(time, 15s) AS
 binned_timestamp, " +
"ROUND(AVG(cpu_utilization), 2) AS avg_cpu_utilization, " +
"ROUND(APPROX_PERCENTILE(cpu_utilization, 0.9), 2) AS p90_cpu_utilization, " +
"ROUND(APPROX_PERCENTILE(cpu_utilization, 0.95), 2) AS p95_cpu_utilization, " +
"ROUND(APPROX_PERCENTILE(cpu_utilization, 0.99), 2) AS p99_cpu_utilization " +
"FROM " +  DATABASE_NAME + "." +  TABLE_NAME + " " +
"WHERE measure_name = 'metrics' " +
"AND hostname = '" + HOSTNAME + "' " +
"AND time > ago(2h) " +
"GROUP BY region, hostname, az, BIN(time, 15s) " +
"ORDER BY binned_timestamp ASC " +
"LIMIT 5";


private String createScheduledQueryHelper(String topicArn, String roleArn,
        String s3ErrorReportBucketName, String query,
        TargetConfiguration targetConfiguration) {
    System.out.println("Creating Scheduled Query");

    CreateScheduledQueryRequest createScheduledQueryRequest =
 CreateScheduledQueryRequest.builder()
            .name(SQ_NAME)
```

```
            .queryString(query)
            .scheduleConfiguration(ScheduleConfiguration.builder()
                    .scheduleExpression(SCHEDULE_EXPRESSION)
                    .build())
            .notificationConfiguration(NotificationConfiguration.builder()
                    .snsConfiguration(SnsConfiguration.builder()
                            .topicArn(topicArn)
                            .build())
                    .build())
            .targetConfiguration(targetConfiguration)
            .errorReportConfiguration(ErrorReportConfiguration.builder()
                    .s3Configuration(S3Configuration.builder()
                            .bucketName(s3ErrorReportBucketName)
                            .objectKeyPrefix(SCHEDULED_QUERY_EXAMPLE)
                            .build())
                    .build())
            .scheduledQueryExecutionRoleArn(roleArn)
            .build();

    try {
        final CreateScheduledQueryResponse response =
 queryClient.createScheduledQuery(createScheduledQueryRequest);
        final String scheduledQueryArn = response.arn();
        System.out.println("Successfully created scheduled query : " +
 scheduledQueryArn);
        return scheduledQueryArn;
    }
    catch (Exception e) {
        System.out.println("Scheduled Query creation failed: " + e);
        throw e;
    }
}

public String createScheduledQuery(String topicArn, String roleArn,
        String databaseName, String tableName, String s3ErrorReportBucketName) {
    List<Pair<String, MeasureValueType>> sourceColToMeasureValueTypes =
 Arrays.asList(
            Pair.of("avg_cpu_utilization", DOUBLE),
            Pair.of("p90_cpu_utilization", DOUBLE),
            Pair.of("p95_cpu_utilization", DOUBLE),
            Pair.of("p99_cpu_utilization", DOUBLE));

    TargetConfiguration targetConfiguration = TargetConfiguration.builder()
            .timestreamConfiguration(TimestreamConfiguration.builder()
```

```
            .databaseName(databaseName)
            .tableName(tableName)
            .timeColumn("binned_timestamp")
            .dimensionMappings(Arrays.asList(
                    DimensionMapping.builder()
                            .name("region")
                            .dimensionValueType("VARCHAR")
                            .build(),
                    DimensionMapping.builder()
                            .name("az")
                            .dimensionValueType("VARCHAR")
                            .build(),
                    DimensionMapping.builder()
                            .name("hostname")
                            .dimensionValueType("VARCHAR")
                            .build()
            ))
            .multiMeasureMappings(MultiMeasureMappings.builder()
                    .targetMultiMeasureName("multi-metrics")
                    .multiMeasureAttributeMappings(
                            sourceColToMeasureValueTypes.stream()
                                    .map(pair ->
 MultiMeasureAttributeMapping.builder()

 .measureValueType(pair.getValue().name())
                                            .sourceColumn(pair.getKey())
                                            .build())
                                    .collect(Collectors.toList()))
                    .build())
            .build())
            .build();

   return createScheduledQueryHelper(topicArn, roleArn, s3ErrorReportBucketName,
 VALID_QUERY, targetConfiguration);
}}
```

Go

```
SQ_ERROR_CONFIGURATION_S3_BUCKET_NAME_PREFIX = "sq-error-configuration-sample-s3-
bucket-"
HOSTNAME            = "host-24Gju"
SQ_NAME             = "daily-sample"
SCHEDULE_EXPRESSION = "cron(0/1 * * * ? *)"
```

```
QUERY               = "SELECT region, az, hostname, BIN(time, 15s) AS
 binned_timestamp, " +
    "ROUND(AVG(cpu_utilization), 2) AS avg_cpu_utilization, " +
    "ROUND(APPROX_PERCENTILE(cpu_utilization, 0.9), 2) AS p90_cpu_utilization, " +
    "ROUND(APPROX_PERCENTILE(cpu_utilization, 0.95), 2) AS p95_cpu_utilization, " +
    "ROUND(APPROX_PERCENTILE(cpu_utilization, 0.99), 2) AS p99_cpu_utilization " +
    "FROM %s.%s " +
    "WHERE measure_name = 'metrics' " +
    "AND hostname = '" + HOSTNAME + "' " +
    "AND time > ago(2h) " +
    "GROUP BY region, hostname, az, BIN(time, 15s) " +
    "ORDER BY binned_timestamp ASC " +
    "LIMIT 5"
s3BucketName = utils.SQ_ERROR_CONFIGURATION_S3_BUCKET_NAME_PREFIX +
 generateRandomStringWithSize(5)

func generateRandomStringWithSize(size int) string {
     rand.Seed(time.Now().UnixNano())
     alphaNumericList := []rune("abcdefghijklmnopqrstuvwxyz0123456789")
     randomPrefix := make([]rune, size)
     for i := range randomPrefix {
         randomPrefix[i] = alphaNumericList[rand.Intn(len(alphaNumericList))]
     }
     return string(randomPrefix)
 }

func (timestreamBuilder TimestreamBuilder) createScheduledQuery(topicArn string,
 roleArn string, s3ErrorReportBucketName string,
query string, targetConfiguration timestreamquery.TargetConfiguration) (string,
 error) {

createScheduledQueryInput := &timestreamquery.CreateScheduledQueryInput{
    Name:        aws.String(SQ_NAME),
    QueryString: aws.String(query),
    ScheduleConfiguration: &timestreamquery.ScheduleConfiguration{
        ScheduleExpression: aws.String(SCHEDULE_EXPRESSION),
    },
    NotificationConfiguration: &timestreamquery.NotificationConfiguration{
        SnsConfiguration: &timestreamquery.SnsConfiguration{
            TopicArn: aws.String(topicArn),
        },
    },
    TargetConfiguration: &targetConfiguration,
    ErrorReportConfiguration: &timestreamquery.ErrorReportConfiguration{
```

```
        S3Configuration: &timestreamquery.S3Configuration{
            BucketName: aws.String(s3ErrorReportBucketName),
        },
    },
    ScheduledQueryExecutionRoleArn: aws.String(roleArn),
}

createScheduledQueryOutput, err :=
 timestreamBuilder.QuerySvc.CreateScheduledQuery(createScheduledQueryInput)

if err != nil {
    fmt.Printf("Error: %s", err.Error())
} else {
    fmt.Println("createScheduledQueryResult is successful")
    return *createScheduledQueryOutput.Arn, nil
}
return "", err
}

 func (timestreamBuilder TimestreamBuilder) CreateValidScheduledQuery(topicArn
 string, roleArn string, s3ErrorReportBucketName string,
     sqDatabaseName string, sqTableName string, databaseName string, tableName
 string) (string, error) {

    targetConfiguration := timestreamquery.TargetConfiguration{
        TimestreamConfiguration: &timestreamquery.TimestreamConfiguration{
            DatabaseName: aws.String(sqDatabaseName),
            TableName:    aws.String(sqTableName),
            TimeColumn:   aws.String("binned_timestamp"),
            DimensionMappings: []*timestreamquery.DimensionMapping{
                {
                    Name:               aws.String("region"),
                    DimensionValueType: aws.String("VARCHAR"),
                },
                {
                    Name:               aws.String("az"),
                    DimensionValueType: aws.String("VARCHAR"),
                },
                {
                    Name:               aws.String("hostname"),
                    DimensionValueType: aws.String("VARCHAR"),
                },
            },
            MultiMeasureMappings: &timestreamquery.MultiMeasureMappings{
```

```
                TargetMultiMeasureName: aws.String("multi-metrics"),
                MultiMeasureAttributeMappings:
 []*timestreamquery.MultiMeasureAttributeMapping{
                    {
                        SourceColumn:     aws.String("avg_cpu_utilization"),
                        MeasureValueType:
 aws.String(timestreamquery.MeasureValueTypeDouble),
                    },
                    {
                        SourceColumn:     aws.String("p90_cpu_utilization"),
                        MeasureValueType:
 aws.String(timestreamquery.MeasureValueTypeDouble),
                    },
                    {
                        SourceColumn:     aws.String("p95_cpu_utilization"),
                        MeasureValueType:
 aws.String(timestreamquery.MeasureValueTypeDouble),
                    },
                    {
                        SourceColumn:     aws.String("p99_cpu_utilization"),
                        MeasureValueType:
 aws.String(timestreamquery.MeasureValueTypeDouble),
                    },
                },
            },
        },
    }
    return timestreamBuilder.createScheduledQuery(topicArn, roleArn,
 s3ErrorReportBucketName,
        fmt.Sprintf(QUERY, databaseName, tableName), targetConfiguration)
 }
```

Python

```
HOSTNAME = "host-24Gju"
SQ_NAME = "daily-sample"
ERROR_BUCKET_NAME = "scheduledquerysamplerrorbucket" +
 ''.join([choice(ascii_lowercase) for _ in range(5)])
QUERY = \
    "SELECT region, az, hostname, BIN(time, 15s) AS binned_timestamp, " \
    "    ROUND(AVG(cpu_utilization), 2) AS avg_cpu_utilization, " \
    "    ROUND(APPROX_PERCENTILE(cpu_utilization, 0.9), 2) AS p90_cpu_utilization, "
 \
```

```
    "    ROUND(APPROX_PERCENTILE(cpu_utilization, 0.95), 2) AS p95_cpu_utilization,
 " \
    "    ROUND(APPROX_PERCENTILE(cpu_utilization, 0.99), 2) AS p99_cpu_utilization "
 \
    "FROM " + database_name + "." + table_name + " " \
    "WHERE measure_name = 'metrics' " \
    "AND hostname = '" + self.HOSTNAME + "' " \
    "AND time > ago(2h) " \
    "GROUP BY region, hostname, az, BIN(time, 15s) " \
    "ORDER BY binned_timestamp ASC " \
    "LIMIT 5"

def create_scheduled_query_helper(self, topic_arn, role_arn, query,
 target_configuration):
    print("\nCreating Scheduled Query")
    schedule_configuration = {
        'ScheduleExpression': 'cron(0/2 * * * ? *)'
    }
    notification_configuration = {
        'SnsConfiguration': {
            'TopicArn': topic_arn
        }
    }
    error_report_configuration = {
        'S3Configuration': {
            'BucketName': ERROR_BUCKET_NAME
        }
    }

    try:
        create_scheduled_query_response = \
            query_client.create_scheduled_query(Name=self.SQ_NAME,
                 QueryString=query,
                 ScheduleConfiguration=schedule_configuration,
                 NotificationConfiguration=notification_configuration,
                 TargetConfiguration=target_configuration,
                 ScheduledQueryExecutionRoleArn=role_arn,
                 ErrorReportConfiguration=error_report_configuration
                 )
        print("Successfully created scheduled query : ",
 create_scheduled_query_response['Arn'])
        return create_scheduled_query_response['Arn']
    except Exception as err:
        print("Scheduled Query creation failed:", err)
```

```
        raise err

def create_valid_scheduled_query(self, topic_arn, role_arn):
    target_configuration = {
        'TimestreamConfiguration': {
            'DatabaseName': self.sq_database_name,
            'TableName': self.sq_table_name,
            'TimeColumn': 'binned_timestamp',
            'DimensionMappings': [
                {'Name': 'region', 'DimensionValueType': 'VARCHAR'},
                {'Name': 'az', 'DimensionValueType': 'VARCHAR'},
                {'Name': 'hostname', 'DimensionValueType': 'VARCHAR'}
            ],
            'MultiMeasureMappings': {
                'TargetMultiMeasureName': 'target_name',
                'MultiMeasureAttributeMappings': [
                    {'SourceColumn': 'avg_cpu_utilization', 'MeasureValueType':
 'DOUBLE',
                     'TargetMultiMeasureAttributeName': 'avg_cpu_utilization'},
                    {'SourceColumn': 'p90_cpu_utilization', 'MeasureValueType':
 'DOUBLE',
                     'TargetMultiMeasureAttributeName': 'p90_cpu_utilization'},
                    {'SourceColumn': 'p95_cpu_utilization', 'MeasureValueType':
 'DOUBLE',
                     'TargetMultiMeasureAttributeName': 'p95_cpu_utilization'},
                    {'SourceColumn': 'p99_cpu_utilization', 'MeasureValueType':
 'DOUBLE',
                     'TargetMultiMeasureAttributeName': 'p99_cpu_utilization'},
                ]
            }
        }
    }

    return self.create_scheduled_query_helper(topic_arn, role_arn, QUERY,
 target_configuration)
```

Node.js

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada aplikasi sampel Node.js Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
const DATABASE_NAME = 'devops_sample_application';
const TABLE_NAME = 'host_metrics_sample_application';
```

```
const SQ_DATABASE_NAME = 'sq_result_database';
const SQ_TABLE_NAME = 'sq_result_table';
const HOSTNAME = "host-24Gju";
const SQ_NAME = "daily-sample";
const SCHEDULE_EXPRESSION = "cron(0/1 * * * ? *)";

// Find the average, p90, p95, and p99 CPU utilization for a specific EC2 host over
 the past 2 hours.
const VALID_QUERY = "SELECT region, az, hostname, BIN(time, 15s) AS
 binned_timestamp, " +
    " ROUND(AVG(cpu_utilization), 2) AS avg_cpu_utilization, " +
    " ROUND(APPROX_PERCENTILE(cpu_utilization, 0.9), 2) AS p90_cpu_utilization, " +
    " ROUND(APPROX_PERCENTILE(cpu_utilization, 0.95), 2) AS p95_cpu_utilization, " +
    " ROUND(APPROX_PERCENTILE(cpu_utilization, 0.99), 2) AS p99_cpu_utilization " +
    "FROM " + DATABASE_NAME + "." + TABLE_NAME + " " +
    "WHERE measure_name = 'metrics' " +
    " AND hostname = '" + HOSTNAME + "' " +
    " AND time > ago(2h) " +
    "GROUP BY region, hostname, az, BIN(time, 15s) " +
    "ORDER BY binned_timestamp ASC " +
    "LIMIT 5";

async function createScheduledQuery(topicArn, roleArn, s3ErrorReportBucketName) {
    console.log("Creating Valid Scheduled Query");
    const DimensionMappingList = [{
            'Name': 'region',
            'DimensionValueType': 'VARCHAR'
        },
        {
            'Name': 'az',
            'DimensionValueType': 'VARCHAR'
        },
        {
            'Name': 'hostname',
            'DimensionValueType': 'VARCHAR'
        }
    ];

    const MultiMeasureMappings = {
        TargetMultiMeasureName: "multi-metrics",
        MultiMeasureAttributeMappings: [{
                'SourceColumn': 'avg_cpu_utilization',
                'MeasureValueType': 'DOUBLE'
            },
```

```
            {
                'SourceColumn': 'p90_cpu_utilization',
                'MeasureValueType': 'DOUBLE'
            },
            {
                'SourceColumn': 'p95_cpu_utilization',
                'MeasureValueType': 'DOUBLE'
            },
            {
                'SourceColumn': 'p99_cpu_utilization',
                'MeasureValueType': 'DOUBLE'
            },
        ]
    }

    const timestreamConfiguration = {
        DatabaseName: SQ_DATABASE_NAME,
        TableName: SQ_TABLE_NAME,
        TimeColumn: "binned_timestamp",
        DimensionMappings: DimensionMappingList,
        MultiMeasureMappings: MultiMeasureMappings
    }

    const createScheduledQueryRequest = {
        Name: SQ_NAME,
        QueryString: VALID_QUERY,
        ScheduleConfiguration: {
            ScheduleExpression: SCHEDULE_EXPRESSION
        },
        NotificationConfiguration: {
            SnsConfiguration: {
                TopicArn: topicArn
            }
        },
        TargetConfiguration: {
            TimestreamConfiguration: timestreamConfiguration
        },
        ScheduledQueryExecutionRoleArn: roleArn,
        ErrorReportConfiguration: {
            S3Configuration: {
                BucketName: s3ErrorReportBucketName
            }
        }
    };
```

```
    try {
        const data = await
 queryClient.createScheduledQuery(createScheduledQueryRequest).promise();
        console.log("Successfully created scheduled query: " + data.Arn);
        return data.Arn;
    } catch (err) {
        console.log("Scheduled Query creation failed: ", err);
        throw err;
    }
}
```

.NET

```
public const string Hostname = "host-24Gju";
public const string SqName = "timestream-sample";
public const string SqDatabaseName = "sq_result_database";
public const string SqTableName = "sq_result_table";

public const string ErrorConfigurationS3BucketNamePrefix = "error-configuration-
sample-s3-bucket-";
public const string ScheduleExpression = "cron(0/2 * * * ? *)";

// Find the average, p90, p95, and p99 CPU utilization for a specific EC2 host over
 the past 2 hours.
public const string ValidQuery = "SELECT region, az, hostname, BIN(time, 15s) AS
 binned_timestamp, " +
      "ROUND(AVG(cpu_utilization), 2) AS avg_cpu_utilization, " +
      "ROUND(APPROX_PERCENTILE(cpu_utilization, 0.9), 2) AS p90_cpu_utilization, " +
      "ROUND(APPROX_PERCENTILE(cpu_utilization, 0.95), 2) AS p95_cpu_utilization, "
 +
      "ROUND(APPROX_PERCENTILE(cpu_utilization, 0.99), 2) AS p99_cpu_utilization " +
      "FROM " + Constants.DATABASE_NAME + "." + Constants.TABLE_NAME + " " +
      "WHERE measure_name = 'metrics' " +
      "AND hostname = '" + Hostname + "' " +
      "AND time > ago(2h) " +
      "GROUP BY region, hostname, az, BIN(time, 15s) " +
      "ORDER BY binned_timestamp ASC " +
      "LIMIT 5";

private async Task<String> CreateValidScheduledQuery(string topicArn, string
 roleArn,
             string databaseName, string tableName, string s3ErrorReportBucketName)
 {
```

```
        List<MultiMeasureAttributeMapping> sourceColToMeasureValueTypes =
            new List<MultiMeasureAttributeMapping>()
            {
                new()
                {
                    SourceColumn = "avg_cpu_utilization",
                    MeasureValueType = MeasureValueType.DOUBLE.Value
                },
                new()
                {
                    SourceColumn = "p90_cpu_utilization",
                    MeasureValueType = MeasureValueType.DOUBLE.Value
                },
                new()
                {
                    SourceColumn = "p95_cpu_utilization",
                    MeasureValueType = MeasureValueType.DOUBLE.Value
                },
                new()
                {
                    SourceColumn = "p99_cpu_utilization",
                    MeasureValueType = MeasureValueType.DOUBLE.Value
                }
            };

        TargetConfiguration targetConfiguration = new TargetConfiguration()
        {
            TimestreamConfiguration = new TimestreamConfiguration()
            {
                DatabaseName = databaseName,
                TableName = tableName,
                TimeColumn = "binned_timestamp",
                DimensionMappings = new List<DimensionMapping>()
                {
                    new()
                    {
                        Name = "region",
                        DimensionValueType = "VARCHAR"
                    },
                    new()
                    {
                        Name = "az",
                        DimensionValueType = "VARCHAR"
                    },
```

```
                new()
                {
                    Name = "hostname",
                    DimensionValueType = "VARCHAR"
                }
            },
            MultiMeasureMappings = new MultiMeasureMappings()
            {
                TargetMultiMeasureName = "multi-metrics",
                MultiMeasureAttributeMappings = sourceColToMeasureValueTypes
            }
        }
    };
    return await CreateScheduledQuery(topicArn, roleArn, s3ErrorReportBucketName,
        ScheduledQueryConstants.ValidQuery, targetConfiguration);
}

private async Task<String> CreateScheduledQuery(string topicArn, string roleArn,
            string s3ErrorReportBucketName, string query, TargetConfiguration
 targetConfiguration)
{
    try
    {
        Console.WriteLine("Creating Scheduled Query");
        CreateScheduledQueryResponse response = await
 _amazonTimestreamQuery.CreateScheduledQueryAsync(
            new CreateScheduledQueryRequest()
            {
                Name = ScheduledQueryConstants.SqName,
                QueryString = query,
                ScheduleConfiguration = new ScheduleConfiguration()
                {
                    ScheduleExpression = ScheduledQueryConstants.ScheduleExpression
                },
                NotificationConfiguration = new NotificationConfiguration()
                {
                    SnsConfiguration = new SnsConfiguration()
                    {
                        TopicArn = topicArn
                    }
                },
                TargetConfiguration = targetConfiguration,
                ErrorReportConfiguration = new ErrorReportConfiguration()
                {
```

```
                    S3Configuration = new S3Configuration()
                    {
                        BucketName = s3ErrorReportBucketName
                    }
                },
                ScheduledQueryExecutionRoleArn = roleArn
            });
        Console.WriteLine($"Successfully created scheduled query :
{response.Arn}");
        return response.Arn;
    }
    catch (Exception e)
    {
        Console.WriteLine($"Scheduled Query creation failed: {e}");
        throw;
    }
}
```

Daftar kueri terjadwal

Anda dapat menggunakan cuplikan kode berikut untuk membuat daftar kueri terjadwal Anda.

Java

```
public void listScheduledQueries() {
    System.out.println("Listing Scheduled Query");
    try {
        String nextToken = null;
        List<String> scheduledQueries = new ArrayList<>();

        do {
            ListScheduledQueriesResult listScheduledQueriesResult =
                    queryClient.listScheduledQueries(new
 ListScheduledQueriesRequest()
                            .withNextToken(nextToken).withMaxResults(10));
            List<ScheduledQuery> scheduledQueryList =
 listScheduledQueriesResult.getScheduledQueries();

            printScheduledQuery(scheduledQueryList);
            nextToken = listScheduledQueriesResult.getNextToken();
        } while (nextToken != null);
    }
```

```
    catch (Exception e) {
        System.out.println("List Scheduled Query failed: " + e);
        throw e;
    }
}

public void printScheduledQuery(List<ScheduledQuery> scheduledQueryList) {
    for (ScheduledQuery scheduledQuery: scheduledQueryList) {
        System.out.println(scheduledQuery.getArn());
    }
}
```

Java v2

```
public void listScheduledQueries() {
    System.out.println("Listing Scheduled Query");
    try {
        String nextToken = null;

        do {
            ListScheduledQueriesResponse listScheduledQueriesResult =

 queryClient.listScheduledQueries(ListScheduledQueriesRequest.builder()
                            .nextToken(nextToken).maxResults(10)
                            .build());
            List<ScheduledQuery> scheduledQueryList =
 listScheduledQueriesResult.scheduledQueries();

            printScheduledQuery(scheduledQueryList);
            nextToken = listScheduledQueriesResult.nextToken();
        } while (nextToken != null);
    }
    catch (Exception e) {
        System.out.println("List Scheduled Query failed: " + e);
        throw e;
    }
}

public void printScheduledQuery(List<ScheduledQuery> scheduledQueryList) {
    for (ScheduledQuery scheduledQuery: scheduledQueryList) {
        System.out.println(scheduledQuery.arn());
    }
}
```

Go

```
func (timestreamBuilder TimestreamBuilder) ListScheduledQueries()
 ([]*timestreamquery.ScheduledQuery, error) {

    var nextToken *string = nil
    var scheduledQueries []*timestreamquery.ScheduledQuery
    for ok := true; ok; ok = nextToken != nil {
        listScheduledQueriesInput := &timestreamquery.ListScheduledQueriesInput{
            MaxResults: aws.Int64(15),
        }
        if nextToken != nil {
            listScheduledQueriesInput.NextToken = aws.String(*nextToken)
        }

        listScheduledQueriesOutput, err :=
 timestreamBuilder.QuerySvc.ListScheduledQueries(listScheduledQueriesInput)
        if err != nil {
            fmt.Printf("Error: %s", err.Error())
            return nil, err
        }
        scheduledQueries = append(scheduledQueries,
 listScheduledQueriesOutput.ScheduledQueries...)
        nextToken = listScheduledQueriesOutput.NextToken
    }
    return scheduledQueries, nil
 }
```

Python

```
def list_scheduled_queries(self):
    print("\nListing Scheduled Queries")
    try:
        response = self.query_client.list_scheduled_queries(MaxResults=10)
        self.print_scheduled_queries(response['ScheduledQueries'])
        next_token = response.get('NextToken', None)
        while next_token:
            response =
 self.query_client.list_scheduled_queries(NextToken=next_token, MaxResults=10)
            self.print_scheduled_queries(response['ScheduledQueries'])
            next_token = response.get('NextToken', None)
    except Exception as err:
        print("List scheduled queries failed:", err)
```

```
        raise err

@staticmethod
def print_scheduled_queries(scheduled_queries):
    for scheduled_query in scheduled_queries:
        print(scheduled_query['Arn'])
```

Node.js

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada aplikasi sampel Node.js Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
async function listScheduledQueries() {
     console.log("Listing Scheduled Query");
     try {
         var nextToken = null;
         do {
             var params = {
                 MaxResults: 10,
                 NextToken: nextToken
             }
             var data = await queryClient.listScheduledQueries(params).promise();
             var scheduledQueryList = data.ScheduledQueries;
             printScheduledQuery(scheduledQueryList);
             nextToken = data.NextToken;
         }
         while (nextToken != null);
     }  catch (err) {
         console.log("List Scheduled Query failed: ", err);
         throw err;
     }
 }

 async function printScheduledQuery(scheduledQueryList) {
     scheduledQueryList.forEach(element => console.log(element.Arn));
 }
```

.NET

```
private async Task ListScheduledQueries()
 {
     try
     {
```

```
        Console.WriteLine("Listing Scheduled Query");
        string nextToken;
        do
        {
            ListScheduledQueriesResponse response =
                await _amazonTimestreamQuery.ListScheduledQueriesAsync(new
 ListScheduledQueriesRequest());
            foreach (var scheduledQuery in response.ScheduledQueries)
            {
                Console.WriteLine($"{scheduledQuery.Arn}");
            }

            nextToken = response.NextToken;
        } while (nextToken != null);
    }
    catch (Exception e)
    {
        Console.WriteLine($"List Scheduled Query failed: {e}");
        throw;
    }
}
```

Jelaskan kueri terjadwal

Anda dapat menggunakan cuplikan kode berikut untuk menggambarkan kueri terjadwal.

Java

```
public void describeScheduledQueries(String scheduledQueryArn) {
    System.out.println("Describing Scheduled Query");
    try {
        DescribeScheduledQueryResult describeScheduledQueryResult =
 queryClient.describeScheduledQuery(new
 DescribeScheduledQueryRequest().withScheduledQueryArn(scheduledQueryArn));
        System.out.println(describeScheduledQueryResult);
    }
    catch (ResourceNotFoundException e) {
        System.out.println("Scheduled Query doesn't exist");
        throw e;
    }
    catch (Exception e) {
        System.out.println("Describe Scheduled Query failed: " + e);
```

```
        throw e;
    }
}
```

Java v2

```
public void describeScheduledQueries(String scheduledQueryArn) {
    System.out.println("Describing Scheduled Query");
    try {
        DescribeScheduledQueryResponse describeScheduledQueryResult =

 queryClient.describeScheduledQuery(DescribeScheduledQueryRequest.builder()
                        .scheduledQueryArn(scheduledQueryArn)
                        .build());
        System.out.println(describeScheduledQueryResult);
    }
    catch (ResourceNotFoundException e) {
        System.out.println("Scheduled Query doesn't exist");
        throw e;
    }
    catch (Exception e) {
        System.out.println("Describe Scheduled Query failed: " + e);
        throw e;
    }
}
```

Go

```
func (timestreamBuilder TimestreamBuilder) DescribeScheduledQuery(scheduledQueryArn
 string) error {

     describeScheduledQueryInput := &timestreamquery.DescribeScheduledQueryInput{
         ScheduledQueryArn: aws.String(scheduledQueryArn),
     }
     describeScheduledQueryOutput, err :=
 timestreamBuilder.QuerySvc.DescribeScheduledQuery(describeScheduledQueryInput)

     if err != nil {
         if aerr, ok := err.(awserr.Error); ok {
             switch aerr.Code() {
             case timestreamquery.ErrCodeResourceNotFoundException:
                 fmt.Println(timestreamquery.ErrCodeResourceNotFoundException,
 aerr.Error())
```

```
            default:
                fmt.Printf("Error: %s", err.Error())
            }
        } else {
            fmt.Printf("Error: %s", aerr.Error())
        }
        return err
    } else {
        fmt.Println("DescribeScheduledQuery is successful, below is the output:")
        fmt.Println(describeScheduledQueryOutput.ScheduledQuery)
        return nil
    }
}
```

Python

```
def describe_scheduled_query(self, scheduled_query_arn):
    print("\nDescribing Scheduled Query")
    try:
        response =
 self.query_client.describe_scheduled_query(ScheduledQueryArn=scheduled_query_arn)
        if 'ScheduledQuery' in response:
            response = response['ScheduledQuery']
            for key in response:
                print("{} :{}".format(key, response[key]))
    except self.query_client.exceptions.ResourceNotFoundException as err:
        print("Scheduled Query doesn't exist")
        raise err
    except Exception as err:
        print("Scheduled Query describe failed:", err)
        raise err
```

Node.js

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada aplikasi sampel Node.js Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
async function describeScheduledQuery(scheduledQueryArn) {
    console.log("Describing Scheduled Query");
    var params = {
        ScheduledQueryArn: scheduledQueryArn
    }
    try {
```

```
        const data = await queryClient.describeScheduledQuery(params).promise();
        console.log(data.ScheduledQuery);
    } catch (err) {
        console.log("Describe Scheduled Query failed: ", err);
        throw err;
    }
}
```

.NET

```
private async Task DescribeScheduledQuery(string scheduledQueryArn)
{
    try
    {
        Console.WriteLine("Describing Scheduled Query");
        DescribeScheduledQueryResponse response = await
_amazonTimestreamQuery.DescribeScheduledQueryAsync(
            new DescribeScheduledQueryRequest()
            {
                ScheduledQueryArn = scheduledQueryArn
            });

Console.WriteLine($"{JsonConvert.SerializeObject(response.ScheduledQuery)}");
    }
    catch (ResourceNotFoundException e)
    {
        Console.WriteLine($"Scheduled Query doesn't exist: {e}");
        throw;
    }
    catch (Exception e)
    {
        Console.WriteLine($"Describe Scheduled Query failed: {e}");
        throw;
    }
}
```

Jalankan kueri terjadwal

Anda dapat menggunakan cuplikan kode berikut untuk menjalankan kueri terjadwal.

Java

```
public void executeScheduledQueries(String scheduledQueryArn, Date invocationTime) {
    System.out.println("Executing Scheduled Query");
    try {
        ExecuteScheduledQueryResult executeScheduledQueryResult =
 queryClient.executeScheduledQuery(new ExecuteScheduledQueryRequest()
                .withScheduledQueryArn(scheduledQueryArn)
                .withInvocationTime(invocationTime)
        );

    }
    catch (ResourceNotFoundException e) {
        System.out.println("Scheduled Query doesn't exist");
        throw e;
    }
    catch (Exception e) {
        System.out.println("Execution Scheduled Query failed: " + e);
        throw e;
    }
}
```

Java v2

```
public void executeScheduledQuery(String scheduledQueryArn) {
    System.out.println("Executing Scheduled Query");
    try {
        ExecuteScheduledQueryResponse executeScheduledQueryResult =
 queryClient.executeScheduledQuery(ExecuteScheduledQueryRequest.builder()
                .scheduledQueryArn(scheduledQueryArn)
                .invocationTime(Instant.now())
                .build()
        );

        System.out.println("Execute ScheduledQuery response code: " +
 executeScheduledQueryResult.sdkHttpResponse().statusCode());

    }
    catch (ResourceNotFoundException e) {
        System.out.println("Scheduled Query doesn't exist");
        throw e;
    }
```

```
    catch (Exception e) {
        System.out.println("Execution Scheduled Query failed: " + e);
        throw e;
    }
}
```

Go

```
func (timestreamBuilder TimestreamBuilder) ExecuteScheduledQuery(scheduledQueryArn
 string, invocationTime time.Time) error {

    executeScheduledQueryInput := &timestreamquery.ExecuteScheduledQueryInput{
        ScheduledQueryArn: aws.String(scheduledQueryArn),
        InvocationTime:    aws.Time(invocationTime),
    }
    executeScheduledQueryOutput, err :=
 timestreamBuilder.QuerySvc.ExecuteScheduledQuery(executeScheduledQueryInput)

    if err != nil {
        if aerr, ok := err.(awserr.Error); ok {
            switch aerr.Code() {
            case timestreamquery.ErrCodeResourceNotFoundException:
                fmt.Println(timestreamquery.ErrCodeResourceNotFoundException,
 aerr.Error())
            default:
                fmt.Printf("Error: %s", aerr.Error())
            }
        } else {
            fmt.Printf("Error: %s", err.Error())
        }
        return err
    } else {
        fmt.Println("ExecuteScheduledQuery is successful, below is the output:")
        fmt.Println(executeScheduledQueryOutput.GoString())
        return nil
    }
}
```

Python

```
def execute_scheduled_query(self, scheduled_query_arn, invocation_time):
    print("\nExecuting Scheduled Query")
    try:
```

```
 self.query_client.execute_scheduled_query(ScheduledQueryArn=scheduled_query_arn,
 InvocationTime=invocation_time)
        print("Successfully started executing scheduled query")
    except self.query_client.exceptions.ResourceNotFoundException as err:
        print("Scheduled Query doesn't exist")
        raise err
    except Exception as err:
        print("Scheduled Query execution failed:", err)
        raise err
```

Node.js

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada contoh aplikasi di Node.js sampel Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
async function executeScheduledQuery(scheduledQueryArn, invocationTime) {
     console.log("Executing Scheduled Query");
     var params = {
         ScheduledQueryArn: scheduledQueryArn,
         InvocationTime: invocationTime
     }
     try {
         await queryClient.executeScheduledQuery(params).promise();
     } catch (err) {
         console.log("Execute Scheduled Query failed: ", err);
         throw err;
     }
 }
```

.NET

```
private async Task ExecuteScheduledQuery(string scheduledQueryArn, DateTime
 invocationTime)
 {
     try
     {
         Console.WriteLine("Running Scheduled Query");
         await _amazonTimestreamQuery.ExecuteScheduledQueryAsync(new
 ExecuteScheduledQueryRequest()
         {
             ScheduledQueryArn = scheduledQueryArn,
             InvocationTime = invocationTime
```

```
            });
            Console.WriteLine("Successfully started manual run of scheduled query");
        }
        catch (ResourceNotFoundException e)
        {
            Console.WriteLine($"Scheduled Query doesn't exist: {e}");
            throw;
        }
        catch (Exception e)
        {
            Console.WriteLine($"Execute Scheduled Query failed: {e}");
            throw;
        }
    }
```

Perbarui kueri terjadwal

Anda dapat menggunakan cuplikan kode berikut untuk memperbarui kueri terjadwal.

Java

```
public void updateScheduledQueries(String scheduledQueryArn) {
    System.out.println("Updating Scheduled Query");
    try {
        queryClient.updateScheduledQuery(new UpdateScheduledQueryRequest()
                .withScheduledQueryArn(scheduledQueryArn)
                .withState(ScheduledQueryState.DISABLED));
        System.out.println("Successfully update scheduled query state");
    }
    catch (ResourceNotFoundException e) {
        System.out.println("Scheduled Query doesn't exist");
        throw e;
    }
    catch (Exception e) {
        System.out.println("Execution Scheduled Query failed: " + e);
        throw e;
    }
}
```

Java v2

```
public void updateScheduledQuery(String scheduledQueryArn, ScheduledQueryState
 state) {
    System.out.println("Updating Scheduled Query");
    try {
        queryClient.updateScheduledQuery(UpdateScheduledQueryRequest.builder()
                .scheduledQueryArn(scheduledQueryArn)
                .state(state)
                .build());
        System.out.println("Successfully update scheduled query state");
    }
    catch (ResourceNotFoundException e) {
        System.out.println("Scheduled Query doesn't exist");
        throw e;
    }
    catch (Exception e) {
        System.out.println("Execution Scheduled Query failed: " + e);
        throw e;
    }
}
```

Go

```
func (timestreamBuilder TimestreamBuilder) UpdateScheduledQuery(scheduledQueryArn
 string) error {

    updateScheduledQueryInput := &timestreamquery.UpdateScheduledQueryInput{
        ScheduledQueryArn: aws.String(scheduledQueryArn),
        State:             aws.String(timestreamquery.ScheduledQueryStateDisabled),
    }
    _, err :=
 timestreamBuilder.QuerySvc.UpdateScheduledQuery(updateScheduledQueryInput)

    if err != nil {
        if aerr, ok := err.(awserr.Error); ok {
            switch aerr.Code() {
            case timestreamquery.ErrCodeResourceNotFoundException:
                fmt.Println(timestreamquery.ErrCodeResourceNotFoundException,
 aerr.Error())
            default:
                fmt.Printf("Error: %s", aerr.Error())
            }
```

```
        } else {
            fmt.Printf("Error: %s", err.Error())
        }
        return err
    } else {
        fmt.Println("UpdateScheduledQuery is successful")
        return nil
    }
}
```

Python

```
def update_scheduled_query(self, scheduled_query_arn, state):
    print("\nUpdating Scheduled Query")
    try:

 self.query_client.update_scheduled_query(ScheduledQueryArn=scheduled_query_arn,
                                                  State=state)
        print("Successfully update scheduled query state to", state)
    except self.query_client.exceptions.ResourceNotFoundException as err:
        print("Scheduled Query doesn't exist")
        raise err
    except Exception as err:
        print("Scheduled Query deletion failed:", err)
        raise err
```

Node.js

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada aplikasi sampel Node.js Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
async function updateScheduledQueries(scheduledQueryArn) {
    console.log("Updating Scheduled Query");
    var params = {
        ScheduledQueryArn: scheduledQueryArn,
        State: "DISABLED"
    }
    try {
        await queryClient.updateScheduledQuery(params).promise();
        console.log("Successfully update scheduled query state");
    } catch (err) {
        console.log("Update Scheduled Query failed: ", err);
        throw err;
```

```
    }
}
```

.NET

```
private async Task UpdateScheduledQuery(string scheduledQueryArn,
 ScheduledQueryState state)
 {
     try
     {
         Console.WriteLine("Updating Scheduled Query");
         await _amazonTimestreamQuery.UpdateScheduledQueryAsync(new
 UpdateScheduledQueryRequest()
         {
             ScheduledQueryArn = scheduledQueryArn,
             State = state
         });
         Console.WriteLine("Successfully update scheduled query state");
     }
     catch (ResourceNotFoundException e)
     {
         Console.WriteLine($"Scheduled Query doesn't exist: {e}");
         throw;
     }
     catch (Exception e)
     {
         Console.WriteLine($"Update Scheduled Query failed: {e}");
         throw;
     }
 }
```

Hapus kueri terjadwal

Anda dapat menggunakan cuplikan kode berikut untuk menghapus kueri terjadwal.

Java

```
public void deleteScheduledQuery(String scheduledQueryArn) {
    System.out.println("Deleting Scheduled Query");

    try {
```

```
        queryClient.deleteScheduledQuery(new
 DeleteScheduledQueryRequest().withScheduledQueryArn(scheduledQueryArn));
        System.out.println("Successfully deleted scheduled query");
    }
    catch (Exception e) {
        System.out.println("Scheduled Query deletion failed: " + e);
    }
}
```

Java v2

```
public void deleteScheduledQuery(String scheduledQueryArn) {
    System.out.println("Deleting Scheduled Query");

    try {
        queryClient.deleteScheduledQuery(DeleteScheduledQueryRequest.builder()
                .scheduledQueryArn(scheduledQueryArn).build());
        System.out.println("Successfully deleted scheduled query");
    }
    catch (Exception e) {
        System.out.println("Scheduled Query deletion failed: " + e);
    }
}
```

Go

```
func (timestreamBuilder TimestreamBuilder) DeleteScheduledQuery(scheduledQueryArn
 string) error {

    deleteScheduledQueryInput := &timestreamquery.DeleteScheduledQueryInput{
        ScheduledQueryArn: aws.String(scheduledQueryArn),
    }
    _, err :=
 timestreamBuilder.QuerySvc.DeleteScheduledQuery(deleteScheduledQueryInput)

    if err != nil {
        fmt.Println("Error:")
        if aerr, ok := err.(awserr.Error); ok {
            switch aerr.Code() {
            case timestreamquery.ErrCodeResourceNotFoundException:
                fmt.Println(timestreamquery.ErrCodeResourceNotFoundException,
 aerr.Error())
            default:
```

```
                fmt.Printf("Error: %s", aerr.Error())
            }
        } else {
            fmt.Printf("Error: %s", err.Error())
        }
        return err
    } else {
        fmt.Println("DeleteScheduledQuery is successful")
        return nil
    }
}
```

Python

```
def delete_scheduled_query(self, scheduled_query_arn):
    print("\nDeleting Scheduled Query")
    try:

 self.query_client.delete_scheduled_query(ScheduledQueryArn=scheduled_query_arn)
        print("Successfully deleted scheduled query :", scheduled_query_arn)
    except Exception as err:
        print("Scheduled Query deletion failed:", err)
        raise err
```

Node.js

Cuplikan berikut menggunakan gaya AWS SDK for JavaScript V2. Hal ini didasarkan pada contoh aplikasi di Node.js sampel Amazon Timestream untuk LiveAnalytics aplikasi pada. GitHub

```
async function deleteScheduleQuery(scheduledQueryArn) {
    console.log("Deleting Scheduled Query");
    const params = {
        ScheduledQueryArn: scheduledQueryArn
    }
    try {
        await queryClient.deleteScheduledQuery(params).promise();
        console.log("Successfully deleted scheduled query");
    } catch (err) {
        console.log("Scheduled Query deletion failed: ", err);
    }
}
```

.NET

```
private async Task DeleteScheduledQuery(string scheduledQueryArn)
 {
     try
     {
         Console.WriteLine("Deleting Scheduled Query");
         await _amazonTimestreamQuery.DeleteScheduledQueryAsync(new
 DeleteScheduledQueryRequest()
         {
             ScheduledQueryArn = scheduledQueryArn
         });
         Console.WriteLine($"Successfully deleted scheduled query :
 {scheduledQueryArn}");
     }
     catch (Exception e)
     {
         Console.WriteLine($"Scheduled Query deletion failed: {e}");
         throw;
     }
 }
```

Menggunakan pemuatan batch di Timestream untuk LiveAnalytics

Dengan pemuatan batch untuk Amazon Timestream untuk LiveAnalytics, Anda dapat menelan CSV file yang disimpan di Amazon S3 ke Timestream dalam batch. Dengan fungsionalitas baru ini, Anda dapat memiliki data Anda di Timestream LiveAnalytics tanpa harus bergantung pada alat lain atau menulis kode khusus. Anda dapat menggunakan pemuatan batch untuk mengisi ulang data dengan waktu tunggu yang fleksibel, seperti data yang tidak segera diperlukan untuk kueri atau analisis.

Anda dapat membuat tugas pemuatan batch dengan menggunakan AWS Management Console, AWS CLI, dan file AWS SDKs. Lihat informasi selengkapnya di Menggunakan pemuatan batch dengan konsol, Menggunakan beban batch dengan AWS CLI, dan Menggunakan beban batch dengan AWS SDKs.

Selain pemuatan batch, Anda dapat menulis beberapa catatan bersamaan dengan WriteRecords API operasi. Untuk panduan tentang mana yang harus digunakan, lihatMemilih antara WriteRecords API operasi dan beban batch.

Topik

Konsep pemuatan Batch di Timestream

Tinjau konsep berikut untuk lebih memahami fungsionalitas pemuatan batch.

Tugas pemuatan Batch — Tugas yang menentukan data sumber dan tujuan Anda di Amazon Timestream. Anda menentukan konfigurasi tambahan seperti model data saat Anda membuat tugas pemuatan batch. Anda dapat membuat tugas pemuatan batch melalui AWS Management Console, AWS CLI, dan file AWS SDKs.

Impor tujuan - Database tujuan dan tabel di Timestream. Untuk informasi tentang membuat database dan tabel, lihat Buat database danMembuat tabel.

Sumber data — CSV File sumber yang disimpan dalam ember S3. Untuk informasi tentang menyiapkan file data, lihatMempersiapkan file data pemuatan batch. Untuk informasi tentang harga S3, lihat harga Amazon S3.

Laporan kesalahan pemuatan Batch — Laporan yang menyimpan informasi tentang kesalahan tugas pemuatan batch. Anda menentukan lokasi S3 untuk laporan kesalahan pemuatan batch sebagai bagian dari tugas pemuatan batch. Untuk informasi tentang informasi dalam laporan, lihatMenggunakan laporan kesalahan pemuatan batch.

Pemetaan model data — Pemetaan beban batch untuk waktu, dimensi, dan ukuran yang berasal dari sumber data di lokasi S3 ke Timestream target untuk tabel. LiveAnalytics Untuk informasi selengkapnya, lihat Pemetaan model data untuk pemuatan batch.

Prasyarat pemuatan Batch

Ini adalah daftar prasyarat untuk menggunakan pemuatan batch. Untuk praktik terbaik, lihat Praktik terbaik pemuatan batch.

- Data sumber pemuatan batch disimpan di Amazon S3 dalam CSV format dengan header.
- Untuk setiap bucket sumber Amazon S3, Anda harus memiliki izin berikut dalam kebijakan terlampir:

```
"s3:GetObject",
"s3:GetBucketAcl"
"s3:ListBucket"
```

Demikian pula, untuk setiap bucket keluaran Amazon S3 tempat laporan ditulis, Anda harus memiliki izin berikut dalam kebijakan terlampir:

```
"s3:PutObject",
"s3:GetBucketAcl"
```

Sebagai contoh:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:GetObject",
        "s3:GetBucketAcl"
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::inputs-source-bucket-name-A”
        "arn:aws:s3:::inputs-source-bucket-name-B”
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "s3:PutObject",
        "s3:GetBucketAcl"
      ],
```

```
      "Resource": [
        "arn:aws:s3:::reports-output-bucket-name”
      ]
      "Effect": "Allow"
    }
  ]
}
```

- Timestream untuk LiveAnalytics CSV mem-parsing dengan memetakan informasi yang disediakan dalam model data ke header. CSV Data harus memiliki kolom yang mewakili stempel waktu, setidaknya satu kolom dimensi, dan setidaknya satu kolom ukuran.
- Bucket S3 yang digunakan dengan beban batch harus berada di wilayah yang sama dan dari akun yang sama dengan Timestream untuk LiveAnalytics tabel yang digunakan dalam pemuatan batch.
- `timestamp`Kolom harus berupa tipe data panjang yang mewakili waktu sejak zaman Unix. Misalnya, stempel waktu `2021-03-25T08:45:21Z` akan direpresentasikan sebagai. `1616661921` Timestream mendukung detik, milidetik, mikrodetik, dan nanodetik untuk presisi stempel waktu. Saat menggunakan bahasa kueri, Anda dapat mengonversi antara format dengan fungsi seperti`to_unixtime`. Untuk informasi selengkapnya, lihat Fungsi tanggal/waktu.
- Timestream mendukung tipe data string untuk nilai dimensi. Ini mendukung tipe data panjang, ganda, string, dan boolean untuk mengukur kolom.

Untuk batas pemuatan batch dan kuota, lihatBeban batch.

Praktik terbaik pemuatan batch

Batch load bekerja paling baik (throughput tinggi) saat mengikuti kondisi dan rekomendasi berikut:

1. CSVfile yang dikirimkan untuk konsumsi berukuran kecil, khususnya dengan ukuran file 100MB-1GB, untuk meningkatkan paralelisme dan kecepatan konsumsi.
2. Hindari menelan data secara bersamaan ke dalam tabel yang sama (misalnya menggunakan WriteRecords API operasi, atau kueri terjadwal) saat pemuatan batch sedang berlangsung. Ini mungkin menyebabkan throttle, dan tugas pemuatan batch akan gagal.
3. Jangan menambah, memodifikasi, atau menghapus file dari bucket S3 yang digunakan dalam pemuatan batch saat tugas pemuatan batch sedang berjalan.
4. Jangan menghapus atau mencabut izin dari tabel atau sumber, atau laporkan bucket S3 yang memiliki tugas pemuatan batch terjadwal atau sedang berlangsung.

5. Saat menelan data dengan serangkaian nilai dimensi kardinalitas tinggi, ikuti panduan di. Rekomendasi untuk mempartisi catatan multi-ukuran
6. Pastikan Anda menguji data untuk kebenaran dengan mengirimkan file kecil. Anda akan dikenakan biaya untuk setiap data yang dikirimkan ke pemuatan batch terlepas dari kebenarannya. Untuk informasi selengkapnya tentang harga, lihat harga Amazon Timestream.
7. Jangan melanjutkan tugas pemuatan batch kecuali `ActiveMagneticStorePartitions` di bawah 250. Pekerjaan mungkin terhambat dan gagal. Mengirimkan beberapa pekerjaan pada saat yang sama untuk database yang sama akan mengurangi jumlahnya.

Berikut ini adalah praktik terbaik konsol:

1. Gunakan pembuat hanya untuk pemodelan data sederhana yang hanya menggunakan satu nama ukuran untuk catatan multi-ukuran.
2. Untuk pemodelan data yang lebih kompleks, gunakanJSON. Misalnya, gunakan JSON saat Anda menggunakan beberapa nama ukuran saat menggunakan catatan multi-ukuran.

Untuk Timestream tambahan untuk praktik LiveAnalytics terbaik, lihatPraktik terbaik.

Mempersiapkan file data pemuatan batch

File data sumber memiliki nilai yang dipisahkan pembatas. Istilah yang lebih spesifik, nilai yang dipisahkan koma (CSV) digunakan secara umum. Pemisah kolom yang valid termasuk koma dan pipa. Catatan dipisahkan oleh baris baru. File harus disimpan di Amazon S3. Saat Anda membuat tugas pemuatan batch baru, lokasi data sumber ditentukan oleh ARN untuk file tersebut. Sebuah file berisi header. Satu kolom mewakili stempel waktu. Setidaknya satu kolom lain mewakili ukuran.

Bucket S3 yang digunakan dengan beban batch harus berada di Wilayah yang sama dengan Timestream untuk LiveAnalytics tabel yang digunakan dalam pemuatan batch. Jangan menambah atau menghapus file dari bucket S3 yang digunakan dalam pemuatan batch setelah tugas pemuatan batch dikirimkan. Untuk informasi tentang bekerja dengan bucket S3, lihat Memulai Amazon S3.

> **Note**
>
> CSVfile yang dihasilkan oleh beberapa aplikasi seperti Excel mungkin berisi tanda urutan byte (BOM) yang bertentangan dengan pengkodean yang diharapkan. Timestream untuk tugas pemuatan LiveAnalytics batch yang mereferensikan CSV file dengan kesalahan saat

diproses secara terprogram. BOM Untuk menghindari hal ini, Anda dapat menghapusBOM, yang merupakan karakter yang tidak terlihat.
Misalnya, Anda dapat menyimpan file dari aplikasi seperti Notepad ++ yang memungkinkan Anda menentukan pengkodean baru. Anda juga dapat menggunakan opsi terprogram yang membaca baris pertama, menghapus karakter dari baris, dan menulis nilai baru di atas baris pertama dalam file.
Saat menyimpan dari Excel, ada beberapa CSV opsi. Menyimpan dengan CSV opsi yang berbeda dapat mencegah masalah yang dijelaskan. Tetapi Anda harus memeriksa hasilnya karena perubahan pengkodean dapat memengaruhi beberapa karakter.

CSVparameter format

Anda menggunakan karakter escape ketika Anda mewakili nilai yang dinyatakan dicadangkan oleh parameter format. Misalnya, jika karakter kutipan adalah kutipan ganda, untuk mewakili kutipan ganda dalam data, tempatkan karakter escape sebelum kutipan ganda.

Untuk informasi tentang kapan harus menentukan ini saat membuat tugas pemuatan batch, lihatBuat tugas pemuatan batch.

Parameter	Opsi
Pemisah kolom	(Koma (',') | Pipa ('|') | Titik koma (';') | Tab ('/t') | Ruang kosong (''))
Karakter melarikan diri	none
Karakter kutipan	Konsol: (Kutipan ganda (“) | Kutipan tunggal ('))
Nilai nol	Ruang kosong ('')
Potong ruang putih	Konsol: (Tidak | Ya)

Pemetaan model data untuk pemuatan batch

Berikut ini membahas skema untuk pemetaan model data dan memberi dan contoh.

Skema pemetaan model data

Sintaks `CreateBatchLoadTask` permintaan dan `BatchLoadTaskDescription` objek dikembalikan oleh panggilan untuk `DescribeBatchLoadTask` menyertakan `DataModelConfiguration` objek yang menyertakan `DataModel` untuk pemuatan batch. `DataModel`Mendefinisikan pemetaan dari data sumber yang disimpan dalam CSV format di lokasi S3 ke Timestream target untuk database dan tabel. LiveAnalytics

`TimeColumn`Bidang menunjukkan lokasi data sumber untuk nilai yang akan dipetakan ke `time` kolom tabel tujuan di Timestream untuk. LiveAnalytics `TimeUnit`Menentukan unit untuk`TimeColumn`, dan dapat menjadi salah satu dari`MILLISECONDS`,, `SECONDSMICROSECONDS`, atau`NANOSECONDS`. Ada juga pemetaan untuk dimensi dan ukuran. Pemetaan dimensi terdiri dari kolom sumber dan bidang target.

Untuk informasi lebih lanjut, lihat DimensionMapping. Pemetaan untuk tindakan memiliki dua opsi, `MixedMeasureMappings` dan. `MultiMeasureMappings`

Untuk meringkas, a `DataModel` berisi pemetaan dari sumber data di lokasi S3 ke Timestream target untuk tabel berikut ini. LiveAnalytics

- Waktu
- Dimensi
- Tindakan

Jika memungkinkan, kami menyarankan Anda memetakan data pengukuran ke catatan multi-ukuran di Timestream untuk. LiveAnalytics Untuk informasi tentang manfaat catatan multi-ukuran, lihatCatatan multi-ukuran.

Jika beberapa ukuran dalam data sumber disimpan dalam satu baris, Anda dapat memetakan beberapa ukuran tersebut ke catatan multi-ukuran di Timestream untuk LiveAnalytics digunakan. `MultiMeasureMappings` Jika ada nilai yang harus dipetakan ke catatan ukuran tunggal, Anda dapat menggunakannya`MixedMeasureMappings`.

`MixedMeasureMappings`dan `MultiMeasureMappings` keduanya termasuk`MultiMeasureAttributeMappings`. Catatan multi-ukuran didukung terlepas dari apakah catatan ukuran tunggal diperlukan.

Jika hanya catatan target multi-ukuran yang diperlukan di Timestream for LiveAnalytics, Anda dapat menentukan pemetaan ukuran dalam struktur berikut.

```
CreateBatchLoadTask
    MeasureNameColumn
    MultiMeasureMappings
        TargetMultiMeasureName
        MultiMeasureAttributeMappings array
```

> **Note**
>
> Kami merekomendasikan penggunaan `MultiMeasureMappings` bila memungkinkan.

Jika catatan target ukuran tunggal diperlukan di Timestream for LiveAnalytics, Anda dapat menentukan pemetaan ukuran dalam struktur berikut.

```
CreateBatchLoadTask
    MeasureNameColumn
    MixedMeasureMappings array
        MixedMeasureMapping
            MeasureName
            MeasureValueType
            SourceColumn
            TargetMeasureName
            MultiMeasureAttributeMappings array
```

Saat Anda menggunakan`MultiMeasureMappings`, `MultiMeasureAttributeMappings` array selalu diperlukan. Bila Anda menggunakan `MixedMeasureMappings` array, jika `MeasureValueType` adalah `MULTI` untuk diberikan`MixedMeasureMapping`, `MultiMeasureAttributeMappings` diperlukan untuk itu`MixedMeasureMapping`. Jika tidak, `MeasureValueType` tunjukkan jenis ukuran untuk catatan ukuran tunggal.

Either way, ada array yang `MultiMeasureAttributeMapping` tersedia. Anda menentukan pemetaan ke catatan multi-ukuran di masing-masing `MultiMeasureAttributeMapping` sebagai berikut:

`SourceColumn`

Kolom dalam data sumber yang terletak di Amazon S3.

`TargetMultiMeasureAttributeName`

Nama nama multi-ukuran target di tabel tujuan. Input ini `MeasureNameColumn` diperlukan ketika tidak disediakan. Jika `MeasureNameColumn` disediakan, nilai dari kolom itu digunakan sebagai nama multi-ukuran.

`MeasureValueType`

Salah satu`DOUBLE`, `BIGINT``BOOLEAN`,`VARCHAR`, atau`TIMESTAMP`.

Pemetaan model data dengan contoh **`MultiMeasureMappings`**

Contoh ini menunjukkan pemetaan ke catatan multi-ukuran, pendekatan yang disukai, yang menyimpan setiap nilai ukuran dalam kolom khusus. Anda dapat mengunduh sampel CSV di sampel CSV. Sampel memiliki judul berikut untuk dipetakan ke kolom target dalam Timestream untuk LiveAnalytics tabel.

- `time`
- `measure_name`
- `region`
- `location`
- `hostname`
- `memory_utilization`
- `cpu_utilization`

Identifikasi `time` dan `measure_name` kolom dalam CSV file. Dalam hal ini peta ini langsung ke Timestream untuk kolom LiveAnalytics tabel dengan nama yang sama.

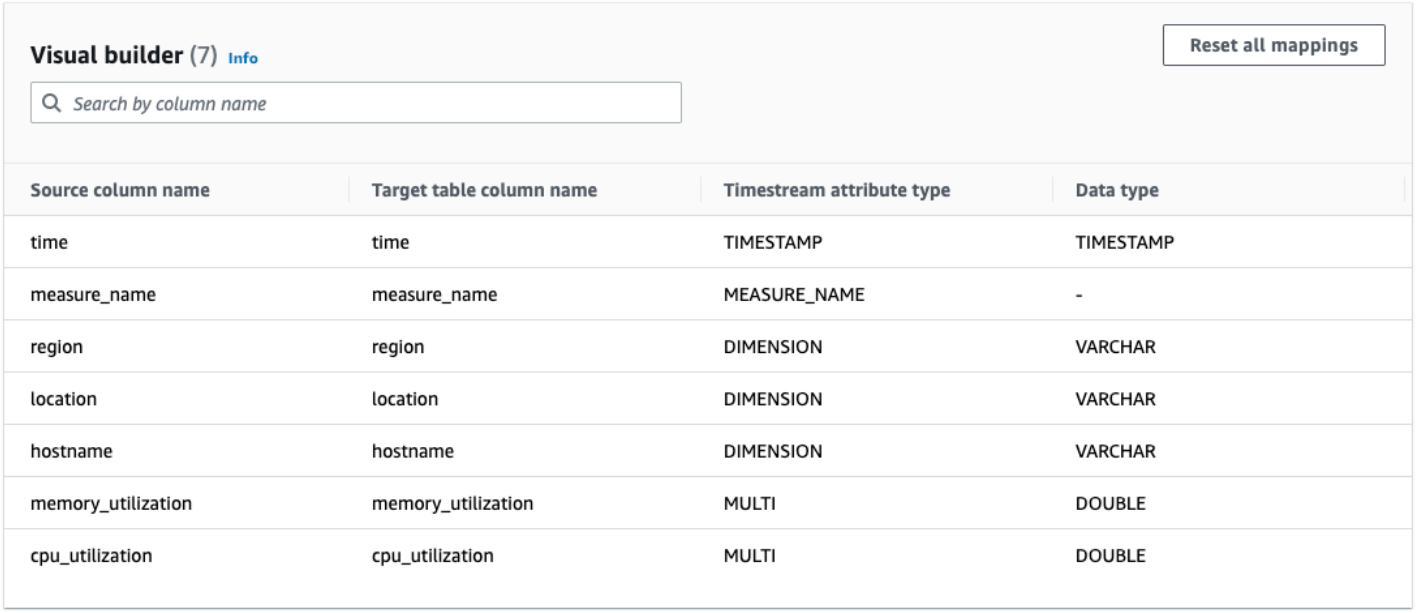
- `time`peta ke `time`
- `measure_name`peta ke `measure_name` (atau nilai yang Anda pilih)

Saat menggunakanAPI, Anda menentukan `time` di `TimeColumn` bidang dan nilai satuan waktu yang didukung seperti `MILLISECONDS` di `TimeUnit` bidang. Ini sesuai dengan nama kolom Sumber dan masukan waktu stempel waktu di konsol. Anda dapat mengelompokkan atau mempartisi catatan menggunakan `measure_name` yang didefinisikan dengan `MeasureNameColumn` kunci.

Dalam sampel,`region`,`location`, dan `hostname` dimensi. Dimensi dipetakan dalam array `DimensionMapping` objek.

Untuk ukuran, nilainya `TargetMultiMeasureAttributeName` akan menjadi kolom di Timestream untuk LiveAnalytics tabel. Anda dapat menyimpan nama yang sama seperti dalam contoh ini. Atau Anda dapat menentukan yang baru. `MeasureValueType`adalah salah satu`DOUBLE`,`BIGINT`,`BOOLEAN`,`VARCHAR`, atau`TIMESTAMP`.

```
{
  "TimeColumn": "time",
  "TimeUnit": "MILLISECONDS",
  "DimensionMappings": [
    {
      "SourceColumn": "region",
      "DestinationColumn": "region"
    },
    {
      "SourceColumn": "location",
      "DestinationColumn": "location"
    },
    {
      "SourceColumn": "hostname",
      "DestinationColumn": "hostname"
    }
  ],
  "MeasureNameColumn": "measure_name",
  "MultiMeasureMappings": {
    "MultiMeasureAttributeMappings": [
      {
        "SourceColumn": "memory_utilization",
        "TargetMultiMeasureAttributeName": "memory_utilization",
        "MeasureValueType": "DOUBLE"
      },
      {
        "SourceColumn": "cpu_utilization",
        "TargetMultiMeasureAttributeName": "cpu_utilization",
        "MeasureValueType": "DOUBLE"
      }
    ]
  }
}
```



Pemetaan model data dengan contoh `MixedMeasureMappings`

Kami menyarankan Anda hanya menggunakan pendekatan ini ketika Anda perlu memetakan ke catatan ukuran tunggal di Timestream untuk. LiveAnalytics

Menggunakan pemuatan batch dengan konsol

Berikut ini adalah langkah-langkah untuk menggunakan pemuatan batch dengan file AWS Management Console. Anda dapat mengunduh sampel CSV di sampel CSV.

Topik

- Akses beban batch
- Buat tugas pemuatan batch
- Lanjutkan tugas pemuatan batch
- Menggunakan pembangun visual

Akses beban batch

Ikuti langkah-langkah ini untuk mengakses pemuatan batch menggunakan file AWS Management Console.

1. Buka konsol Amazon Timestream.
2. Di panel navigasi, pilih Alat Manajemen, lalu pilih Tugas pemuatan Batch.

3. Dari sini, Anda dapat melihat daftar tugas pemuatan batch dan menelusuri tugas yang diberikan untuk detail selengkapnya. Anda juga dapat membuat dan melanjutkan tugas.

Buat tugas pemuatan batch

Ikuti langkah-langkah ini untuk membuat tugas pemuatan batch menggunakan AWS Management Console.

1. Buka konsol Amazon Timestream.
2. Di panel navigasi, pilih Alat Manajemen, lalu pilih Tugas pemuatan Batch.
3. Pilih Buat tugas pemuatan batch.
4. Di tujuan Impor, pilih yang berikut ini.

   - Database target - Pilih nama database yang dibuat diBuat database.
   - Tabel target - Pilih nama tabel yang dibuat diMembuat tabel.

   Jika perlu, Anda dapat menambahkan tabel dari panel ini dengan tombol Buat tabel baru.
5. Dari lokasi S3 sumber data di Sumber data, pilih bucket S3 tempat data sumber disimpan. Gunakan tombol Browse S3 untuk melihat sumber daya S3 yang dapat diakses oleh AWS akun aktif, atau masukkan lokasi S3. URL Sumber data harus berada di wilayah yang sama.
6. Dalam Pengaturan format file (bagian yang dapat diperluas), Anda dapat menggunakan pengaturan default untuk mengurai data input. Anda juga dapat memilih Pengaturan lanjutan. Dari sana Anda dapat memilih parameter CSV format, dan memilih parameter untuk mengurai data input. Untuk informasi tentang parameter ini, lihatCSVparameter format.
7. Dari Konfigurasi pemetaan model data, konfigurasikan model data. Untuk panduan model data tambahan, lihat Pemetaan model data untuk pemuatan batch

   - Dari Pemetaan model data, pilih Input konfigurasi pemetaan, dan pilih salah satu dari berikut ini.
     - Pembuat visual - Untuk memetakan data secara visual, pilih TargetMultiMeasureNameatau MeasureNameColumn. Kemudian dari Visual builder, petakan kolom.

       Pembuat visual secara otomatis mendeteksi dan memuat header kolom sumber dari file sumber data saat satu CSV file dipilih sebagai sumber data. Pilih atribut dan tipe data untuk membuat pemetaan Anda.

Untuk informasi tentang menggunakan pembuat visual, lihatMenggunakan pembangun visual.

- JSONeditor — JSON Editor bentuk bebas untuk mengonfigurasi model data Anda. Pilih opsi ini jika Anda terbiasa dengan Timestream LiveAnalytics dan ingin membangun pemetaan model data lanjutan.
- JSONfile dari S3 - Pilih file JSON model yang telah Anda simpan di S3. Pilih opsi ini jika Anda sudah mengonfigurasi model data dan ingin menggunakannya kembali untuk pemuatan batch tambahan.

8. Dari lokasi log Kesalahan S3 di laporan log kesalahan, pilih lokasi S3 yang akan digunakan untuk melaporkan kesalahan. Untuk informasi tentang cara menggunakan laporan ini, lihatMenggunakan laporan kesalahan pemuatan batch.
9. Untuk jenis kunci Enkripsi, pilih salah satu dari berikut ini.
   - Kunci terkelola Amazon S3 (SSE-S3) — Kunci enkripsi yang dibuat, dikelola, dan digunakan Amazon S3 untuk Anda.
   - AWS KMS key (SSE-KMS) — Kunci enkripsi yang dilindungi oleh AWS Key Management Service (AWS KMS).
10. Pilih Berikutnya.
11. Pada halaman Tinjau dan buat, tinjau pengaturan dan edit seperlunya.

> **Note**
>
> Anda tidak dapat mengubah pengaturan tugas pemuatan batch setelah tugas dibuat. Waktu penyelesaian tugas akan bervariasi berdasarkan jumlah data yang diimpor.

12. Pilih Buat tugas pemuatan batch.

Lanjutkan tugas pemuatan batch

Saat Anda memilih tugas pemuatan batch dengan status “Kemajuan dihentikan” yang masih dapat dilanjutkan, Anda diminta untuk melanjutkan tugas. Ada juga spanduk dengan tombol Lanjutkan tugas saat Anda melihat detail untuk tugas-tugas tersebut. Tugas yang dapat dilanjutkan memiliki tanggal “resume berdasarkan”. Setelah tanggal tersebut berakhir, tugas tidak dapat dilanjutkan.

Menggunakan pembangun visual

Anda dapat menggunakan pembuat visual untuk memetakan kolom data sumber satu atau beberapa CSV file yang disimpan dalam bucket S3 ke kolom tujuan dalam Timestream untuk LiveAnalytics tabel.

> **Note**
>
> Peran Anda akan membutuhkan `SelectObjectContent` izin untuk file tersebut. Tanpa ini, Anda perlu menambah dan menghapus kolom secara manual.

Mode kolom sumber beban otomatis

Timestream for LiveAnalytics dapat secara otomatis memindai CSV file sumber untuk nama kolom jika Anda menentukan satu ember saja. Ketika tidak ada pemetaan yang ada, Anda dapat memilih Impor kolom sumber.

1. Dengan opsi pembangun Visual yang dipilih dari pengaturan input konfigurasi Pemetaan, atur masukan waktu stempel waktu. `Milliseconds`adalah pengaturan default.
2. Klik Muat kolom sumber tombol untuk mengimpor header kolom yang ditemukan di file data sumber. Tabel akan diisi dengan nama header kolom sumber dari file sumber data.
3. Pilih nama kolom tabel Target, Tipe atribut Timestream, dan Tipe data untuk setiap kolom sumber.

   Untuk detail tentang kolom ini dan nilai yang mungkin, lihatBidang pemetaan.
4. Gunakan drag-to-fill fitur ini untuk mengatur nilai untuk beberapa kolom sekaligus.

Tambahkan kolom sumber secara manual

Jika Anda menggunakan bucket atau CSV awalan dan bukan satu punCSV, Anda dapat menambahkan dan menghapus pemetaan kolom dari editor visual dengan tombol Tambahkan pemetaan kolom dan Hapus pemetaan kolom. Ada juga tombol untuk mengatur ulang pemetaan.

Bidang pemetaan

- Nama kolom sumber - Nama kolom dalam file sumber yang mewakili ukuran untuk diimpor. Timestream for LiveAnalytics dapat mengisi nilai ini secara otomatis saat Anda menggunakan kolom sumber Impor.

- Nama kolom tabel target - Input opsional yang menunjukkan nama kolom untuk ukuran dalam tabel target.
- Tipe atribut Timestream - Jenis atribut data di kolom sumber tertentu sepertiDIMENSION.
  - TIMESTAMP- Menentukan kapan ukuran dikumpulkan.
  - MULTI— Beberapa ukuran diwakili.
  - DIMENSION— Metadata deret waktu.
  - MEASURE_ NAME — Untuk catatan ukuran tunggal, ini adalah nama ukuran.
- Tipe data — Tipe kolom Timestream, sepertiBOOLEAN.
  - BIGINT— Bilangan bulat 64-bit.
  - BOOLEAN— Dua nilai kebenaran logika — benar dan salah.
  - DOUBLE- Nomor presisi variabel 64-bit.
  - TIMESTAMP— Sebuah contoh dalam waktu yang menggunakan waktu presisi nanodetikUTC, dan melacak waktu sejak zaman Unix.

Menggunakan beban batch dengan AWS CLI

Pengaturan

Untuk mulai menggunakan pemuatan batch, lakukan langkah-langkah berikut.

1. Instal AWS CLI menggunakan instruksi diMengakses Amazon Timestream LiveAnalytics untuk menggunakan AWS CLI.
2. Jalankan perintah berikut untuk memverifikasi bahwa CLI perintah Timestream telah diperbarui. Verifikasi yang create-batch-load-task ada dalam daftar.

   ```
   aws timestream-write help
   ```
3. Siapkan sumber data menggunakan instruksi diMempersiapkan file data pemuatan batch.
4. Buat database dan tabel menggunakan instruksi diMengakses Amazon Timestream LiveAnalytics untuk menggunakan AWS CLI.
5. Buat bucket S3 untuk output laporan. Ember harus berada di Wilayah yang sama. Untuk informasi selengkapnya tentang bucket, lihat Membuat, mengonfigurasi, dan bekerja dengan bucket Amazon S3.
6. Buat tugas pemuatan batch. Untuk langkah, lihat Buat tugas pemuatan batch.
7. Konfirmasikan status tugas. Untuk langkah, lihat Jelaskan tugas pemuatan batch.

Buat tugas pemuatan batch

Anda dapat membuat tugas pemuatan batch dengan `create-batch-load-task` perintah. Saat Anda membuat tugas pemuatan batch menggunakanCLI, Anda dapat menggunakan JSON parameter`cli-input-json`, yang memungkinkan Anda menggabungkan parameter ke dalam satu JSON fragmen. Anda juga dapat memisahkan detail tersebut menggunakan beberapa parameter lain termasuk`data-model-configuration`,`data-source-configuration`,`report-configuration`,`target-database-name`, dan`target-table-name`.

Sebagai contoh, lihat Buat contoh tugas pemuatan batch

Jelaskan tugas pemuatan batch

Anda dapat mengambil deskripsi tugas pemuatan batch sebagai berikut.

```
aws timestream-write describe-batch-load-task --task-id <value>
```

Berikut adalah respons contohnya:

```
{
    "BatchLoadTaskDescription": {
        "TaskId": "<TaskId>",
        "DataSourceConfiguration": {
            "DataSourceS3Configuration": {
                "BucketName": "test-batch-load-west-2",
                "ObjectKeyPrefix": "sample.csv"
            },
            "CsvConfiguration": {},
            "DataFormat": "CSV"
        },
        "ProgressReport": {
            "RecordsProcessed": 2,
            "RecordsIngested": 0,
            "FileParseFailures": 0,
            "RecordIngestionFailures": 2,
            "FileFailures": 0,
            "BytesIngested": 119
        },
        "ReportConfiguration": {
            "ReportS3Configuration": {
                "BucketName": "test-batch-load-west-2",
                "ObjectKeyPrefix": "<ObjectKeyPrefix>",
```

```
                "EncryptionOption": "SSE_S3"
            }
        },
        "DataModelConfiguration": {
            "DataModel": {
                "TimeColumn": "timestamp",
                "TimeUnit": "SECONDS",
                "DimensionMappings": [
                    {
                        "SourceColumn": "vehicle",
                        "DestinationColumn": "vehicle"
                    },
                    {
                        "SourceColumn": "registration",
                        "DestinationColumn": "license"
                    }
                ],
                "MultiMeasureMappings": {
                    "TargetMultiMeasureName": "test",
                    "MultiMeasureAttributeMappings": [
                        {
                            "SourceColumn": "wgt",
                            "TargetMultiMeasureAttributeName": "weight",
                            "MeasureValueType": "DOUBLE"
                        },
                        {
                            "SourceColumn": "spd",
                            "TargetMultiMeasureAttributeName": "speed",
                            "MeasureValueType": "DOUBLE"
                        },
                        {
                            "SourceColumn": "fuel",
                            "TargetMultiMeasureAttributeName": "fuel",
                            "MeasureValueType": "DOUBLE"
                        },
                        {
                            "SourceColumn": "miles",
                            "TargetMultiMeasureAttributeName": "miles",
                            "MeasureValueType": "DOUBLE"
                        }
                    ]
                }
            }
        },
```

```
        "TargetDatabaseName": "BatchLoadExampleDatabase",
        "TargetTableName": "BatchLoadExampleTable",
        "TaskStatus": "FAILED",
        "RecordVersion": 1,
        "CreationTime": 1677167593.266,
        "LastUpdatedTime": 1677167602.38
    }
}
```

Buat daftar tugas pemuatan batch

Anda dapat membuat daftar tugas pemuatan batch sebagai berikut.

```
aws timestream-write list-batch-load-tasks
```

Output muncul sebagai berikut.

```
{
    "BatchLoadTasks": [
        {
            "TaskId": "<TaskId>",
            "TaskStatus": "FAILED",
            "DatabaseName": "BatchLoadExampleDatabase",
            "TableName": "BatchLoadExampleTable",
            "CreationTime": 1677167593.266,
            "LastUpdatedTime": 1677167602.38
        }
    ]
}
```

Lanjutkan tugas pemuatan batch

Anda dapat melanjutkan tugas pemuatan batch sebagai berikut.

```
aws timestream-write resume-batch-load-task --task-id <value>
```

Respons dapat menunjukkan keberhasilan atau mengandung informasi kesalahan.

Buat contoh tugas pemuatan batch

Example

1. Buat Timestream untuk LiveAnalytics database bernama `BatchLoad` dan tabel bernama`BatchLoadTest`. Verifikasi dan, jika perlu, sesuaikan nilai untuk `MemoryStoreRetentionPeriodInHours` dan`MagneticStoreRetentionPeriodInDays`.

```
aws timestream-write create-database --database-name BatchLoad \

aws timestream-write create-table --database-name BatchLoad \
--table-name BatchLoadTest \
--retention-properties "{\"MemoryStoreRetentionPeriodInHours\": 12,
 \"MagneticStoreRetentionPeriodInDays\": 100}"
```

2. Menggunakan konsol, buat bucket S3 dan salin `sample.csv` file ke lokasi itu. Anda dapat mengunduh sampel CSV di sampel CSV.
3. Menggunakan konsol, buat bucket S3 untuk Timestream LiveAnalytics untuk menulis laporan jika tugas pemuatan batch selesai dengan kesalahan.
4. Buat tugas pemuatan batch. Pastikan untuk mengganti *`$INPUT_BUCKET`* and *`$REPORT_BUCKET`* dengan ember yang Anda buat pada langkah sebelumnya.

```
aws timestream-write create-batch-load-task \
--data-model-configuration "{\
            \"DataModel\": {\
              \"TimeColumn\": \"timestamp\",\
              \"TimeUnit\": \"SECONDS\",\
              \"DimensionMappings\": [\
                {\
                  \"SourceColumn\": \"vehicle\"\
                },\
                {\
                  \"SourceColumn\": \"registration\",\
                  \"DestinationColumn\": \"license\"\
                }\
              ],
              \"MultiMeasureMappings\": {\
                \"TargetMultiMeasureName\": \"mva_measure_name\",\
                \"MultiMeasureAttributeMappings\": [\
                  {\
                    \"SourceColumn\": \"wgt\",\
```

```
                  \"TargetMultiMeasureAttributeName\": \"weight\",\
                  \"MeasureValueType\": \"DOUBLE\"\
                },\
                {\
                  \"SourceColumn\": \"spd\",\
                  \"TargetMultiMeasureAttributeName\": \"speed\",\
                  \"MeasureValueType\": \"DOUBLE\"\
                },\
                {\
                  \"SourceColumn\": \"fuel_consumption\",\
                  \"TargetMultiMeasureAttributeName\": \"fuel\",\
                  \"MeasureValueType\": \"DOUBLE\"\
                },\
                {\
                  \"SourceColumn\": \"miles\",\
                  \"MeasureValueType\": \"BIGINT\"\
                }\
              ]\
            }\
          }\
        }" \
--data-source-configuration "{
          \"DataSourceS3Configuration\": {\
            \"BucketName\": \"$INPUT_BUCKET\",\
            \"ObjectKeyPrefix\": \"$INPUT_OBJECT_KEY_PREFIX\"
          },\
          \"DataFormat\": \"CSV\"\
        }" \
--report-configuration "{\
          \"ReportS3Configuration\": {\
            \"BucketName\": \"$REPORT_BUCKET\",\
            \"EncryptionOption\": \"SSE_S3\"\
          }\
        }" \
--target-database-name BatchLoad \
--target-table-name BatchLoadTest
```

Perintah sebelumnya mengembalikan output berikut.

```
{
    "TaskId": "TaskId "
}
```

5. Periksa kemajuan tugas. Pastikan Anda mengganti *$TASK_ID* dengan id tugas yang dikembalikan pada langkah sebelumnya.

```
aws timestream-write describe-batch-load-task --task-id $TASK_ID
```

Contoh Output

```
{
    "BatchLoadTaskDescription": {
        "ProgressReport": {
            "BytesIngested": 1024,
            "RecordsIngested": 2,
            "FileFailures": 0,
            "RecordIngestionFailures": 0,
            "RecordsProcessed": 2,
            "FileParseFailures": 0
        },
        "DataModelConfiguration": {
            "DataModel": {
                "DimensionMappings": [
                    {
                        "SourceColumn": "vehicle",
                        "DestinationColumn": "vehicle"
                    },
                    {
                        "SourceColumn": "registration",
                        "DestinationColumn": "license"
                    }
                ],
                "TimeUnit": "SECONDS",
                "TimeColumn": "timestamp",
                "MultiMeasureMappings": {
                    "MultiMeasureAttributeMappings": [
                        {
                            "TargetMultiMeasureAttributeName": "weight",
                            "SourceColumn": "wgt",
                            "MeasureValueType": "DOUBLE"
                        },
                        {
                            "TargetMultiMeasureAttributeName": "speed",
                            "SourceColumn": "spd",
                            "MeasureValueType": "DOUBLE"
```

```
                        },
                        {
                            "TargetMultiMeasureAttributeName": "fuel",
                            "SourceColumn": "fuel_consumption",
                            "MeasureValueType": "DOUBLE"
                        },
                        {
                            "TargetMultiMeasureAttributeName": "miles",
                            "SourceColumn": "miles",
                            "MeasureValueType": "DOUBLE"
                        }
                    ],
                    "TargetMultiMeasureName": "mva_measure_name"
                }
            }
        },
        "TargetDatabaseName": "BatchLoad",
        "CreationTime": 1672960381.735,
        "TaskStatus": "SUCCEEDED",
        "RecordVersion": 1,
        "TaskId": "TaskId ",
        "TargetTableName": "BatchLoadTest",
        "ReportConfiguration": {
            "ReportS3Configuration": {
                "EncryptionOption": "SSE_S3",
                "ObjectKeyPrefix": "ObjectKeyPrefix ",
                "BucketName": "test-report-bucket"
            }
        },
        "DataSourceConfiguration": {
            "DataSourceS3Configuration": {
                "ObjectKeyPrefix": "sample.csv",
                "BucketName": "test-input-bucket"
            },
            "DataFormat": "CSV",
            "CsvConfiguration": {}
        },
        "LastUpdatedTime": 1672960387.334
    }
}
```

Menggunakan beban batch dengan AWS SDKs

Untuk contoh cara membuat, mendeskripsikan, dan membuat daftar tugas pemuatan batch dengan AWS SDKs, lihatBuat tugas pemuatan batch,Jelaskan tugas pemuatan batch,Buat daftar tugas pemuatan batch, danLanjutkan tugas pemuatan batch.

Menggunakan laporan kesalahan pemuatan batch

Tugas pemuatan batch memiliki salah satu nilai status berikut:

- `CREATED`(Dibuat) - Tugas dibuat.
- `IN_PROGRESS`(Sedang berlangsung) — Tugas sedang berlangsung.
- `FAILED`(Gagal) — Tugas telah selesai. Tetapi satu atau lebih kesalahan terdeteksi.
- `SUCCEEDED`(Selesai) — Tugas telah selesai tanpa kesalahan.
- `PROGRESS_STOPPED`(Kemajuan berhenti) — Tugas telah berhenti tetapi tidak selesai. Anda dapat mencoba melanjutkan tugas.
- `PENDING_RESUME`(Resume tertunda) — Tugas sedang menunggu untuk dilanjutkan.

Ketika ada kesalahan, laporan log kesalahan dibuat di bucket S3 yang ditentukan untuk itu. Kesalahan dikategorikan sebagai taskErrors atau fileErrors dalam array terpisah. Berikut ini adalah contoh laporan kesalahan.

```
{
    "taskId": "9367BE28418C5EF902676482220B631C",
    "taskErrors": [],
    "fileErrors": [
        {
            "fileName": "example.csv",
            "errors": [
                {
                    "reason": "The record timestamp is outside the time range of the
 data ingestion window.",
                    "lineRanges": [
                        [
                            2,
                            3
                        ]
                    ]
                }
```

```
            ]
        }
    ]
}
```

Menggunakan kueri terjadwal di Timestream untuk LiveAnalytics

Fitur kueri terjadwal di Amazon Timestream for LiveAnalytics adalah solusi yang dikelola sepenuhnya, tanpa server, dan dapat diskalakan untuk menghitung dan menyimpan agregat, rollup, dan bentuk data pra-proses lainnya yang biasanya digunakan untuk dasbor operasional, laporan bisnis, analitik ad-hoc, dan aplikasi lainnya. Kueri terjadwal membuat analitik real-time lebih berkinerja dan hemat biaya, sehingga Anda dapat memperoleh wawasan tambahan dari data Anda, dan dapat terus membuat keputusan bisnis yang lebih baik.

Dengan kueri terjadwal, Anda menentukan kueri analitik real-time yang menghitung agregat, rollup, dan operasi lainnya pada data—dan Amazon Timestream untuk menjalankan kueri ini LiveAnalytics secara berkala dan otomatis dan secara andal menulis hasil kueri ke dalam tabel terpisah. Data biasanya dihitung dan diperbarui ke dalam tabel ini dalam beberapa menit.

Anda kemudian dapat mengarahkan dasbor dan laporan untuk menanyakan tabel yang berisi data agregat alih-alih menanyakan tabel sumber yang jauh lebih besar. Ini mengarah pada kinerja dan keuntungan biaya yang dapat melebihi urutan besarnya. Ini karena tabel dengan data agregat mengandung data yang jauh lebih sedikit daripada tabel sumber, sehingga mereka menawarkan kueri yang lebih cepat dan penyimpanan data yang lebih murah.

Selain itu, tabel dengan kueri terjadwal menawarkan semua fungsionalitas Timestream yang ada untuk LiveAnalytics tabel. Misalnya, Anda dapat menanyakan tabel menggunakanSQL. Anda dapat memvisualisasikan data yang disimpan dalam tabel menggunakan Grafana. Anda juga dapat memasukkan data ke dalam tabel menggunakan Amazon Kinesis, Amazon, AWS IoT CoreMSK, dan Telegraf. Anda dapat mengonfigurasi kebijakan penyimpanan data pada tabel ini untuk pengelolaan siklus hidup data otomatis.

Karena retensi data tabel yang berisi data agregat sepenuhnya dipisahkan dari tabel sumber, Anda juga dapat memilih untuk mengurangi retensi data tabel sumber dan menyimpan data agregat untuk durasi yang lebih lama, dengan sebagian kecil dari biaya penyimpanan data. Kueri terjadwal membuat analitik real-time lebih cepat, lebih murah, dan karenanya lebih mudah diakses oleh lebih banyak pelanggan, sehingga mereka dapat memantau aplikasi mereka dan mendorong keputusan bisnis berbasis data yang lebih baik.

Topik

Manfaat kueri terjadwal

Berikut ini adalah manfaat dari kueri terjadwal:

- Kemudahan operasional - Kueri terjadwal tanpa server dan dikelola sepenuhnya.
- Kinerja dan biaya — Karena kueri terjadwal menghitung agregat, rollup, atau operasi analitik real-time lainnya untuk data Anda dan menyimpan hasilnya dalam tabel, kueri yang mengakses tabel yang diisi oleh kueri terjadwal mengandung lebih sedikit data daripada tabel sumber. Oleh karena itu, kueri yang dijalankan pada tabel ini lebih cepat dan lebih murah. Tabel yang diisi oleh perhitungan terjadwal mengandung lebih sedikit data daripada tabel sumbernya, dan karenanya membantu mengurangi biaya penyimpanan. Anda juga dapat menyimpan data ini untuk durasi yang lebih lama di penyimpanan memori dengan biaya sebagian kecil dari biaya mempertahankan data sumber di penyimpanan memori.
- Interoperabilitas — Tabel yang diisi oleh kueri terjadwal menawarkan semua fungsionalitas Timestream yang ada untuk LiveAnalytics tabel dan dapat digunakan dengan semua layanan dan alat yang bekerja dengan Timestream untuk. LiveAnalytics Lihat Bekerja dengan Layanan Lain untuk detailnya.

Kasus penggunaan kueri terjadwal

Anda dapat menggunakan kueri terjadwal untuk laporan bisnis yang merangkum aktivitas pengguna akhir dari aplikasi Anda, sehingga Anda dapat melatih model pembelajaran mesin untuk

personalisasi. Anda juga dapat menggunakan kueri terjadwal untuk alarm yang mendeteksi anomali, gangguan jaringan, atau aktivitas penipuan, sehingga Anda dapat segera mengambil tindakan perbaikan.

Selain itu, Anda dapat menggunakan kueri terjadwal untuk tata kelola data yang lebih efektif. Anda dapat melakukan ini dengan memberikan akses tabel sumber secara eksklusif ke kueri terjadwal, dan memberikan pengembang Anda akses ke hanya tabel yang diisi oleh kueri terjadwal. Ini meminimalkan dampak kueri yang tidak disengaja dan berjalan lama.

Contoh: Menggunakan analitik real-time untuk mendeteksi pembayaran penipuan dan membuat keputusan bisnis yang lebih baik

Pertimbangkan sistem pembayaran yang memproses transaksi yang dikirim dari beberapa point-of-sale terminal yang didistribusikan di kota-kota metropolitan besar di Amerika Serikat. Anda ingin menggunakan Amazon Timestream LiveAnalytics untuk menyimpan dan menganalisis data transaksi, sehingga Anda dapat mendeteksi transaksi penipuan dan menjalankan kueri analitik real-time. Pertanyaan ini dapat membantu Anda menjawab pertanyaan bisnis seperti mengidentifikasi point-of-sale terminal tersibuk dan paling jarang digunakan per jam, jam tersibuk setiap hari untuk setiap kota, dan kota dengan transaksi terbanyak per jam.

Proses sistem ~ 100K transaksi per menit. Setiap transaksi yang disimpan di Amazon Timestream LiveAnalytics adalah 100 byte. Anda telah mengonfigurasi 10 kueri yang berjalan setiap menit untuk mendeteksi berbagai jenis pembayaran penipuan. Anda juga telah membuat 25 kueri yang mengumpulkan dan mengiris/memotong data Anda di berbagai dimensi untuk membantu menjawab pertanyaan bisnis Anda. Masing-masing kueri ini memproses data jam terakhir.

Anda telah membuat dasbor untuk menampilkan data yang dihasilkan oleh kueri ini. Dasbor berisi 25 widget, disegarkan setiap jam, dan biasanya diakses oleh 10 pengguna pada waktu tertentu. Terakhir, penyimpanan memori Anda dikonfigurasi dengan periode retensi data 2 jam dan penyimpanan magnetik dikonfigurasi untuk memiliki periode retensi data 6 bulan.

Dalam hal ini, Anda dapat menggunakan kueri analitik real-time yang menghitung ulang data setiap kali dasbor diakses dan di-refresh, atau menggunakan tabel turunan untuk dasbor. Biaya kueri untuk dasbor berdasarkan kueri analitik real-time adalah $120,70 per bulan. Sebaliknya, biaya kueri dasbor yang didukung oleh tabel turunan akan menjadi $12,27 per bulan (lihat Amazon Timestream untuk harga). LiveAnalytics Dalam hal ini, menggunakan tabel turunan mengurangi biaya kueri sebesar ~ 10 kali.

Konsep kueri terjadwal

Query string - Ini adalah query yang hasilnya Anda pra-komputasi dan menyimpan di Timestream lain untuk LiveAnalytics tabel. Anda dapat menentukan kueri terjadwal menggunakan luas SQL permukaan penuh Timestream untuk LiveAnalytics, yang memberi Anda fleksibilitas menulis kueri dengan ekspresi tabel umum, kueri bersarang, fungsi jendela, atau segala jenis fungsi agregat dan skalar yang didukung oleh Timestream untuk bahasa kueri. LiveAnalytics

Ekspresi jadwal - Memungkinkan Anda menentukan kapan instance kueri terjadwal Anda dijalankan. Anda dapat menentukan ekspresi menggunakan ekspresi cron (seperti dijalankan pada jam 8 pagi UTC setiap hari) atau ekspresi tingkat (seperti dijalankan setiap 10 menit).

Konfigurasi target - Memungkinkan Anda menentukan bagaimana Anda memetakan hasil kueri terjadwal ke dalam tabel tujuan tempat hasil kueri terjadwal ini akan disimpan.

Konfigurasi pemberitahuan -Timestream untuk menjalankan instance kueri terjadwal LiveAnalytics secara otomatis berdasarkan ekspresi jadwal Anda. Anda menerima pemberitahuan untuk setiap kueri yang dijalankan pada SNS topik yang Anda konfigurasikan saat membuat kueri terjadwal. Pemberitahuan ini menentukan apakah instance berhasil dijalankan atau mengalami kesalahan apa pun. Selain itu, ia memberikan informasi seperti byte yang diukur, data yang ditulis ke tabel target, waktu pemanggilan berikutnya, dan sebagainya.

Berikut ini adalah contoh pesan notifikasi semacam ini.

```
{
    "type":"AUTO_TRIGGER_SUCCESS",
    "arn":"arn:aws:timestream:us-east-1:123456789012:scheduled-query/
 PT1mPerMinutePerRegionMeasureCount-9376096f7309",
    "nextInvocationEpochSecond":1637302500,
    "scheduledQueryRunSummary":
    {
        "invocationEpochSecond":1637302440,
        "triggerTimeMillis":1637302445697,
        "runStatus":"AUTO_TRIGGER_SUCCESS",
        "executionStats":
        {
            "executionTimeInMillis":21669,
            "dataWrites":36864,
            "bytesMetered":13547036820,
            "recordsIngested":1200,
            "queryResultRows":1200
```

```
        }
    }
}
```

Dalam pesan notifikasi ini, `bytesMetered` adalah byte yang dipindai kueri pada tabel sumber, dan dataWrites byte ditulis ke tabel target.

> **Note**
>
> Jika Anda menggunakan notifikasi ini secara terprogram, ketahuilah bahwa bidang baru dapat ditambahkan ke pesan notifikasi di masa mendatang.

Lokasi laporan kesalahan - Kueri terjadwal menjalankan dan menyimpan data secara asinkron dalam tabel target. Jika sebuah instance mengalami kesalahan apa pun (misalnya, data tidak valid yang tidak dapat disimpan), catatan yang mengalami kesalahan ditulis ke laporan kesalahan di lokasi laporan kesalahan yang Anda tentukan saat pembuatan kueri terjadwal. Anda menentukan bucket S3 dan awalan untuk lokasi. Timestream untuk LiveAnalytics menambahkan nama kueri terjadwal dan waktu pemanggilan ke awalan ini untuk membantu Anda mengidentifikasi kesalahan yang terkait dengan instance tertentu dari kueri terjadwal.

Penandaan - Anda dapat secara opsional menentukan tag yang dapat Anda kaitkan dengan kueri terjadwal. Untuk detail selengkapnya, lihat Menandai Timestream for LiveAnalytics Resources.

Contoh

Dalam contoh berikut, Anda menghitung agregat sederhana menggunakan kueri terjadwal:

```
SELECT region, bin(time, 1m) as minute,
    SUM(CASE WHEN measure_name = 'metrics' THEN 20 ELSE 5 END) as numDataPoints
FROM raw_data.devops
WHERE time BETWEEN @scheduled_runtime - 10m AND @scheduled_runtime + 1m
GROUP BY bin(time, 1m), region
```

`@scheduled_runtime parameter`- Dalam contoh ini, Anda akan melihat kueri menerima parameter `@scheduled_runtime` bernama khusus. Ini adalah parameter khusus (tipe Timestamp) yang ditetapkan layanan saat menjalankan instance spesifik dari kueri terjadwal sehingga Anda dapat secara deterministik mengontrol rentang waktu dimana instance spesifik dari kueri terjadwal menganalisis data dalam tabel sumber. Anda dapat menggunakan kueri Anda `@scheduled_runtime` di lokasi mana pun di mana jenis Timestamp diharapkan.

Pertimbangkan contoh di mana Anda menetapkan ekspresi jadwal: cron (0/5 * * *? *) di mana kueri terjadwal akan berjalan pada menit 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55 setiap jam. Untuk instance yang dipicu pada 2021-12-01 00:05:00, parameter @scheduled_runtime diinisialisasi ke nilai ini, sehingga instance saat ini beroperasi pada data dalam kisaran 2021-11-30 23:55:00 hingga 2021-12-01 00:06:00.

Contoh dengan rentang waktu yang tumpang tindih - Seperti yang akan Anda lihat dalam contoh ini, dua contoh berikutnya dari kueri terjadwal dapat tumpang tindih dalam rentang waktu mereka. Ini adalah sesuatu yang dapat Anda kontrol berdasarkan kebutuhan Anda, predikat waktu yang Anda tentukan, dan ekspresi jadwal. Dalam hal ini, tumpang tindih ini memungkinkan perhitungan ini untuk memperbarui agregat berdasarkan data apa pun yang kedatangannya sedikit tertunda, hingga 10 menit dalam contoh ini. Query run yang dipicu pada 2021-12-01 00:00:00 akan mencakup rentang waktu 2021-11-30 23:50:00 hingga 2021-12-30 00:01:00 dan proses kueri yang dipicu pada 2021-12-01 00:05:00 akan mencakup kisaran 2021-11-30 23:55:00 hingga 2021-12-01 00:06:00.

Untuk memastikan kebenaran dan memastikan bahwa agregat yang disimpan dalam tabel target cocok dengan agregat yang dihitung dari tabel sumber, Timestream untuk LiveAnalytics memastikan bahwa perhitungan pada 2021-12-01 00:05:00 akan dilakukan hanya setelah perhitungan pada 2021-12-01 00:00:00 telah selesai. Hasil perhitungan terakhir dapat memperbarui agregat yang terwujud sebelumnya jika nilai yang lebih baru dihasilkan. Secara internal, Timestream untuk LiveAnalytics menggunakan versi rekaman di mana catatan yang dihasilkan oleh instance terakhir dari kueri terjadwal akan diberi nomor versi yang lebih tinggi. Oleh karena itu, agregat yang dihitung dengan pemanggilan pada 2021-12-01 00:05:00 dapat memperbarui agregat yang dihitung dengan pemanggilan pada 2021-12-01 00:00:00, dengan asumsi data yang lebih baru tersedia di tabel sumber.

Pemicu otomatis vs. pemicu manual - Setelah kueri terjadwal dibuat, Timestream for LiveAnalytics akan secara otomatis menjalankan instance berdasarkan jadwal yang ditentukan. Pemicu otomatis semacam itu dikelola sepenuhnya oleh layanan.

Namun, mungkin ada skenario di mana Anda mungkin ingin secara manual memulai beberapa contoh kueri terjadwal. Contohnya termasuk jika instance tertentu gagal dalam menjalankan kueri, jika ada data yang datang terlambat atau pembaruan dalam tabel sumber setelah jadwal otomatis berjalan, atau jika Anda ingin memperbarui tabel target untuk rentang waktu yang tidak tercakup oleh kueri otomatis berjalan (misalnya, untuk rentang waktu sebelum pembuatan kueri terjadwal).

Anda dapat menggunakan ExecuteScheduledQuery API untuk secara manual memulai instance spesifik dari kueri terjadwal dengan meneruskan InvocationTime parameter, yang merupakan nilai

yang digunakan untuk parameter @scheduled_runtime. Berikut ini adalah beberapa pertimbangan penting saat menggunakan: ExecuteScheduledQuery API

- Jika Anda memicu beberapa pemanggilan ini, Anda perlu memastikan bahwa pemanggilan ini tidak menghasilkan hasil dalam rentang waktu yang tumpang tindih. Jika Anda tidak dapat memastikan rentang waktu yang tidak tumpang tindih, maka pastikan bahwa kueri ini dijalankan secara berurutan satu demi satu. Jika Anda secara bersamaan memulai beberapa proses kueri yang tumpang tindih dalam rentang waktunya, maka Anda dapat melihat kegagalan pemicu di mana Anda mungkin melihat konflik versi dalam laporan kesalahan untuk kueri ini berjalan.
- Anda dapat memulai pemanggilan dengan nilai stempel waktu apa pun untuk @scheduled_runtime. Jadi, Anda bertanggung jawab untuk mengatur nilai dengan tepat sehingga rentang waktu yang sesuai diperbarui dalam tabel target yang sesuai dengan rentang di mana data diperbarui di tabel sumber.

Jadwalkan ekspresi untuk kueri terjadwal

Anda dapat membuat kueri terjadwal pada jadwal otomatis dengan menggunakan Amazon Timestream LiveAnalytics untuk kueri terjadwal yang menggunakan ekspresi cron atau rate. Semua kueri terjadwal menggunakan zona UTC waktu, dan presisi minimum yang mungkin untuk jadwal adalah 1 menit.

Dua cara untuk menentukan ekspresi jadwal adalah cron dan rate. Ekspresi cron menawarkan kontrol jadwal berbutir lebih halus, sementara ekspresi laju lebih sederhana untuk diekspresikan tetapi tidak memiliki kontrol halus.

Misalnya, dengan ekspresi cron, Anda dapat menentukan kueri terjadwal yang dipicu pada waktu tertentu pada hari tertentu setiap minggu atau bulan, atau menit tertentu setiap jam hanya pada hari Senin - Jumat, dan seterusnya. Sebaliknya, ekspresi tingkat memulai kueri terjadwal pada tingkat reguler, seperti sekali setiap menit, jam, atau hari, dimulai dari waktu yang tepat ketika kueri terjadwal dibuat.

Ekspresi cron

- Sintaksis

```
cron(fields)
```

  Ekspresi cron memiliki enam bidang yang diperlukan, yang dipisahkan oleh spasi putih.

Bidang	Nilai	Wildcard
Menit	0-59	, - * /
Jam	0-23	, - * /
D ay-of-month	1-31	, - * ? / L W
Bulan	1-12 atau JAN - DEC	, - * /
D ay-of-week	1-7 atau SUN - SAT	, - * ? L #
Tahun	1970-2199	, - * /

Karakter wildcard

- Wildcard *, * (koma) mencakup nilai tambahan. Di bidang Bulan,, JANFEB, MAR akan mencakup Januari, Februari, dan Maret.
- Wildcard *-* (dash) menentukan rentang. Di bidang Tanggal, 1-15 akan mencakup tanggal 1 hingga 15 pada bulan yang ditentukan.
- Wildcard *** (asterisk) mencakup semua nilai di lapangan. Di bidang Jam, *** akan mencakup setiap jam. Anda tidak dapat menggunakan *** di Day-of-week bidang Day-of-month dan bidang. Jika Anda menggunakannya dalam satu, Anda harus menggunakan*? * di sisi lain.
- Wildcard */* (garis miring ke depan) menentukan kenaikan. Di bidang Menit, Anda dapat memasukkan 1/10 untuk menentukan setiap menit ke-10, mulai dari menit pertama jam (misalnya, menit ke-11, ke-21, dan 31, dan seterusnya).
- *? * (tanda tanya) wildcard menentukan satu atau yang lain. Di Day-of-month lapangan Anda bisa memasukkan* 7* dan jika Anda tidak peduli hari apa dalam minggu ke-7, Anda bisa masuk*? * di Day-of-week lapangan.
- Wildcard *L* di Day-of-week kolom Day-of-month or menentukan hari terakhir bulan atau minggu.
- Wildcard W di Day-of-month lapangan menentukan hari kerja. Di Day-of-month lapangan, 3W menentukan hari kerja yang paling dekat dengan hari ketiga bulan itu.
- Wildcard *#* di Day-of-week bidang menentukan contoh tertentu dari hari yang ditentukan dalam seminggu dalam sebulan. Sebagai contoh, 3#2 akan menjadi hari Selasa kedua setiap bulan:

3 mengacu pada hari Selasa karena itu adalah hari ketiga setiap minggu, dan 2 mengacu pada hari kedua dari jenis tersebut dalam bulan tersebut.

> **Note**
>
> Jika Anda menggunakan karakter '#', Anda hanya dapat menentukan satu ekspresi di day-of-week bidang. Misalnya, “3 #1 ,6 #3" tidak valid karena ditafsirkan sebagai dua ekspresi.

Batasan

- Anda tidak dapat menentukan Day-of-week bidang Day-of-month dan dalam ekspresi cron yang sama. Jika Anda menentukan nilai (atau *) di salah satu bidang, Anda harus menggunakan *? * (tanda tanya) di sisi lain.
- Ekspresi cron yang mengarah ke tingkat lebih cepat dari 1 menit tidak didukung.

Contoh

Menit	Jam	Hari dalam sebulan	Bulan	Hari dalam seminggu	Tahun	Arti
0	10	*	*	?	*	Jalankan pukul 10:00 pagi (UTC) setiap hari.
15	12	*	*	?	*	Jalankan pukul 12:15 siang (UTC) setiap hari.

Menit	Jam	Hari dalam sebulan	Bulan	Hari dalam seminggu	Tahun	Arti
0	18	?	*	MON-FRI	*	Jalankan pukul 6:00 sore (UTC) setiap hari Senin sampai Jumat.
0	8	1	*	?	*	Jalankan pukul 8:00 pagi (UTC) setiap hari pertama setiap bulan.
0/15	*	*	*	?	*	Jalankan setiap 15 menit.
0/10	*	*	*	MON-FRI	*	Jalankan setiap 10 menit Senin sampai Jumat.

Menit	Jam	Hari dalam sebulan	Bulan	Hari dalam seminggu	Tahun	Arti
0/5	8-17	?	*	MON-FRI	*	Jalankan setiap 5 menit Senin hingga Jumat antara pukul 8:00 pagi dan 5:55 sore ()UTC.

Ekspresi rate

- Ekspresi rate dimulai ketika Anda membuat aturan peristiwa terjadwal, dan kemudian aturan berjalan pada jadwal yang ditetapkan. Ekspresi rate memiliki dua field wajib berikut. Field dipisahkan oleh white space.

  Sintaksis

  ```
  rate(value unit)
  ```

  - `value`: Angka positif.
  - `unit`: Satuan waktu. Unit yang berbeda diperlukan untuk nilai 1 (misalnya, menit) dan nilai lebih dari 1 (misalnya, menit). Nilai yang valid: menit | menit-menit | jam | jam-jam | hari | hari-hari

Pemetaan model data untuk kueri terjadwal

Timestream untuk LiveAnalytics mendukung pemodelan data yang fleksibel dalam tabelnya dan fleksibilitas yang sama ini berlaku untuk hasil kueri terjadwal yang diwujudkan ke Timestream lain untuk tabel. LiveAnalytics Dengan kueri terjadwal, Anda dapat menanyakan tabel apa pun, apakah itu memiliki data dalam catatan multi-ukuran atau catatan ukuran tunggal dan menulis hasil kueri menggunakan catatan multi-ukuran atau ukuran tunggal.

Anda menggunakan spesifikasi kueri terjadwal untuk memetakan hasil kueri ke kolom yang sesuai di tabel turunan tujuan. TargetConfiguration Bagian berikut menjelaskan berbagai cara menentukan ini TargetConfiguration untuk mencapai model data yang berbeda dalam tabel turunan. Secara khusus, Anda akan melihat:

- Cara menulis ke catatan multi-ukuran ketika hasil kueri tidak memiliki nama ukuran dan Anda menentukan nama ukuran target di TargetConfiguration.
- Bagaimana Anda menggunakan nama ukuran dalam hasil kueri untuk menulis catatan multi-ukuran.
- Bagaimana Anda dapat mendefinisikan model untuk menulis beberapa catatan dengan atribut multi-ukuran yang berbeda.
- Bagaimana Anda dapat mendefinisikan model untuk menulis ke catatan ukuran tunggal dalam tabel turunan.
- Bagaimana Anda dapat menanyakan catatan ukuran tunggal dan/atau catatan multi-ukuran dalam kueri terjadwal dan hasilnya terwujud menjadi catatan ukuran tunggal atau catatan multi-ukuran, yang memungkinkan Anda memilih fleksibilitas model data.

Contoh: Nama ukuran target untuk catatan multi-ukuran

Dalam contoh ini, Anda akan melihat bahwa kueri membaca data dari tabel dengan data multi-ukuran dan menulis hasilnya ke tabel lain menggunakan catatan multi-ukuran. Hasil kueri terjadwal tidak memiliki kolom nama ukuran alami. Di sini, Anda menentukan nama ukuran dalam tabel turunan menggunakan TargetMultiMeasureName properti di TargetConfiguration. TimestreamConfiguration.

```
{
    "Name" : "CustomMultiMeasureName",
    "QueryString" : "SELECT region, bin(time, 1h) as hour, AVG(memory_cached)
 as avg_mem_cached_1h, MIN(memory_free) as min_mem_free_1h, MAX(memory_used) as
 max_mem_used_1h, SUM(disk_io_writes) as sum_1h, AVG(disk_used) as avg_disk_used_1h,
 AVG(disk_free) as avg_disk_free_1h, MAX(cpu_user) as max_cpu_user_1h, MIN(cpu_idle) as
 min_cpu_idle_1h, MAX(cpu_system) as max_cpu_system_1h FROM raw_data.devops_multi WHERE
 time BETWEEN bin(@scheduled_runtime, 1h) - 14h AND bin(@scheduled_runtime, 1h) - 2h
 AND measure_name = 'metrics' GROUP BY region, bin(time, 1h)",
    "ScheduleConfiguration" : {
        "ScheduleExpression" : "cron(0 0/1 * * ? *)"
    },
    "NotificationConfiguration" : {
        "SnsConfiguration" : {
            "TopicArn" : "******"
```

```
        }
    },
    "ScheduledQueryExecutionRoleArn": "******",
    "TargetConfiguration": {
        "TimestreamConfiguration": {
            "DatabaseName" : "derived",
            "TableName" : "dashboard_metrics_1h_agg_1",
            "TimeColumn" : "hour",
            "DimensionMappings" : [
                {
                    "Name": "region",
                    "DimensionValueType" : "VARCHAR"
                }
            ],
            "MultiMeasureMappings" : {
                "TargetMultiMeasureName": "dashboard-metrics",
                "MultiMeasureAttributeMappings" : [
                    {
                        "SourceColumn" : "avg_mem_cached_1h",
                        "MeasureValueType" : "DOUBLE",
                        "TargetMultiMeasureAttributeName" : "avgMemCached"
                    },
                    {
                        "SourceColumn" : "min_mem_free_1h",
                        "MeasureValueType" : "DOUBLE"
                    },
                    {
                        "SourceColumn" : "max_mem_used_1h",
                        "MeasureValueType" : "DOUBLE"
                    },
                    {
                        "SourceColumn" : "sum_1h",
                        "MeasureValueType" : "DOUBLE",
                        "TargetMultiMeasureAttributeName" : "totalDiskWrites"
                    },
                    {
                        "SourceColumn" : "avg_disk_used_1h",
                        "MeasureValueType" : "DOUBLE"
                    },
                    {
                        "SourceColumn" : "avg_disk_free_1h",
                        "MeasureValueType" : "DOUBLE"
                    },
                    {
```

```
                        "SourceColumn" : "max_cpu_user_1h",
                        "MeasureValueType" : "DOUBLE",
                        "TargetMultiMeasureAttributeName" : "CpuUserP100"
                    },
                    {
                        "SourceColumn" : "min_cpu_idle_1h",
                        "MeasureValueType" : "DOUBLE"
                    },
                    {
                        "SourceColumn" : "max_cpu_system_1h",
                        "MeasureValueType" : "DOUBLE",
                        "TargetMultiMeasureAttributeName" : "CpuSystemP100"
                    }
                ]
            }
        }
    },
    "ErrorReportConfiguration": {
        "S3Configuration" : {
            "BucketName" : "******",
            "ObjectKeyPrefix": "errors",
            "EncryptionOption": "SSE_S3"
        }
    }
}
```

Pemetaan dalam contoh ini membuat satu catatan multi-ukuran dengan nama ukuran dasbor-metrik dan nama atribut, min_mem_free_1h, max_mem_used_1h,, avg_disk_used_1h avgMemCached, avg_disk_free_1h, P100, min_cpu_idle_1h, P100. totalDiskWrites CpuUser CpuSystem Perhatikan penggunaan opsional TargetMultiMeasureAttributeName untuk mengganti nama kolom keluaran kueri ke nama atribut berbeda yang digunakan untuk perwujudan hasil.

Berikut ini adalah skema untuk tabel tujuan setelah kueri terjadwal ini terwujud. Seperti yang dapat Anda lihat dari Timestream for tipe LiveAnalytics atribut dalam hasil berikut, hasilnya diwujudkan menjadi catatan multi-ukuran dengan nama ukuran tunggal`dashboard-metrics`, seperti yang ditunjukkan dalam skema pengukuran.

Kolom	Tipe	Timestream untuk jenis LiveAnalytics atribut
region	varchar	DIMENSION

Kolom	Tipe	Timestream untuk jenis LiveAnalytics atribut
ukuran_nama	varchar	MEASURE_NAME
Waktu	timestamp	TIMESTAMP
CpuSystemP100	double	MULTI
avgMemCached	double	MULTI
min_cpu_idle_1h	double	MULTI
avg_disk_free_1h	double	MULTI
avg_disk_used_1h	double	MULTI
totalDiskWrites	double	MULTI
max_mem_used_1h	double	MULTI
min_mem_free_1h	double	MULTI
CpuUserP100	double	MULTI

Berikut ini adalah langkah-langkah terkait yang diperoleh dengan SHOW MEASURES kueri.

ukuran_nama	data_type	Dimensi
dasbor-metrik	multi	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]

Contoh: Menggunakan nama ukuran dari kueri terjadwal dalam catatan multi-ukuran

Dalam contoh ini, Anda akan melihat pembacaan kueri dari tabel dengan catatan ukuran tunggal dan mewujudkan hasilnya menjadi catatan multi-ukuran. Dalam hal ini, hasil kueri terjadwal memiliki kolom yang nilainya dapat digunakan sebagai nama ukuran dalam tabel target tempat hasil kueri terjadwal terwujud. Kemudian Anda dapat menentukan nama ukuran untuk catatan

multi-ukuran di tabel turunan menggunakan MeasureNameColumn properti di TargetConfiguration. TimestreamConfiguration.

```
{
    "Name" : "UsingMeasureNameFromQueryResult",
    "QueryString" : "SELECT region, bin(time, 1h) as hour, measure_name, AVG(CASE WHEN
 measure_name IN ('memory_cached', 'disk_used', 'disk_free') THEN measure_value::double
 ELSE NULL END) as avg_1h, MIN(CASE WHEN measure_name IN ('memory_free', 'cpu_idle')
 THEN measure_value::double ELSE NULL END) as min_1h, SUM(CASE WHEN measure_name
 IN ('disk_io_writes') THEN measure_value::double ELSE NULL END) as sum_1h,
 MAX(CASE WHEN measure_name IN ('memory_used', 'cpu_user', 'cpu_system') THEN
 measure_value::double ELSE NULL END) as max_1h FROM raw_data.devops WHERE time
 BETWEEN bin(@scheduled_runtime, 1h) - 14h AND bin(@scheduled_runtime, 1h) - 2h AND
 measure_name IN ('memory_free', 'memory_used', 'memory_cached', 'disk_io_writes',
 'disk_used', 'disk_free', 'cpu_user', 'cpu_system', 'cpu_idle') GROUP BY region,
 measure_name, bin(time, 1h)",
    "ScheduleConfiguration" : {
        "ScheduleExpression" : "cron(0 0/1 * * ? *)"
    },
    "NotificationConfiguration" : {
        "SnsConfiguration" : {
            "TopicArn" : "******"
        }
    },
    "ScheduledQueryExecutionRoleArn": "******",
    "TargetConfiguration": {
        "TimestreamConfiguration": {
            "DatabaseName" : "derived",
            "TableName" : "dashboard_metrics_1h_agg_2",
            "TimeColumn" : "hour",
            "DimensionMappings" : [
                {
                    "Name": "region",
                    "DimensionValueType" : "VARCHAR"
                }
            ],
            "MeasureNameColumn" : "measure_name",
            "MultiMeasureMappings" : {
                "MultiMeasureAttributeMappings" : [
                    {
                        "SourceColumn" : "avg_1h",
                        "MeasureValueType" : "DOUBLE"
                    },
```

```
                    {
                        "SourceColumn" : "min_1h",
                        "MeasureValueType" : "DOUBLE",
                        "TargetMultiMeasureAttributeName": "p0_1h"
                    },
                    {
                        "SourceColumn" : "sum_1h",
                        "MeasureValueType" : "DOUBLE"
                    },
                    {
                        "SourceColumn" : "max_1h",
                        "MeasureValueType" : "DOUBLE",
                        "TargetMultiMeasureAttributeName": "p100_1h"
                    }
                ]
            }
        }
    },
    "ErrorReportConfiguration": {
        "S3Configuration" : {
            "BucketName" : "******",
            "ObjectKeyPrefix": "errors",
            "EncryptionOption": "SSE_S3"
        }
    }
}
```

Pemetaan dalam contoh ini akan membuat catatan multi-ukuran dengan atribut avg_1h, p0_1h, sum_1h, p100_1h dan akan menggunakan nilai kolom ukuran_name dalam hasil kueri sebagai nama ukuran untuk catatan multi-ukuran di tabel tujuan. Selain itu perhatikan bahwa contoh sebelumnya secara opsional menggunakan TargetMultiMeasureAttributeName with subset dari pemetaan untuk mengganti nama atribut. Misalnya, min_1h diubah namanya menjadi p0_1h dan max_1h diubah namanya menjadi p100_1h.

Berikut ini adalah skema untuk tabel tujuan setelah kueri terjadwal ini terwujud. Seperti yang dapat Anda lihat dari Timestream untuk jenis LiveAnalytics atribut dalam hasil berikut, hasilnya diwujudkan ke dalam catatan multi-ukuran. Jika Anda melihat skema ukuran, ada sembilan nama ukuran berbeda yang dicerna yang sesuai dengan nilai yang terlihat dalam hasil kueri.

Kolom	Tipe	Timestream untuk jenis LiveAnalytics atribut
region	varchar	DIMENSION
ukuran_nama	varchar	MEASURE_NAME
Waktu	timestamp	TIMESTAMP
sum_1h	double	MULTI
p100_1h	double	MULTI
p0_1h	double	MULTI
avg_1h	double	MULTI

Berikut ini adalah langkah-langkah terkait yang diperoleh dengan SHOW MEASURES kueri.

ukuran_nama	data_type	Dimensi
cpu_idle	multi	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]
cpu_sistem	multi	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]
cpu_user	multi	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]
disk_free	multi	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]
disk_io_write	multi	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]
disk_used	multi	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]

ukuran_nama	data_type	Dimensi
memory_cache	multi	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]
memory_free	multi	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]
memory_free	multi	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]

Contoh: Memetakan hasil ke catatan multi-ukuran yang berbeda dengan atribut yang berbeda

Contoh berikut menunjukkan bagaimana Anda dapat memetakan kolom yang berbeda dalam hasil kueri Anda ke dalam catatan multi-ukuran yang berbeda dengan nama ukuran yang berbeda. Jika Anda melihat definisi kueri terjadwal berikut, hasil kueri memiliki kolom berikut: wilayah, jam, avg_mem_cached_1h, min_mem_free_1h, max_mem_used_1h, total_disk_io_writes_1h, avg_disk_used_1h, avg_disk_free_1h, max_pu_user__1h, max_cpu_system_1h, min_cpu_system_1h. `region`dipetakan ke dimensi, dan `hour` dipetakan ke kolom waktu.

MixedMeasureMappings Properti di TargetConfiguration. TimestreamConfigurationmenentukan cara memetakan ukuran ke catatan multi-ukuran dalam tabel turunan.

Dalam contoh khusus ini, avg_mem_cached_1h, min_mem_free_1h, max_mem_used_1h digunakan dalam satu catatan multi-ukuran dengan nama ukuran mem_aggregates, total_disk_io_writes_1h, avg_disk_used_1h, avg_disk_free_1h digunakan dalam catatan multi-ukuran lain dengan nama ukuran dari disk_aggregates, dan akhirnya max_cpu_user_1h, max_cpu_system_1h, min_cpu_system_1h digunakan dalam catatan multi-ukuran lain dengan nama ukuran cpu_aggregates.

Dalam pemetaan ini, Anda juga dapat secara opsional menggunakan TargetMultiMeasureAttributeName untuk mengganti nama kolom hasil kueri agar memiliki nama atribut yang berbeda di tabel tujuan. Misalnya, kolom hasil avg_mem_cached_1h diganti namanya menjadi, total_disk_io_writes_1h diganti namanya menjadi, dll. avgMemCached totalIOWrites

Saat Anda mendefinisikan pemetaan untuk catatan multi-ukuran, Timestream untuk LiveAnalytics memeriksa setiap baris dalam hasil kueri dan secara otomatis mengabaikan nilai kolom yang memiliki

nilai. NULL Akibatnya, dalam kasus pemetaan dengan beberapa nama ukuran, jika semua nilai kolom untuk grup tersebut dalam pemetaan adalah untuk baris tertentu, maka tidak ada nilai NULL untuk nama ukuran itu yang dicerna untuk baris itu.

Misalnya, dalam pemetaan berikut, avg_mem_cached_1h, min_mem_free_1h, dan max_mem_used_1h dipetakan untuk mengukur nama mem_aggregates. Jika untuk baris hasil kueri tertentu, semua nilai kolom ini adalahNULL, Timestream for tidak LiveAnalytics akan menyerap mem_aggregates ukuran untuk baris itu. Jika semua sembilan kolom untuk baris tertentu adalahNULL, maka Anda akan melihat kesalahan pengguna dilaporkan dalam laporan kesalahan Anda.

```
{
    "Name" : "AggsInDifferentMultiMeasureRecords",
    "QueryString" : "SELECT region, bin(time, 1h) as hour, AVG(CASE WHEN measure_name
 = 'memory_cached' THEN measure_value::double ELSE NULL END) as avg_mem_cached_1h,
 MIN(CASE WHEN measure_name = 'memory_free' THEN measure_value::double ELSE
 NULL END) as min_mem_free_1h, MAX(CASE WHEN measure_name = 'memory_used' THEN
 measure_value::double ELSE NULL END) as max_mem_used_1h, SUM(CASE WHEN measure_name =
 'disk_io_writes' THEN measure_value::double ELSE NULL END) as total_disk_io_writes_1h,
 AVG(CASE WHEN measure_name = 'disk_used' THEN measure_value::double ELSE NULL END) as
 avg_disk_used_1h, AVG(CASE WHEN measure_name = 'disk_free' THEN measure_value::double
 ELSE NULL END) as avg_disk_free_1h, MAX(CASE WHEN measure_name = 'cpu_user' THEN
 measure_value::double ELSE NULL END) as max_cpu_user_1h, MAX(CASE WHEN measure_name
 = 'cpu_system' THEN measure_value::double ELSE NULL END) as max_cpu_system_1h,
 MIN(CASE WHEN measure_name = 'cpu_idle' THEN measure_value::double ELSE NULL END)
 as min_cpu_system_1h FROM raw_data.devops WHERE time BETWEEN bin(@scheduled_runtime,
 1h) - 14h AND bin(@scheduled_runtime, 1h) - 2h AND measure_name IN ('memory_cached',
 'memory_free', 'memory_used', 'disk_io_writes', 'disk_used', 'disk_free', 'cpu_user',
 'cpu_system', 'cpu_idle') GROUP BY region, bin(time, 1h)",
    "ScheduleConfiguration" : {
        "ScheduleExpression" : "cron(0 0/1 * * ? *)"
    },
    "NotificationConfiguration" : {
        "SnsConfiguration" : {
            "TopicArn" : "******"
        }
    },
    "ScheduledQueryExecutionRoleArn": "******",
    "TargetConfiguration": {
        "TimestreamConfiguration": {
            "DatabaseName" : "derived",
            "TableName" : "dashboard_metrics_1h_agg_3",
```

```
            "TimeColumn" : "hour",
            "DimensionMappings" : [
                {
                    "Name": "region",
                    "DimensionValueType" : "VARCHAR"
                }
            ],
            "MixedMeasureMappings" : [
                {
                    "MeasureValueType" : "MULTI",
                    "TargetMeasureName" : "mem_aggregates",
                    "MultiMeasureAttributeMappings" : [
                        {
                            "SourceColumn" : "avg_mem_cached_1h",
                            "MeasureValueType" : "DOUBLE",
                            "TargetMultiMeasureAttributeName": "avgMemCached"
                        },
                        {
                            "SourceColumn" : "min_mem_free_1h",
                            "MeasureValueType" : "DOUBLE"
                        },
                        {
                            "SourceColumn" : "max_mem_used_1h",
                            "MeasureValueType" : "DOUBLE",
                            "TargetMultiMeasureAttributeName": "maxMemUsed"
                        }
                    ]
                },
                {
                    "MeasureValueType" : "MULTI",
                    "TargetMeasureName" : "disk_aggregates",
                    "MultiMeasureAttributeMappings" : [
                        {
                            "SourceColumn" : "total_disk_io_writes_1h",
                            "MeasureValueType" : "DOUBLE",
                            "TargetMultiMeasureAttributeName": "totalIOWrites"
                        },
                        {
                            "SourceColumn" : "avg_disk_used_1h",
                            "MeasureValueType" : "DOUBLE"
                        },
                        {
                            "SourceColumn" : "avg_disk_free_1h",
                            "MeasureValueType" : "DOUBLE"
```

```
                        }
                    ]
                },
                {
                    "MeasureValueType" : "MULTI",
                    "TargetMeasureName" : "cpu_aggregates",
                    "MultiMeasureAttributeMappings" : [
                        {
                            "SourceColumn" : "max_cpu_user_1h",
                            "MeasureValueType" : "DOUBLE"
                        },
                        {
                            "SourceColumn" : "max_cpu_system_1h",
                            "MeasureValueType" : "DOUBLE"
                        },
                        {
                            "SourceColumn" : "min_cpu_idle_1h",
                            "MeasureValueType" : "DOUBLE",
                            "TargetMultiMeasureAttributeName": "minCpuIdle"
                        }
                    ]
                }
            ]
        }
    },
    "ErrorReportConfiguration": {
        "S3Configuration" : {
            "BucketName" : "******",
            "ObjectKeyPrefix": "errors",
            "EncryptionOption": "SSE_S3"
        }
    }
}
```

Berikut ini adalah skema untuk tabel tujuan setelah kueri terjadwal ini terwujud.

Kolom	Tipe	Timestream untuk jenis LiveAnalytics atribut
region	varchar	DIMENSION
ukuran_nama	varchar	MEASURE_NAME

Kolom	Tipe	Timestream untuk jenis LiveAnalytics atribut
Waktu	timestamp	TIMESTAMP
minCpuIdle	double	MULTI
max_cpu_system_1h	double	MULTI
max_cpu_user_1h	double	MULTI
avgMemCached	double	MULTI
maxMemUsed	double	MULTI
min_mem_free_1h	double	MULTI
avg_disk_free_1h	double	MULTI
avg_disk_used_1h	double	MULTI
totalIOWrites	double	MULTI

Berikut ini adalah langkah-langkah terkait yang diperoleh dengan SHOW MEASURES kueri.

ukuran_nama	data_type	Dimensi
cpu_agregat	multi	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]
disk_agregat	multi	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]
mem_agregat	multi	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]

Contoh: Memetakan hasil ke catatan ukuran tunggal dengan nama ukuran dari hasil kueri

Berikut ini adalah contoh dari query terjadwal yang hasilnya diwujudkan ke dalam catatan ukuran tunggal. Dalam contoh ini, hasil kueri memiliki kolom ukuran_name yang nilainya akan digunakan sebagai nama ukuran dalam tabel target. Anda menggunakan MixedMeasureMappings atribut di TargetConfiguration. TimestreamConfiguration untuk menentukan pemetaan kolom hasil kueri ke ukuran skalar di tabel target.

Dalam definisi contoh berikut, hasil kueri diharapkan sembilan nilai ukuran_name yang berbeda. Anda mencantumkan semua nama ukuran ini dalam pemetaan dan menentukan kolom mana yang akan digunakan untuk nilai ukuran tunggal untuk nama ukuran tersebut. Misalnya, dalam pemetaan ini, jika nama ukuran memory_cached terlihat untuk baris hasil tertentu, maka nilai di kolom avg_1h digunakan sebagai nilai untuk ukuran ketika data ditulis ke tabel target. Anda dapat secara opsional menggunakan TargetMeasureName untuk memberikan nama ukuran baru untuk nilai ini.

```
{
    "Name" : "UsingMeasureNameColumnForSingleMeasureMapping",
    "QueryString" : "SELECT region, bin(time, 1h) as hour, measure_name, AVG(CASE WHEN
 measure_name IN ('memory_cached', 'disk_used', 'disk_free') THEN measure_value::double
 ELSE NULL END) as avg_1h, MIN(CASE WHEN measure_name IN ('memory_free', 'cpu_idle')
 THEN measure_value::double ELSE NULL END) as min_1h, SUM(CASE WHEN measure_name
 IN ('disk_io_writes') THEN measure_value::double ELSE NULL END) as sum_1h,
 MAX(CASE WHEN measure_name IN ('memory_used', 'cpu_user', 'cpu_system') THEN
 measure_value::double ELSE NULL END) as max_1h FROM raw_data.devops WHERE time
 BETWEEN bin(@scheduled_runtime, 1h) - 14h AND bin(@scheduled_runtime, 1h) - 2h AND
 measure_name IN ('memory_free', 'memory_used', 'memory_cached', 'disk_io_writes',
 'disk_used', 'disk_free', 'cpu_user', 'cpu_system', 'cpu_idle') GROUP BY region,
 bin(time, 1h), measure_name",
    "ScheduleConfiguration" : {
        "ScheduleExpression" : "cron(0 0/1 * * ? *)"
    },
    "NotificationConfiguration" : {
        "SnsConfiguration" : {
            "TopicArn" : "******"
        }
    },
    "ScheduledQueryExecutionRoleArn": "******",
    "TargetConfiguration": {
        "TimestreamConfiguration": {
            "DatabaseName" : "derived",
```

```
            "TableName" : "dashboard_metrics_1h_agg_4",
            "TimeColumn" : "hour",
            "DimensionMappings" : [
                {
                    "Name": "region",
                    "DimensionValueType" : "VARCHAR"
                }
            ],
            "MeasureNameColumn" : "measure_name",
            "MixedMeasureMappings" : [
                {
                    "MeasureName" : "memory_cached",
                    "MeasureValueType" : "DOUBLE",
                    "SourceColumn" : "avg_1h",
                    "TargetMeasureName" : "AvgMemCached"
                },
                {
                    "MeasureName" : "disk_used",
                    "MeasureValueType" : "DOUBLE",
                    "SourceColumn" : "avg_1h"
                },
                {
                    "MeasureName" : "disk_free",
                    "MeasureValueType" : "DOUBLE",
                    "SourceColumn" : "avg_1h"
                },
                {
                    "MeasureName" : "memory_free",
                    "MeasureValueType" : "DOUBLE",
                    "SourceColumn" : "min_1h",
                    "TargetMeasureName" : "MinMemFree"
                },
                {
                    "MeasureName" : "cpu_idle",
                    "MeasureValueType" : "DOUBLE",
                    "SourceColumn" : "min_1h"
                },
                {
                    "MeasureName" : "disk_io_writes",
                    "MeasureValueType" : "DOUBLE",
                    "SourceColumn" : "sum_1h",
                    "TargetMeasureName" : "total-disk-io-writes"
                },
                {
```

```
                    "MeasureName" : "memory_used",
                    "MeasureValueType" : "DOUBLE",
                    "SourceColumn" : "max_1h",
                    "TargetMeasureName" : "maxMemUsed"
                },
                {
                    "MeasureName" : "cpu_user",
                    "MeasureValueType" : "DOUBLE",
                    "SourceColumn" : "max_1h"
                },
                {
                    "MeasureName" : "cpu_system",
                    "MeasureValueType" : "DOUBLE",
                    "SourceColumn" : "max_1h"
                }
            ]
        }
    },
    "ErrorReportConfiguration": {
        "S3Configuration" : {
            "BucketName" : "******",
            "ObjectKeyPrefix": "errors",
            "EncryptionOption": "SSE_S3"
        }
    }
}
```

Berikut ini adalah skema untuk tabel tujuan setelah kueri terjadwal ini terwujud. Seperti yang Anda lihat dari skema, tabel menggunakan catatan ukuran tunggal. Jika Anda mencantumkan skema ukuran untuk tabel, Anda akan melihat sembilan ukuran yang ditulis berdasarkan pemetaan yang disediakan dalam spesifikasi.

Kolom	Tipe	Timestream untuk jenis LiveAnalytics atribut
region	varchar	DIMENSION
ukuran_nama	varchar	MEASURE_NAME
Waktu	timestamp	TIMESTAMP
ukuran_nilai: :ganda	double	MEASURE_VALUE

Berikut ini adalah langkah-langkah terkait yang diperoleh dengan SHOW MEASURES kueri.

ukuran_nama	data_type	Dimensi
AvgMemCached	double	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]
MinMemFree	double	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]
cpu_idle	double	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]
cpu_sistem	double	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]
cpu_user	double	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]
disk_free	double	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]
disk_used	double	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]
maxMemUsed	double	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]
total-disk-io-writes	double	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]

Contoh: Memetakan hasil ke catatan ukuran tunggal dengan kolom hasil kueri sebagai nama ukuran

Dalam contoh ini, Anda memiliki kueri yang hasilnya tidak memiliki kolom nama ukuran. Sebagai gantinya, Anda menginginkan nama kolom hasil kueri sebagai nama ukuran saat memetakan output ke catatan ukuran tunggal. Sebelumnya ada contoh di mana hasil serupa ditulis ke catatan multi-ukuran. Dalam contoh ini, Anda akan melihat cara memetakannya ke catatan ukuran tunggal jika itu sesuai dengan skenario aplikasi Anda.

Sekali lagi, Anda menentukan pemetaan ini menggunakan MixedMeasureMappings properti di TargetConfiguration. TimestreamConfiguration. Dalam contoh berikut, Anda melihat bahwa hasil query memiliki sembilan kolom. Anda menggunakan kolom hasil sebagai nama ukuran dan nilai sebagai nilai ukuran tunggal.

Misalnya, untuk baris tertentu dalam hasil kueri, nama kolom avg_mem_cached_1h digunakan sebagai nama kolom dan nilai yang terkait dengan kolom, dan avg_mem_cached_1h digunakan sebagai nilai ukuran untuk catatan ukuran tunggal. Anda juga dapat menggunakan TargetMeasureName untuk menggunakan nama ukuran yang berbeda di tabel target. Misalnya, untuk nilai di kolom sum_1h, pemetaan menentukan untuk menggunakan total_disk_io_writes_1h sebagai nama ukuran dalam tabel target. Jika ada nilai kolomNULL, maka ukuran yang sesuai diabaikan.

```
{
    "Name" : "SingleMeasureMappingWithoutMeasureNameColumnInQueryResult",
    "QueryString" : "SELECT region, bin(time, 1h) as hour, AVG(CASE WHEN measure_name
 = 'memory_cached' THEN measure_value::double ELSE NULL END) as avg_mem_cached_1h,
 AVG(CASE WHEN measure_name = 'disk_used' THEN measure_value::double ELSE NULL END) as
 avg_disk_used_1h, AVG(CASE WHEN measure_name = 'disk_free' THEN measure_value::double
 ELSE NULL END) as avg_disk_free_1h, MIN(CASE WHEN measure_name = 'memory_free' THEN
 measure_value::double ELSE NULL END) as min_mem_free_1h, MIN(CASE WHEN measure_name =
 'cpu_idle' THEN measure_value::double ELSE NULL END) as min_cpu_idle_1h, SUM(CASE WHEN
 measure_name = 'disk_io_writes' THEN measure_value::double ELSE NULL END) as sum_1h,
 MAX(CASE WHEN measure_name = 'memory_used' THEN measure_value::double ELSE NULL END)
 as max_mem_used_1h, MAX(CASE WHEN measure_name = 'cpu_user' THEN measure_value::double
 ELSE NULL END) as max_cpu_user_1h, MAX(CASE WHEN measure_name = 'cpu_system' THEN
 measure_value::double ELSE NULL END) as max_cpu_system_1h FROM raw_data.devops WHERE
 time BETWEEN bin(@scheduled_runtime, 1h) - 14h AND bin(@scheduled_runtime, 1h) - 2h
 AND measure_name IN ('memory_free', 'memory_used', 'memory_cached', 'disk_io_writes',
 'disk_used', 'disk_free', 'cpu_user', 'cpu_system', 'cpu_idle') GROUP BY region,
 bin(time, 1h)",
    "ScheduleConfiguration" : {
        "ScheduleExpression" : "cron(0 0/1 * * ? *)"
    },
    "NotificationConfiguration" : {
        "SnsConfiguration" : {
            "TopicArn" : "******"
        }
    },
    "ScheduledQueryExecutionRoleArn": "******",
    "TargetConfiguration": {
        "TimestreamConfiguration": {
```

```
            "DatabaseName" : "derived",
            "TableName" : "dashboard_metrics_1h_agg_5",
            "TimeColumn" : "hour",
            "DimensionMappings" : [
                {
                    "Name": "region",
                    "DimensionValueType" : "VARCHAR"
                }
            ],
            "MixedMeasureMappings" : [
                {
                    "MeasureValueType" : "DOUBLE",
                    "SourceColumn" : "avg_mem_cached_1h"
                },
                {
                    "MeasureValueType" : "DOUBLE",
                    "SourceColumn" : "avg_disk_used_1h"
                },
                {
                    "MeasureValueType" : "DOUBLE",
                    "SourceColumn" : "avg_disk_free_1h"
                },
                {
                    "MeasureValueType" : "DOUBLE",
                    "SourceColumn" : "min_mem_free_1h"
                },
                {
                    "MeasureValueType" : "DOUBLE",
                    "SourceColumn" : "min_cpu_idle_1h"
                },
                {
                    "MeasureValueType" : "DOUBLE",
                    "SourceColumn" : "sum_1h",
                    "TargetMeasureName" : "total_disk_io_writes_1h"
                },
                {
                    "MeasureValueType" : "DOUBLE",
                    "SourceColumn" : "max_mem_used_1h"
                },
                {
                    "MeasureValueType" : "DOUBLE",
                    "SourceColumn" : "max_cpu_user_1h"
                },
                {
```

```
                    "MeasureValueType" : "DOUBLE",
                    "SourceColumn" : "max_cpu_system_1h"
                }
            ]
        }
    },
    "ErrorReportConfiguration": {
        "S3Configuration" : {
            "BucketName" : "******",
            "ObjectKeyPrefix": "errors",
            "EncryptionOption": "SSE_S3"
        }
    }
}
```

Berikut ini adalah skema untuk tabel tujuan setelah kueri terjadwal ini terwujud. Seperti yang Anda lihat bahwa tabel target menyimpan catatan dengan nilai ukuran tunggal tipe ganda. Demikian pula, skema ukuran untuk tabel menunjukkan sembilan nama ukuran. Perhatikan juga bahwa nama ukuran total_disk_io_writes_1h hadir sejak pemetaan mengganti nama sum_1h menjadi total_disk_io_writes_1h.

Kolom	Tipe	Timestream untuk jenis LiveAnalytics atribut
region	varchar	DIMENSION
ukuran_nama	varchar	MEASURE_NAME
Waktu	timestamp	TIMESTAMP
ukuran_nilai: :ganda	double	MEASURE_VALUE

Berikut ini adalah langkah-langkah terkait yang diperoleh dengan SHOW MEASURES kueri.

ukuran_nama	data_type	Dimensi
avg_disk_free_1h	double	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]

ukuran_nama	data_type	Dimensi
avg_disk_used_1h	double	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]
avg_mem_cached_1h	double	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]
max_cpu_system_1h	double	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]
max_cpu_user_1h	double	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]
max_mem_used_1h	double	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]
min_cpu_idle_1h	double	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]
min_mem_free_1h	double	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]
total-disk-io-writes	double	[{'dimension_name': 'wilayah', 'data_type': 'varchar'}]

Pesan pemberitahuan kueri terjadwal

Bagian ini menjelaskan pesan yang dikirim oleh Timestream LiveAnalytics saat membuat, menghapus, menjalankan, atau memperbarui status kueri terjadwal.

Nama pesan pemberitahuan	Struktur	Deskripsi
CreatingNotificationMessage	`CreatingNotificationMessage { String arn; NotificationType type;`	Pesan notifikasi ini dikirim sebelum mengirim respons untuk`CreateScheduledQuery` . Kueri

Nama pesan pemberitahuan	Struktur	Deskripsi
	<pre>}</pre>	terjadwal diaktifkan setelah mengirim pemberitahuan ini. arn - ARN Kueri terjadwal yang sedang dibuat. jenis - SCHEDULED _ QUERY _ CREATING

Nama pesan pemberitahuan	Struktur	Deskripsi
UpdateNotificationMessage	`UpdateNotificationMessage { String arn; NotificationType type; QueryState state; }`	Pesan notifikasi ini dikirim saat kueri terjadwal diperbarui. Timestream for LiveAnalytics dapat menonaktifkan kueri terjadwal, secara otomatis, jika terjadi kesalahan yang tidak dapat dipulihkan, seperti: • AssumeRole kegagalan • Setiap kesalahan 4xx yang ditemui saat berkomunikasi dengan KMS ketika KMS kunci yang dikelola pelanggan ditentukan. • Setiap kesalahan 4xx yang ditemui selama menjalankan kueri terjadwal. • Setiap kesalahan 4xx yang ditemui selama menelan hasil kueri arn - ARN Kueri terjadwal yang sedang diperbarui. jenis - SCHEDULED _ QUERY _ UPDATE negara - ENABLED atau DISABLED

Nama pesan pemberitahuan	Struktur	Deskripsi
DeleteNotificationMessage	`DeletionNotificationMessage { String arn; NotificationType type; }`	Pesan pemberitahuan ini dikirim ketika kueri terjadwal telah dihapus. arn - ARN Kueri terjadwal yang sedang dibuat. jenis - SCHEDULED _ QUERY _ DELETED

Nama pesan pemberitahuan	Struktur	Deskripsi

SuccessNotificationMessage	`SuccessNotificationMessage { NotificationType type; String arn; Date nextInvocationEpochSecond; ScheduledQueryRunSummary runSummary; }`
	
	`ScheduledQueryRunSummary { Date invocationTime; Date triggerTime; String runStatus; ExecutionStats executionstats; ErrorReportLocation errorReportLocation; String failureReason; }`
	
	`ExecutionStats { Long bytesMetered; Long dataWrites; Long queryResultRows; Long recordsIngested; Long executionTimeInMillis; }`
	
	`ErrorReportLocation {`
		Pesan notifikasi ini dikirim setelah kueri terjadwal dijalankan dan hasilnya berhasil dicerna.
		
		ARN- ARN Kueri terjadwal yang sedang dihapus.
		
		NotificationType- AUTO _ TRIGGER _ SUCCESS atau MANUAL _ TRIGGER _SUCCESS.
		
		nextInvocationEpochKedua - Waktu berikutnya Timestream untuk LiveAnalytics akan menjalankan query terjadwal.
		
		runSummary- Informasi tentang menjalankan kueri terjadwal.

Nama pesan pemberitahuan	Struktur	Deskripsi

	S3ReportLocation s3ReportLocation;
	}
	
	S3ReportLocation {
	String bucketName;
	String objectKey;
	}	

Nama pesan pemberitahuan	Struktur	Deskripsi

FailureNotificationMessage	`FailureNotificationMessage {`
	`  NotificationType type;`
	`  String arn;`
	`  ScheduledQueryRunSummary runSummary;`
	`}`
	
	`ScheduledQueryRunSummary {`
	`  Date invocationTime;`
	`  Date triggerTime;`
	`  String runStatus;`
	`  ExecutionStats executionstats;`
	`  ErrorReportLocation errorReportLocation;`
	`  String failureReason;`
	`}`
	
	`ExecutionStats {`
	`  Long bytesMetered;`
	`  Long dataWrites;`
	`  Long queryResultRows;`
	`  Long recordsIngested;`
	`  Long executionTimeInMillis;`
	`}`
	
	`ErrorReportLocation {`
	`  S3ReportLocation s3ReportLocation;`
	`}`
		Pesan notifikasi ini dikirim ketika kegagalan ditemui selama menjalankan kueri terjadwal atau saat menelan hasil kueri.
		
		arn - ARN Kueri terjadwal yang sedang dijalankan.
		
		ketik - AUTO TRIGGER _ _ FAILURE atau MANUAL _ TRIGGER _FAILURE.
		
		runSummary- Informasi tentang menjalankan kueri terjadwal.

Nama pesan pemberitahuan	Struktur	Deskripsi
	`S3ReportLocation { String bucketName; String objectKey; }`	

Laporan kesalahan kueri terjadwal

Bagian ini menjelaskan lokasi, format, dan alasan laporan kesalahan yang dihasilkan oleh Timestream LiveAnalytics ketika kesalahan ditemui dengan menjalankan kueri terjadwal.

Topik

Alasan laporan kesalahan kueri terjadwal

Laporan kesalahan dihasilkan untuk kesalahan yang dapat dipulihkan. Laporan kesalahan tidak dibuat untuk kesalahan yang tidak dapat dipulihkan. Timestream for LiveAnalytics dapat menonaktifkan kueri terjadwal secara otomatis ketika kesalahan yang tidak dapat dipulihkan ditemui. Ini termasuk:

- `AssumeRole`kegagalan
- Setiap kesalahan 4xx yang ditemui saat berkomunikasi dengan KMS ketika kunci yang dikelola pelanggan ditentukan KMS
- Kesalahan 4xx apa pun yang ditemui saat kueri terjadwal berjalan
- Setiap kesalahan 4xx yang ditemui selama menelan hasil kueri

Untuk kesalahan yang tidak dapat dipulihkan, Timestream untuk LiveAnalytics mengirimkan pemberitahuan kegagalan dengan pesan kesalahan yang tidak dapat dipulihkan. Pemberitahuan pembaruan juga dikirim yang menunjukkan bahwa kueri terjadwal dinonaktifkan.

Lokasi laporan kesalahan kueri terjadwal

Lokasi laporan kesalahan kueri terjadwal memiliki konvensi penamaan berikut:

```
s3://customer-bucket/customer-prefix/
```

Berikut ini adalah contoh query terjadwalARN:

```
arn:aws:timestream:us-east-1:000000000000:scheduled-query/test-query-hd734tegrgfd
```

```
s3://customer-bucket/customer-prefix/test-query-hd734tegrgfd/<InvocationTime>/<Auto or
 Manual>/<Actual Trigger Time>
```

`Auto` menunjukkan kueri terjadwal yang dijadwalkan secara otomatis oleh Timestream untuk dan LiveAnalytics *`Manual`* menunjukkan kueri terjadwal yang dipicu secara manual oleh pengguna melalui `ExecuteScheduledQuery` API tindakan di Amazon Timestream LiveAnalytics for Query. Untuk informasi lebih lanjut tentang`ExecuteScheduledQuery`, lihat ExecuteScheduledQuery.

Format laporan kesalahan kueri terjadwal

Laporan kesalahan memiliki JSON format berikut:

```
{
    "reportId": <String>,            // A unique string ID for all error reports
 belonging to a particular scheduled query run
    "errors": [ <Error>, ... ],      // One or more errors
}
```

Jenis kesalahan kueri terjadwal

`Error`Objek dapat berupa salah satu dari tiga jenis:

- Kesalahan Pencerapan Rekaman

```
{
    "reason": <String>,              // The error message String
    "records": [ <Record>, ... ],    // One or more rejected records )
```

```
}
```

- Kesalahan Penguraian Baris dan Validasi

```
{
    "reason": <String>,        // The error message String
    "rawLine": <String>,       // [Optional] The raw line String that is being parsed
 into record(s) to be ingested. This line has encountered the above-mentioned parse
 error.
}
```

- Kesalahan Umum

```
{
    "reason": <String>,        // The error message
}
```

Contoh laporan kesalahan kueri terjadwal

Berikut ini adalah contoh laporan kesalahan yang dihasilkan karena kesalahan konsumsi.

```
{
    "reportId": "C9494AABE012D1FBC162A67EA2C18255",
    "errors": [
        {
            "reason": "The record timestamp is outside the time range
 [2021-11-12T14:18:13.354Z, 2021-11-12T16:58:13.354Z) of the memory store.",
            "records": [
                {
                    "dimensions": [
                        {
                            "name": "dim0",
                            "value": "d0_1",
                            "dimensionValueType": null
                        },
                        {
                            "name": "dim1",
                            "value": "d1_1",
                            "dimensionValueType": null
                        }
                    ],
                    "measureName": "random_measure_value",
```

```
                    "measureValue": "3.141592653589793",
                    "measureValues": null,
                    "measureValueType": "DOUBLE",
                    "time": "1637166175635000000",
                    "timeUnit": "NANOSECONDS",
                    "version": null
                },
                {
                    "dimensions": [
                        {
                            "name": "dim0",
                            "value": "d0_2",
                            "dimensionValueType": null
                        },
                        {
                            "name": "dim1",
                            "value": "d1_2",
                            "dimensionValueType": null
                        }
                    ],
                    "measureName": "random_measure_value",
                    "measureValue": "6.283185307179586",
                    "measureValues": null,
                    "measureValueType": "DOUBLE",
                    "time": "1637166175636000000",
                    "timeUnit": "NANOSECONDS",
                    "version": null
                },
                {
                    "dimensions": [
                        {
                            "name": "dim0",
                            "value": "d0_3",
                            "dimensionValueType": null
                        },
                        {
                            "name": "dim1",
                            "value": "d1_3",
                            "dimensionValueType": null
                        }
                    ],
                    "measureName": "random_measure_value",
                    "measureValue": "9.42477796076938",
                    "measureValues": null,
```

```
                    "measureValueType": "DOUBLE",
                    "time": "1637166175637000000",
                    "timeUnit": "NANOSECONDS",
                    "version": null
                },
                {
                    "dimensions": [
                        {
                            "name": "dim0",
                            "value": "d0_4",
                            "dimensionValueType": null
                        },
                        {
                            "name": "dim1",
                            "value": "d1_4",
                            "dimensionValueType": null
                        }
                    ],
                    "measureName": "random_measure_value",
                    "measureValue": "12.566370614359172",
                    "measureValues": null,
                    "measureValueType": "DOUBLE",
                    "time": "1637166175638000000",
                    "timeUnit": "NANOSECONDS",
                    "version": null
                }
            ]
        }
    ]
}
```

Pola kueri terjadwal dan contoh

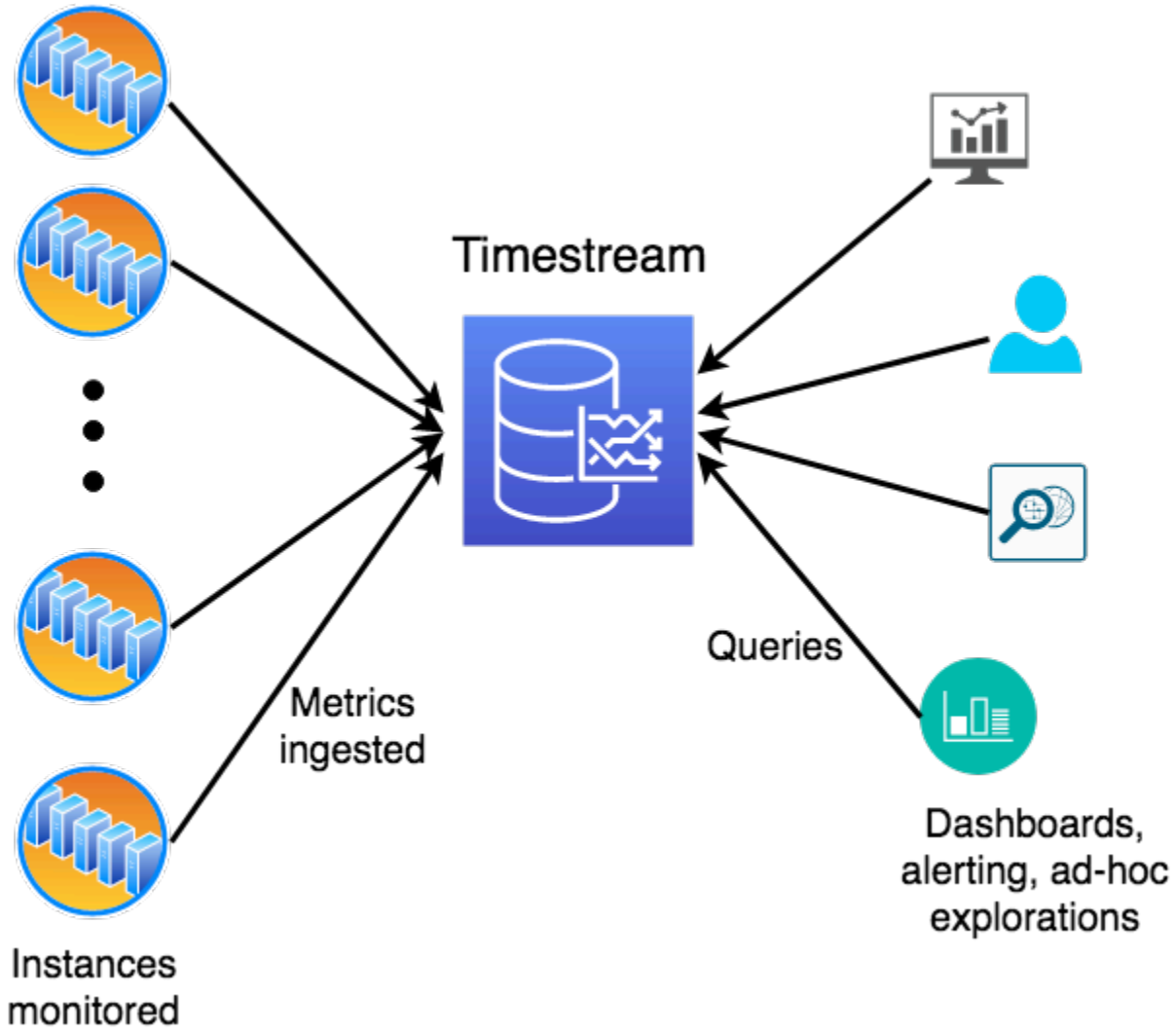
Bagian ini menjelaskan pola penggunaan untuk kueri terjadwal serta end-to-end contoh.

Topik

- Skema sampel kueri terjadwal
- Pola kueri terjadwal
- Contoh kueri terjadwal

Skema sampel kueri terjadwal

Dalam contoh ini kita akan menggunakan contoh aplikasi meniru DevOps skenario pemantauan metrik dari armada besar server. Pengguna ingin memperingatkan tentang penggunaan sumber daya yang anomali, membuat dasbor tentang perilaku dan pemanfaatan armada agregat, dan melakukan analisis canggih pada data terbaru dan historis untuk menemukan korelasi. Diagram berikut memberikan ilustrasi pengaturan di mana satu set instance yang dipantau memancarkan metrik ke Timestream untuk. LiveAnalytics Kumpulan pengguna bersamaan lainnya mengeluarkan kueri untuk peringatan, dasbor, atau analisis ad-hoc, di mana kueri dan konsumsi berjalan secara paralel.



Aplikasi yang dipantau dimodelkan sebagai layanan berskala tinggi yang digunakan di beberapa wilayah di seluruh dunia. Setiap wilayah dibagi lagi menjadi sejumlah unit penskalaan yang disebut sel yang memiliki tingkat isolasi dalam hal infrastruktur di dalam wilayah tersebut. Setiap sel dibagi

lagi menjadi silo, yang mewakili tingkat isolasi perangkat lunak. Setiap silo memiliki lima layanan mikro yang terdiri dari satu contoh layanan yang terisolasi. Setiap microservice memiliki beberapa server dengan tipe instans dan versi OS yang berbeda, yang digunakan di tiga zona ketersediaan. Atribut ini yang mengidentifikasi server yang memancarkan metrik dimodelkan sebagai dimensi di Timestream for. LiveAnalytics Dalam arsitektur ini, kita memiliki hierarki dimensi (seperti region, cell, silo, dan microservice_name) dan dimensi lain yang melintasi hierarki (seperti instance_type dan availability_zone).

Aplikasi ini memancarkan berbagai metrik (seperti cpu_user dan memory_free) dan peristiwa (seperti task_completed dan gc_reclaimed). Setiap metrik atau peristiwa dikaitkan dengan delapan dimensi (seperti wilayah atau sel) yang secara unik mengidentifikasi server yang memancarkannya. Data ditulis dengan 20 metrik yang disimpan bersama dalam catatan multi-ukuran dengan metrik nama ukuran dan semua 5 peristiwa disimpan bersama dalam catatan multi-ukuran lain dengan peristiwa nama ukuran. Model data, skema, dan pembuatan data dapat ditemukan di generator data sumber terbuka. Selain skema dan distribusi data, generator data memberikan contoh penggunaan beberapa penulis untuk menelan data secara paralel, menggunakan penskalaan konsumsi Timestream LiveAnalytics untuk menelan jutaan pengukuran per detik. Di bawah ini kami menunjukkan skema (tabel dan skema ukuran) dan beberapa data sampel dari kumpulan data.

Topik

- Catatan multi-ukuran
- Catatan ukuran tunggal

Catatan multi-ukuran

Skema Tabel

Di bawah ini adalah skema tabel setelah data dicerna menggunakan catatan multi-ukuran. Ini adalah output dari DESCRIBE query. Dengan asumsi data dicerna ke dalam database raw_data dan tabel devops, di bawah ini adalah query.

```
DESCRIBE "raw_data"."devops"
```

Kolom	Tipe	Timestream untuk jenis LiveAnalytics atribut
availability_zone	varchar	DIMENSION

Kolom	Tipe	Timestream untuk jenis LiveAnalytics atribut
microservice_name	varchar	DIMENSION
instance_name	varchar	DIMENSION
process_name	varchar	DIMENSION
os_versi	varchar	DIMENSION
jdk_version	varchar	DIMENSION
sel	varchar	DIMENSION
region	varchar	DIMENSION
silo	varchar	DIMENSION
instance_type	varchar	DIMENSION
ukuran_nama	varchar	MEASURE_NAME
Waktu	timestamp	TIMESTAMP
memory_free	double	MULTI
cpu_mencuri	double	MULTI
cpu_iowait	double	MULTI
cpu_user	double	MULTI
memory_cache	double	MULTI
disk_io_reads	double	MULTI
cpu_hi	double	MULTI
latency_per_read	double	MULTI
network_bytes_out	double	MULTI

Kolom	Tipe	Timestream untuk jenis LiveAnalytics atribut
cpu_idle	double	MULTI
disk_free	double	MULTI
memory_used	double	MULTI
cpu_sistem	double	MULTI
file_descriptors_in_use	double	MULTI
disk_used	double	MULTI
cpu_bagus	double	MULTI
disk_io_write	double	MULTI
cpu_si	double	MULTI
latency_per_write	double	MULTI
network_bytes_in	double	MULTI
task_end_state	varchar	MULTI
gc_jeda	double	MULTI
task_completed	bigint	MULTI
gc_direklamasi	double	MULTI

Skema Ukur

Di bawah ini adalah skema ukuran yang dikembalikan oleh SHOW MEASURES kueri.

```
SHOW MEASURES FROM "raw_data"."devops"
```

ukuran_nama	data_type	Dimensi
peristiwa	multi	[{"data_type” :"varchar”, "dimension_name” :"availab ility_zone "}, {" data_type ” :"varchar”, "dimensio n_name” :"microservice_nam e "}, {" data_type” :"varchar”, "dimension_name” :"instanc e_name "}, {" data_type ” :"instance_name "}, {" data_type” a_type” :"varchar”, "dimension_name” :"process _name "}, {" data_type ” :"varchar”, "dimensio n_name” :"jdk_version "}, {" data_type” :"varchar”, "dimension_name” :"cell "}, {" data_type” :"varchar”, "dimension_name” :"region "}, {" data_type” :"varchar”, "dimension_name” :"silo "}]
metrik	multi	[{"data_type” :"varchar”, "dimension_name” :"availab ility_zone "}, {" data_type ” :"varchar”, "dimensio n_name” :"microservice_nam e "}, {" data_type” :"varchar”, "dimension_name” :"instanc e_name "}, {" data_type ” :"instance_name "}, {" data_type” a_type” :"varchar”, "dimension_name” :"os_vers ion "}, {" data_type” :"varchar ”, "dimension_name” :"cell

ukuran_nama	data_type	Dimensi
		"}, {" data_type” :"varchar”, "dimension_name” :"region "}, {" data_type” :"varchar”, "dimension_name” :"varchar”, "dimension_name” :"varchar” _name” :"silo "}, {" data_type ” :"varchar”, "dimensio n_name” :"instance_type "}]

Contoh Data

re	S	S	a	m	in	in	o	p	jo	u	W	c	c	c	c	c	c	c	c	m	m	d	la	d	la	d	d	n	n	fil	m	ta	g	ta	ga	ccdimpk
			ity	ic	n	ty		a	o	m			m		ri	t				e	cl	e	e	ri	e			y	y	ri n	e	st		le	a	nasi
u e	ka ti r- s	ka ti r- s - si	ka ti r-	a	i- za - a u - e -2 si 0 a o	m e	A			m	1 1 1	6	0	3	0	0	0	0	0	5	8	5	9	3	2	6	2	8	4	3	5						
u e	ka ti r- s	ka ti r- s - si	ka ti r-	a	i- za - a u -	m e	A			m	1 1 1	5	0	3	0	0	0	0	0	9	3	5	3	2	9	5	3	7	5	6	2						

re	S	S	a	m	in	in	o	p	jo	u	W	c	c	c	c	c	c	c	c	c	m	m	d	la	d	la	d	d	n	n	fil	m	ta	g	ta	ge	ccdimek
					e -2 si 0 a: o																																		
u: e	ka ti r- s	ka ti r- s - si	ka ti r-	a	i- za - a: u: - e: -2 si 0 a: o	m e:	A			m	1 1 1:	4	0	4	0	0	0	0	0	9	4	5	3	8	3	7	6	8	5	4	8						

re	S	S	a	m	in	in	o	p	jo	u	W	c	c	c	c	c	c	c	c	c	m	m	d	la	d	la	d	d	n	n	fil	m	ta	g	ta	ge	c	in
			ity	ic	n	ty		a	o	m			m			ri	t					e	cl	e	e	ri	e			yt	yt	n	e	st			le	am	nasi
u	k	k	k	a	i-	m	A			m	1	3	0	5	0	0	0	0	0	4	7	2	4	4	3	8	6	2	1	2	1						
e	ti	ti	ti		za	e					1																											
	r-	r-	r-		-						1																											
	s	s			a																																	
			si		u																																	
					e																																	
					-2																																	
					s																																	
					si																																	
					0																																	
					a																																	
					o																																	
u	k	k	k	a	i-	m	A			m	1	5	0	3	0	0	0	0	0	5	3	8	5	7	1	8	6	7	6	5	3						
e	ti	ti	ti		za	e					1																											
	r-	r-	r-		-						1																											
	s	s			a																																	
			si		u																																	
					e																																	
					-2																																	
					s																																	
					si																																	
					0																																	
					a																																	
					o																																	

re	S	S	a	m	in	in	o	p	jd	u	W	c	c	c	c	c	c	c	c	c	m	m	d	la	d	la	d	d	n	n	fil	m	ta	g	ta	ge	cc	in	ex
			ity	ic	n	ty		a	o	m			m			ri	t					e	cl	e	e	ri	e			yt	yt	ri	e	st		le	an		
u	k	k	k	a	i-			h	Jl	p	1																						7	S	8	3	6	3	4.8
e	ti	ti	ti		z			g			1																							W						
	r-	r-	r-		a						1																							_						
	s	s			u						1																							R						
					e																																		
					-2																																		
					s																																		
					si																																		
					0																																		
					a																																		
					o																																		
u	k	k	k	a	i-			h	Jl	p	1																						7	S	2	4	7	3	4.5
e	ti	ti	ti		z			g			1																							I						
	r-	r-	r-		a						1																							T						
	s	s			u						1																													
					e																																		
					-2																																		
					s																																		
					si																																		
					0																																		
					a																																		
					o																																		

re	S	S	a	m	in	in	o	p	jd	u	W	c	c	c	c	c	c	c	c	c	m	m	d	la	d	la	d	d	n	n	fil	m	ta	g	ta	ge	cc	ind
			ity	ic	n	ty		a	o	m			m		ri	t					e	cl	e	e	ri	e			yt	yt	ri	e	st		le	an		nasi
u	k	k	k	a	i-			h	Jl	p	1																					6	S	6	2	72	2_3	W
e	ti	ti	ti		z			g			1																						IT					JL
	r-	r-	r-		a						1																						T						
	s	s			-						1																												
					a																																		
		si			u																																		
					-																																		
					e																																		
					-2																																		
					s																																		
					si																																		
					0																																		
					a																																		
					o																																		
u	k	k	k	a	i-			h	Jl	p	1																					2	S	5	1	99	9_	W
e	ti	ti	ti		z			g			1																						IT					JL
	r-	r-	r-		a						1																						T						
	s	s			-																																		
					a																																		
		si			u																																		
					-																																		
					e																																		
					-2																																		
					s																																		
					si																																		
					0																																		
					a																																		
					o																																		

re	S	S	a	m	in	in	o	p	jd	u	W	c	c	c	c	c	c	c	c	c	m	m	d	la	d	la	d	d	n	n	fil	m	ta	g	ta	ga	c	m	e
			ity	ic	n	ty		a	o	m			m		ri	t					e	cl	e	e	ri	e			y	y	ri	e	st		le	an			
u	k	k	k	a	i-			h	J	p	1																					3	S	7	2	82	2.9			
e	ti	ti	ti		z			g	o		1																						W								
	r-	r-	r-		a						1																						_								
	s	s			u																												R								
		si			e																																			
					-2																																			
					s																																			
					si																																			
					0																																			
					a																																			
					o																																			

Catatan ukuran tunggal

Timestream for LiveAnalytics juga memungkinkan Anda untuk menelan data dengan satu ukuran per catatan deret waktu. Di bawah ini adalah detail skema saat dicerna menggunakan catatan ukuran tunggal.

Skema Tabel

Di bawah ini adalah skema tabel setelah data dicerna menggunakan catatan multi-ukuran. Ini adalah output dari DESCRIBE query. Dengan asumsi data dicerna ke dalam database raw_data dan tabel devops, di bawah ini adalah query.

```
DESCRIBE "raw_data"."devops_single"
```

Kolom	Tipe	Timestream untuk jenis LiveAnalytics atribut
availability_zone	varchar	DIMENSION

Kolom	Tipe	Timestream untuk jenis LiveAnalytics atribut
microservice_name	varchar	DIMENSION
instance_name	varchar	DIMENSION
process_name	varchar	DIMENSION
os_versi	varchar	DIMENSION
jdk_version	varchar	DIMENSION
sel	varchar	DIMENSION
region	varchar	DIMENSION
silo	varchar	DIMENSION
instance_type	varchar	DIMENSION
ukuran_nama	varchar	MEASURE_NAME
Waktu	timestamp	TIMESTAMP
ukuran_nilai: :ganda	double	MEASURE_VALUE
ukuran_nilai: :bigint	bigint	MEASURE_VALUE
ukuran_nilai: :varchar	varchar	MEASURE_VALUE

Skema Ukur

Di bawah ini adalah skema ukuran yang dikembalikan oleh SHOW MEASURES kueri.

```
SHOW MEASURES FROM "raw_data"."devops_single"
```

ukuran_nama	data_type	Dimensi
cpu_hi	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimensi sion_name': 'os_versi on', 'data_type': 'varchar' }, {'dimension_name': 'sel', 'data_type': 'varchar'}, {'dimension_name': 'wilayah' , 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type', 'data_type'} type ':' varchar '}, {' dimension_name ':' instance_type ',' data_type ':' varchar '}]
cpu_idle	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimensi sion_name': 'os_versi on', 'data_type': 'varchar' }, {'dimension_name': 'sel', 'data_type': 'varchar'}, {'dimension_name': 'wilayah' , 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type', 'data_type'} type ':'

ukuran_nama	data_type	Dimensi
		varchar '}, {' dimension_name ':' instance_type ',' data_type ':' varchar '}]
cpu_iowait	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimensi sion_name': 'os_versi on', 'data_type': 'varchar' }, {'dimension_name': 'sel', 'data_type': 'varchar'}, {'dimension_name': 'wilayah' , 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type', 'data_type'} type ':' varchar '}, {' dimension_name ':' instance_type ',' data_type ':' varchar '}]

ukuran_nama	data_type	Dimensi
cpu_bagus	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimensi sion_name': 'os_versi on', 'data_type': 'varchar' }, {'dimension_name': 'sel', 'data_type': 'varchar'}, {'dimension_name': 'wilayah' , 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type', 'data_type'} type ':' varchar '}, {' dimension_name ':' instance_type ',' data_type ':' varchar '}]

ukuran_nama	data_type	Dimensi
cpu_si	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimensi sion_name': 'os_versi on', 'data_type': 'varchar' }, {'dimension_name': 'sel', 'data_type': 'varchar'}, {'dimension_name': 'wilayah' , 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type', 'data_type'} type ':' varchar '}, {' dimension_name ':' instance_type ',' data_type ':' varchar '}]

ukuran_nama	data_type	Dimensi
cpu_mencuri	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimensi sion_name': 'os_versi on', 'data_type': 'varchar' }, {'dimension_name': 'sel', 'data_type': 'varchar'}, {'dimension_name': 'wilayah' , 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type', 'data_type'} type ':' varchar '}, {' dimension_name ':' instance_type ',' data_type ':' varchar '}]

ukuran_nama	data_type	Dimensi
cpu_sistem	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimensi sion_name': 'os_versi on', 'data_type': 'varchar' }, {'dimension_name': 'sel', 'data_type': 'varchar'}, {'dimension_name': 'wilayah' , 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type', 'data_type'} type ':' varchar '}, {' dimension_name ':' instance_type ',' data_type ':' varchar '}]

ukuran_nama	data_type	Dimensi
cpu_user	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimensi sion_name': 'os_versi on', 'data_type': 'varchar' }, {'dimension_name': 'sel', 'data_type': 'varchar'}, {'dimension_name': 'wilayah' , 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type', 'data_type'} type ':' varchar '}, {' dimension_name ':' instance_type ',' data_type ':' varchar '}]

ukuran_nama	data_type	Dimensi
disk_free	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimensi sion_name': 'os_versi on', 'data_type': 'varchar' }, {'dimension_name': 'sel', 'data_type': 'varchar'}, {'dimension_name': 'wilayah' , 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type', 'data_type'} type ':' varchar '}, {' dimension_name ':' instance_type ',' data_type ':' varchar '}]

ukuran_nama	data_type	Dimensi
disk_io_reads	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimensi sion_name': 'os_versi on', 'data_type': 'varchar' }, {'dimension_name': 'sel', 'data_type': 'varchar'}, {'dimension_name': 'wilayah' , 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type', 'data_type'} type ':' varchar '}, {' dimension_name ':' instance_type ',' data_type ':' varchar '}]

ukuran_nama	data_type	Dimensi
disk_io_write	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimensi sion_name': 'os_versi on', 'data_type': 'varchar' }, {'dimension_name': 'sel', 'data_type': 'varchar'}, {'dimension_name': 'wilayah' , 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type', 'data_type'} type ':' varchar '}, {' dimension_name ':' instance_type ',' data_type ':' varchar '}]

ukuran_nama	data_type	Dimensi
disk_used	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimensi sion_name': 'os_versi on', 'data_type': 'varchar' }, {'dimension_name': 'sel', 'data_type': 'varchar'}, {'dimension_name': 'wilayah' , 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type', 'data_type'} type ':' varchar '}, {' dimension_name ':' instance_type ',' data_type ':' varchar '}]

ukuran_nama	data_type	Dimensi
file_descriptors_in_use	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimensi sion_name': 'os_versi on', 'data_type': 'varchar' }, {'dimension_name': 'sel', 'data_type': 'varchar'}, {'dimension_name': 'wilayah' , 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type', 'data_type'} type ':' varchar '}, {' dimension_name ':' instance_type ',' data_type ':' varchar '}]

ukuran_nama	data_type	Dimensi
gc_jeda	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimensi sion_name': 'process_name', 'data_type': 'varchar'}, {'dimension_name': 'jdk_version', 'data_type': 'varchar'}, {'dimension_name': 'sel', 'data_type': 'varchar'}, {'dimension_name': 'wilayah' , 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}]

ukuran_nama	data_type	Dimensi
gc_direklamasi	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimensi sion_name': 'process_name', 'data_type': 'varchar'}, {'dimension_name': 'jdk_version', 'data_type': 'varchar'}, {'dimension_name': 'sel', 'data_type': 'varchar'}, {'dimension_name': 'wilayah' , 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}]

ukuran_nama	data_type	Dimensi
latency_per_read	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimensi sion_name': 'os_versi on', 'data_type': 'varchar' }, {'dimension_name': 'sel', 'data_type': 'varchar'}, {'dimension_name': 'wilayah' , 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type', 'data_type'} type ':' varchar '}, {' dimension_name ':' instance_type ',' data_type ':' varchar '}]

ukuran_nama	data_type	Dimensi
latency_per_write	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimensi sion_name': 'os_versi on', 'data_type': 'varchar' }, {'dimension_name': 'sel', 'data_type': 'varchar'}, {'dimension_name': 'wilayah' , 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type', 'data_type'} type ':' varchar '}, {' dimension_name ':' instance_type ',' data_type ':' varchar '}]

ukuran_nama	data_type	Dimensi
memory_cache	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimensi sion_name': 'os_versi on', 'data_type': 'varchar' }, {'dimension_name': 'sel', 'data_type': 'varchar'}, {'dimension_name': 'wilayah' , 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type', 'data_type'} type ':' varchar '}, {' dimension_name ':' instance_type ',' data_type ':' varchar '}]

ukuran_nama	data_type	Dimensi
memory_free	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimensi sion_name': 'process_name', 'data_type': 'varchar'}, {'dimension_name': 'os_version', 'data_type': 'varchar'}, {'dimension_name': 'jdk_version', 'data_type': 'varchar'}, {'dimension_name': 'sel', 'data_type': 'varchar'}, {'dimension_name': 'wilayah' , 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'instance _type', 'data_ tipe': 'varchar'}]

ukuran_nama	data_type	Dimensi
memory_used	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimensi sion_name': 'os_versi on', 'data_type': 'varchar' }, {'dimension_name': 'sel', 'data_type': 'varchar'}, {'dimension_name': 'wilayah' , 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type', 'data_type'} type ':' varchar '}, {' dimension_name ':' instance_type ',' data_type ':' varchar '}]

ukuran_nama	data_type	Dimensi
network_bytes_in	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimensi sion_name': 'os_versi on', 'data_type': 'varchar' }, {'dimension_name': 'sel', 'data_type': 'varchar'}, {'dimension_name': 'wilayah' , 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type', 'data_type'} type ':' varchar '}, {' dimension_name ':' instance_type ',' data_type ':' varchar '}]

ukuran_nama	data_type	Dimensi
network_bytes_out	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimensi sion_name': 'os_versi on', 'data_type': 'varchar' }, {'dimension_name': 'sel', 'data_type': 'varchar'}, {'dimension_name': 'wilayah' , 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type', 'data_type'} type ':' varchar '}, {' dimension_name ':' instance_type ',' data_type ':' varchar '}]

ukuran_nama	data_type	Dimensi
task_completed	bigint	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimensi sion_name': 'process_name', 'data_type': 'varchar'}, {'dimension_name': 'jdk_version', 'data_type': 'varchar'}, {'dimension_name': 'sel', 'data_type': 'varchar'}, {'dimension_name': 'wilayah' , 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}]

ukuran_nama	data_type	Dimensi
task_end_state	varchar	[{'dimension_name': 'availability_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimensi sion_name': 'process_name', 'data_type': 'varchar'}, {'dimension_name': 'jdk_version', 'data_type': 'varchar'}, {'dimension_name': 'sel', 'data_type': 'varchar'}, {'dimension_name': 'wilayah', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}]

Contoh Data

availa ity_zo	micro ice_n	instar name	proce ame	os_ve	jdk_v on	Sel	regio	Silo	instar type	ukura ma	Wakt	ukura lai: :g	ukura lai: :b	ukuran_ni lai: :varchar
eu-barat 1-1	hercu	i-zaZsv - hercu eu-west -1-sel-9-silo-2 0000		AL20		eu-barat 1-sel-9	eu-west-	eu-barat 1-sel-9-s ilo-2	r5.4xl e	cpu_l	34:57	0.871		

availa ity_zo	micro ice_n	instar name	proce ame	os_ve	jdk_v on	Sel	regio	Silo	instar type	ukura ma	Wakt	ukura lai: :g	ukura lai: :b	ukuran_ni lai: :varchar
		azona om												
eu-barat-1-1	hercu	i-zaZsv - hercu eu-west -1- sel-9- silo-2 0000 azona om		AL20		eu-barat-1-sel-9	eu-west-	eu-barat-1-sel-9-silo-2	r5.4xl e	cpu_i	34:57	3.462		
eu-barat-1-1	hercu	i-zaZsv - hercu eu-west -1- sel-9- silo-2 0000 azona om		AL20		eu-barat-1-sel-9	eu-west-	eu-barat-1-sel-9-silo-2	r5.4xl e	cpu_i t	34:57	0.102		

availa ity_zo	micro ice_n	instar name	proce ame	os_ve	jdk_v on	Sel	regio	Silo	instar type	ukura ma	Wakt	ukura lai: :g	ukura lai: :b	ukuran_ni lai: :varchar
eu-barat 1-1	hercu	i-zaZsv - hercu eu-west -1- sel-9- silo-2 0000 azona om		AL20		eu-barat 1- sel-9	eu-west-	eu-barat 1- sel-9- s ilo-2	r5.4xl e	cpu_t	34:57	0.630		
eu-barat 1-1	hercu	i-zaZsv - hercu eu-west -1- sel-9- silo-2 0000 azona om		AL20		eu-barat 1- sel-9	eu-west-	eu-barat 1- sel-9- s ilo-2	r5.4xl e	cpu_s	34:57	0,164		

availa ity_zo	micro ice_n	instar name	proce ame	os_ve	jdk_v on	Sel	regio	Silo	instar type	ukura ma	Wakt	ukura lai: :g	ukura lai: :b	ukuran_ni lai: :varchar
eu-barat 1-1	hercu	i-zaZsv - hercu eu-west -1-sel-9- silo-2 0000 azona om		AL20		eu-barat 1-sel-9	eu-west-	eu-barat 1-sel-9- s ilo-2	r5.4xl e	cpu_r ri	34:57	0.107		
eu-barat 1-1	hercu	i-zaZsv - hercu eu-west -1-sel-9- silo-2 0000 azona om		AL20		eu-barat 1-sel-9	eu-west-	eu-barat 1-sel-9- s ilo-2	r5.4xl e	cpu_s m	34:57	0,457		

availa ity_zo	micro ice_n	instar name	proce ame	os_ve	jdk_v on	Sel	regio	Silo	instar type	ukura ma	Wakt	ukura lai: :g	ukura lai: :b	ukuran_ni lai: :varchar
eu- barat 1-1	hercu	i- zaZsv - hercu eu- west -1- sel-9- silo-2 0000 azona om		AL20		eu- barat 1- sel-9	eu- west-	eu- barat 1- sel-9- s ilo-2	r5.4xl e	cpu_u	34:57	94.20		
eu- barat 1-1	hercu	i- zaZsv - hercu eu- west -1- sel-9- silo-2 0000 azona om		AL20		eu- barat 1- sel-9	eu- west-	eu- barat 1- sel-9- s ilo-2	r5.4xl e	disk_	34:57	72.51		

availa ity_zo	micro ice_n	instar name	proce ame	os_ve	jdk_v on	Sel	regio	Silo	instar type	ukura ma	Wakt	ukura lai: :g	ukura lai: :b	ukuran_ni lai: :varchar
eu- barat 1-1	hercu	i- zaZsv - hercu eu- west -1- sel-9- silo-2 0000 azona om		AL20		eu- barat 1- sel-9	eu- west-	eu- barat 1- sel-9- s ilo-2	r5.4xl e	disk_i eads	34:57	81.73		
eu- barat 1-1	hercu	i- zaZsv - hercu eu- west -1- sel-9- silo-2 0000 azona om		AL20		eu- barat 1- sel-9	eu- west-	eu- barat 1- sel-9- s ilo-2	r5.4xl e	disk_i rite	34:57	77.11		

availa ity_zo	micro ice_n	instar name	proce ame	os_ve	jdk_v on	Sel	regio	Silo	instar type	ukura ma	Wakt	ukura lai: :g	ukura lai: :b	ukuran_ni lai: :varchar
eu-barat 1-1	hercu	i-zaZsv - hercu eu-west -1-sel-9- silo-2 0000 azona om		AL20		eu-barat 1-sel-9	eu-west-	eu-barat 1-sel-9- s ilo-2	r5.4xl e	disk_	34:57	89.42		
eu-barat 1-1	hercu	i-zaZsv - hercu eu-west -1-sel-9- silo-2 0000 azona om		AL20		eu-barat 1-sel-9	eu-west-	eu-barat 1-sel-9- s ilo-2	r5.4xl e	file_d riptor n_use	34:57	30.08		

availa ity_zo	micro ice_n	instar name	proce ame	os_ve	jdk_v on	Sel	regio	Silo	instar type	ukura ma	Wakt	ukura lai: :g	ukura lai: :b	ukuran_ni lai: :varchar
eu-barat 1-1	hercu	i-zaZsv - hercu eu-west -1- sel-9- silo-2 0000 azona om	serve		JDK_	eu-barat 1- sel-9	eu-west-	eu-barat 1- sel-9- s ilo-2		gc_je	34:57	60.28		
eu-barat 1-1	hercu	i-zaZsv - hercu eu-west -1- sel-9- silo-2 0000 azona om	serve		JDK_	eu-barat 1- sel-9	eu-west-	eu-barat 1- sel-9- s ilo-2		gc_di amas	34:57	75.28		

availa ity_zo	micro ice_n	instar name	proce ame	os_ve	jdk_v on	Sel	regio	Silo	instar type	ukura ma	Wakt	ukura lai: :g	ukura lai: :b	ukuran_ni lai: :varchar
eu- barat- 1-1	hercu	i- zaZsv - hercu eu- west -1- sel-9- silo-2 0000 azona om		AL20		eu- barat- 1- sel-9	eu- west-	eu- barat- 1- sel-9- s ilo-2	r5.4xl e	laten er_re	34:57	8.076		
eu- barat- 1-1	hercu	i- zaZsv - hercu eu- west -1- sel-9- silo-2 0000 azona om		AL20		eu- barat- 1- sel-9	eu- west-	eu- barat- 1- sel-9- s ilo-2	r5.4xl e	laten er_wr	34:57	58.11		

availa ity_zo	micro ice_n	instar name	proce ame	os_ve	jdk_v on	Sel	regio	Silo	instar type	ukura ma	Wakt	ukura lai: :g	ukura lai: :b	ukuran_ni lai: :varchar
eu- barat- 1-1	hercu	i- zaZsv - hercu eu- west -1- sel-9- silo-2 0000 azona om		AL20		eu- barat- 1- sel-9	eu- west-	eu- barat- 1- sel-9- s ilo-2	r5.4xl e	mem che	34:57	87.56		
eu- barat- 1-1	hercu	i- zaZsv - hercu eu- west -1- sel-9- silo-2 0000 azona om	serve		JDK_	eu- barat- 1- sel-9	eu- west-	eu- barat- 1- sel-9- s ilo-2		mem ee	34:57	18.95		

availa ity_zo	micro ice_n	instar name	proce ame	os_ve	jdk_v on	Sel	regio	Silo	instar type	ukura ma	Wakt	ukura lai: :g	ukura lai: :b	ukuran_ni lai: :varchar
eu-barat 1-1	hercu	i-zaZsv - hercu eu-west -1-sel-9- silo-2 0000 azona om		AL20		eu-barat 1-sel-9	eu-west-	eu-barat 1-sel-9- s ilo-2	r5.4xl e	mem ee	34:57	97.20		
eu-barat 1-1	hercu	i-zaZsv - hercu eu-west -1-sel-9- silo-2 0000 azona om		AL20		eu-barat 1-sel-9	eu-west-	eu-barat 1-sel-9- s ilo-2	r5.4xl e	mem ed	34:57	12.37		

availa ity_zo	micro ice_n	instar name	proce ame	os_ve	jdk_v on	Sel	regio	Silo	instar type	ukura ma	Wakt	ukura lai: :g	ukura lai: :b	ukuran_ni lai: :varchar
eu- barat 1-1	hercu	i- zaZsv - hercu eu- west -1- sel-9- silo-2 0000 azona om		AL20		eu- barat 1- sel-9	eu- west-	eu- barat 1- sel-9- s ilo-2	r5.4xl e	netwo ytes_	34:57	31.02		
eu- barat 1-1	hercu	i- zaZsv - hercu eu- west -1- sel-9- silo-2 0000 azona om		AL20		eu- barat 1- sel-9	eu- west-	eu- barat 1- sel-9- s ilo-2	r5.4xl e	netwo ytes_	34:57	0,514		

availa ity_zo	micro ice_n	instar name	proce ame	os_ve	jdk_v on	Sel	regio	Silo	instar type	ukura ma	Wakt	ukura lai: :g	ukura lai: :b	ukuran_ni lai: :varchar
eu-barat-1-1	hercu	i-zaZsv-hercu eu-west-1-sel-9-silo-20000 azona om	serve		JDK_	eu-barat-1-sel-9	eu-west-	eu-barat-1-sel-9-silo-2		task_ leted	34:57		69	
eu-barat-1-1	hercu	i-zaZsv-hercu eu-west-1-sel-9-silo-20000 azona om	serve		JDK_	eu-barat-1-sel-9	eu-west-	eu-barat-1-sel-9-silo-2		task_ state	34:57			SUCCESS_ ITH_RESUL T

Pola kueri terjadwal

Di bagian ini Anda akan menemukan beberapa pola umum tentang bagaimana Anda dapat menggunakan Amazon Timestream untuk Kueri LiveAnalytics Terjadwal untuk mengoptimalkan dasbor Anda agar memuat lebih cepat dan dengan biaya yang lebih rendah. Contoh di bawah ini menggunakan skenario DevOps aplikasi untuk menggambarkan konsep-konsep kunci yang berlaku untuk query terjadwal secara umum, terlepas dari skenario aplikasi.

Kueri Terjadwal di Timestream untuk LiveAnalytics memungkinkan Anda mengekspresikan kueri Anda menggunakan luas SQL permukaan penuh Timestream untuk. LiveAnalytics Kueri Anda dapat menyertakan satu atau beberapa tabel sumber, melakukan agregasi atau kueri lain yang diizinkan oleh Timestream untuk LiveAnalytics SQL bahasa, dan kemudian mewujudkan hasil kueri di tabel tujuan lain di Timestream for. LiveAnalytics Untuk kemudahan eksposisi, bagian ini mengacu pada tabel target kueri terjadwal ini sebagai tabel turunan.

Berikut ini adalah poin-poin kunci yang dibahas dalam bagian ini.

- Menggunakan agregat tingkat armada sederhana untuk menjelaskan bagaimana Anda dapat menentukan kueri terjadwal dan memahami beberapa konsep dasar.
- Bagaimana Anda dapat menggabungkan hasil dari target kueri terjadwal (tabel turunan) dengan hasil dari tabel sumber untuk mendapatkan manfaat biaya dan kinerja dari kueri terjadwal.
- Apa trade-off Anda saat mengonfigurasi periode penyegaran kueri terjadwal.
- Menggunakan kueri terjadwal untuk beberapa skenario umum.
  - Melacak titik data terakhir dari setiap instance sebelum tanggal tertentu.
  - Nilai yang berbeda untuk dimensi yang akan digunakan untuk mengisi variabel di dasbor.
- Bagaimana Anda menangani data yang terlambat tiba dalam konteks kueri terjadwal.
- Bagaimana Anda dapat menggunakan eksekusi manual satu kali untuk menangani berbagai skenario yang tidak secara langsung dicakup oleh pemicu otomatis untuk kueri terjadwal.

Topik

Skenario

Contoh berikut menggunakan skenario DevOps pemantauan yang diuraikan dalamSkema sampel kueri terjadwal.

Contoh menyediakan definisi kueri terjadwal di mana Anda dapat menyambungkan konfigurasi yang sesuai untuk tempat menerima pemberitahuan status eksekusi untuk kueri terjadwal, tempat menerima laporan untuk kesalahan yang ditemui selama eksekusi kueri terjadwal, dan IAM peran yang digunakan kueri terjadwal untuk melakukan operasinya.

Anda dapat membuat kueri terjadwal ini setelah mengisi opsi sebelumnya, membuat tabel target (atau turunan), dan mengeksekusi melalui. AWS CLI Misalnya, asumsikan bahwa definisi kueri terjadwal disimpan dalam file,`scheduled_query_example.json`. Anda dapat membuat kueri menggunakan CLI perintah.

```
aws timestream-query create-scheduled-query --cli-input-json  file://
scheduled_query_example.json --profile aws_profile --region us-east-1
```

Pada perintah sebelumnya, profil yang diteruskan menggunakan opsi --profile harus memiliki izin yang sesuai untuk membuat kueri terjadwal. Lihat Kebijakan berbasis identitas untuk Kueri Terjadwal untuk petunjuk terperinci tentang kebijakan dan izin.

Agregat tingkat armada sederhana

Contoh pertama ini memandu Anda melalui beberapa konsep dasar saat bekerja dengan kueri terjadwal menggunakan contoh sederhana komputasi agregat tingkat armada. Dengan menggunakan contoh ini, Anda akan mempelajari yang berikut ini.

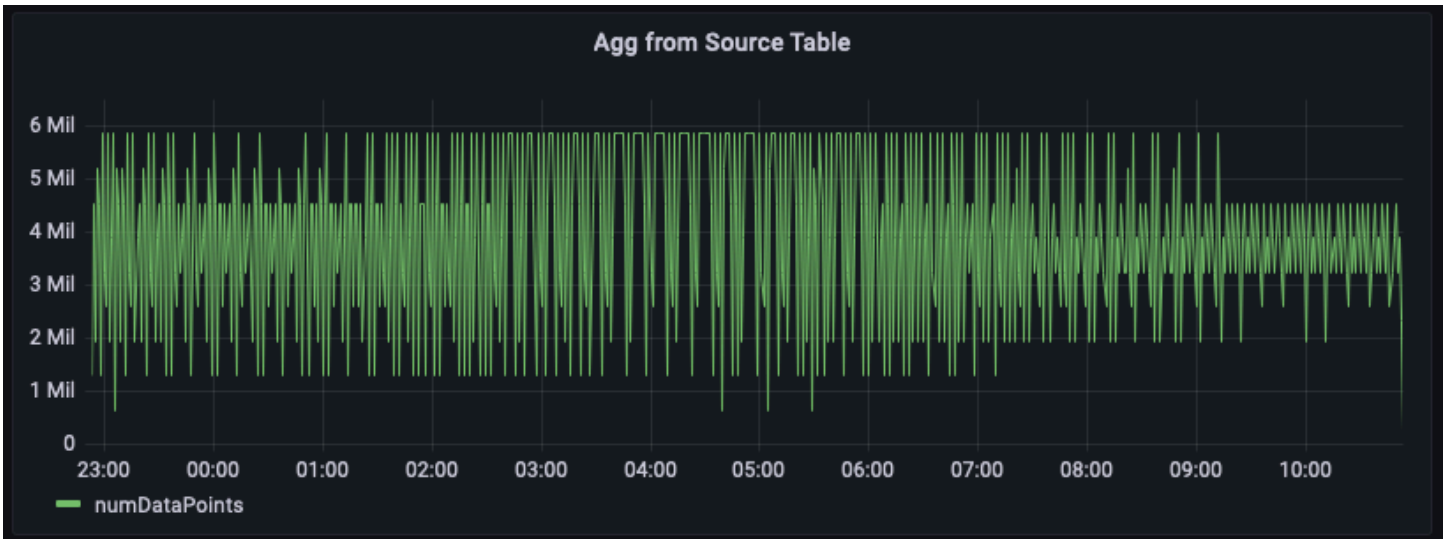
- Cara mengambil kueri dasbor Anda yang digunakan untuk mendapatkan statistik agregat dan memetakannya ke kueri terjadwal.
- Bagaimana Timestream untuk LiveAnalytics mengelola eksekusi instance yang berbeda dari kueri terjadwal Anda.
- Bagaimana Anda dapat memiliki contoh yang berbeda dari kueri terjadwal tumpang tindih dalam rentang waktu dan bagaimana kebenaran data dipertahankan pada tabel target untuk memastikan bahwa dasbor Anda menggunakan hasil kueri terjadwal memberi Anda hasil yang cocok dengan agregat yang sama yang dihitung pada data mentah.
- Cara mengatur rentang waktu dan menyegarkan irama untuk kueri terjadwal Anda.
- Bagaimana Anda dapat melayani sendiri melacak hasil kueri terjadwal untuk menyetelnya sehingga latensi eksekusi untuk instance kueri berada dalam penundaan yang dapat diterima untuk menyegarkan dasbor Anda.

Topik

- Agregat dari tabel sumber
- Kueri terjadwal untuk pra-komputasi agregat
- Agregat dari tabel turunan
- Agregat menggabungkan sumber dan tabel turunan
- Agregat dari komputasi terjadwal yang sering diperbarui

Agregat dari tabel sumber

Dalam contoh ini, Anda melacak jumlah metrik yang dipancarkan oleh server dalam wilayah tertentu dalam setiap menit. Grafik di bawah ini adalah contoh yang memplot deret waktu ini untuk wilayah us-east-1.



Di bawah ini adalah contoh query untuk menghitung agregat ini dari data mentah. Ini menyaring baris untuk wilayah us-east-1 dan kemudian menghitung jumlah per menit dengan menghitung 20 metrik (jika ukuran_name adalah metrik) atau 5 peristiwa (jika pengukuran_nama adalah peristiwa). Dalam contoh ini, ilustrasi grafik menunjukkan bahwa jumlah metrik yang dipancarkan bervariasi antara 1,5 Juta hingga 6 Juta per menit. Saat merencanakan deret waktu ini selama beberapa jam (12 jam terakhir dalam gambar ini), kueri atas data mentah ini menganalisis ratusan juta baris.

```
WITH grouped_data AS (
    SELECT region, bin(time, 1m) as minute, SUM(CASE WHEN measure_name = 'metrics' THEN
 20 ELSE 5 END) as numDataPoints
    FROM "raw_data"."devops"
    WHERE time BETWEEN from_milliseconds(1636699996445) AND
 from_milliseconds(1636743196445)
```

```
        AND region = 'us-east-1'
    GROUP BY region, measure_name, bin(time, 1m)
)
SELECT minute, SUM(numDataPoints) AS numDataPoints
FROM grouped_data
GROUP BY minute
ORDER BY 1 desc, 2 desc
```

Kueri terjadwal untuk pra-komputasi agregat

Jika Anda ingin mengoptimalkan dasbor agar memuat lebih cepat dan menurunkan biaya dengan memindai lebih sedikit data, Anda dapat menggunakan kueri terjadwal untuk menghitung agregat ini terlebih dahulu. Kueri terjadwal di Timestream untuk LiveAnalytics memungkinkan Anda mewujudkan pra-perhitungan ini di Timestream lain untuk LiveAnalytics tabel, yang selanjutnya dapat Anda gunakan untuk dasbor Anda.

Langkah pertama dalam membuat kueri terjadwal adalah mengidentifikasi kueri yang ingin Anda pra-komputasi. Perhatikan bahwa dasbor sebelumnya digambar untuk wilayah us-east-1. Namun, pengguna yang berbeda mungkin menginginkan agregat yang sama untuk wilayah yang berbeda, katakanlah us-west-2 atau eu-west-1. Untuk menghindari pembuatan kueri terjadwal untuk setiap kueri tersebut, Anda dapat menghitung terlebih dahulu agregat untuk setiap wilayah dan mewujudkan agregat per wilayah di Timestream lain untuk tabel. LiveAnalytics

Kueri di bawah ini memberikan contoh pra-komputasi yang sesuai. Seperti yang Anda lihat, ini mirip dengan ekspresi tabel umum grouped_data yang digunakan dalam kueri pada data mentah, kecuali untuk dua perbedaan: 1) tidak menggunakan predikat wilayah, sehingga kita dapat menggunakan satu kueri untuk menghitung sebelumnya untuk semua wilayah; dan 2) menggunakan predikat waktu berparameter dengan parameter khusus @scheduled_runtime yang dijelaskan secara rinci di bawah ini.

```
SELECT region, bin(time, 1m) as minute,
    SUM(CASE WHEN measure_name = 'metrics' THEN 20 ELSE 5 END) as numDataPoints
FROM raw_data.devops
WHERE time BETWEEN @scheduled_runtime - 10m AND @scheduled_runtime + 1m
GROUP BY bin(time, 1m), region
```

Kueri sebelumnya dapat dikonversi menjadi kueri terjadwal menggunakan spesifikasi berikut. Kueri terjadwal diberi Nama, yang merupakan mnemonik yang mudah digunakan. Ini kemudian mencakup QueryString, a ScheduleConfiguration, yang merupakan ekspresi cron. Ini menentukan

TargetConfiguration yang memetakan hasil query ke tabel tujuan di Timestream untuk. LiveAnalytics Akhirnya, ini menentukan sejumlah konfigurasi lain, seperti NotificationConfiguration, di mana pemberitahuan dikirim untuk eksekusi individual kueri, ErrorReportConfiguration di mana laporan ditulis jika kueri menemukan kesalahan, dan ScheduledQueryExecutionRoleArn, yang merupakan peran yang digunakan untuk melakukan operasi untuk kueri terjadwal.

```
{
    "Name": "MultiPT5mPerMinutePerRegionMeasureCount",
    "QueryString": "SELECT region, bin(time, 1m) as minute, SUM(CASE WHEN measure_name
 = 'metrics' THEN 20 ELSE 5 END) as numDataPoints FROM raw_data.devops WHERE time
 BETWEEN @scheduled_runtime - 10m AND @scheduled_runtime + 1m GROUP BY bin(time, 1m),
 region",
    "ScheduleConfiguration": {
        "ScheduleExpression": "cron(0/5 * * * ? *)"
    },
    "NotificationConfiguration": {
        "SnsConfiguration": {
            "TopicArn": "******"
        }
    },
    "TargetConfiguration": {
        "TimestreamConfiguration": {
            "DatabaseName": "derived",
            "TableName": "per_minute_aggs_pt5m",
            "TimeColumn": "minute",
            "DimensionMappings": [
                {
                    "Name": "region",
                    "DimensionValueType": "VARCHAR"
                }
            ],
            "MultiMeasureMappings": {
                "TargetMultiMeasureName": "numDataPoints",
                "MultiMeasureAttributeMappings": [
                    {
                        "SourceColumn": "numDataPoints",
                        "MeasureValueType": "BIGINT"
                    }
                ]
            }
        }
    },
    "ErrorReportConfiguration": {
```

```
        "S3Configuration" : {
            "BucketName" : "******",
            "ObjectKeyPrefix": "errors",
            "EncryptionOption": "SSE_S3"
        }
    },
    "ScheduledQueryExecutionRoleArn": "******"
}
```

Dalam contoh, ScheduleExpression cron (0/5 * * *? *) menyiratkan bahwa kueri dijalankan setiap 5 menit sekali pada tanggal 5, 10, 15,.. menit setiap jam setiap hari. Stempel waktu ini ketika instance spesifik dari kueri ini dipicu adalah apa yang diterjemahkan ke parameter @scheduled_runtime yang digunakan dalam kueri. Misalnya, pertimbangkan contoh kueri terjadwal ini yang dijalankan pada 2021-12-01 00:00:00. Untuk contoh ini, parameter @scheduled_runtime diinisialisasi ke stempel waktu 2021-12-01 00:00:00 saat menjalankan kueri. Oleh karena itu, instance khusus ini akan dijalankan pada stempel waktu 2021-12-01 00:00:00 dan akan menghitung agregat per menit dari rentang waktu 2021-11-30 23:50:00 hingga 2021-12-01 00:01:00. Demikian pula, contoh berikutnya dari kueri ini dipicu pada stempel waktu 2021-12-01 00:05:00 dan dalam hal ini, kueri akan menghitung agregat per menit dari rentang waktu 2021-11-30 23:55:00 hingga 2021-12-01 00:06:00. Oleh karena itu, parameter @scheduled_runtime menyediakan kueri terjadwal untuk menghitung agregat sebelumnya untuk rentang waktu yang dikonfigurasi menggunakan waktu pemanggilan untuk kueri.

Perhatikan bahwa dua contoh kueri berikutnya tumpang tindih dalam rentang waktu mereka. Ini adalah sesuatu yang dapat Anda kontrol berdasarkan kebutuhan Anda. Dalam hal ini, tumpang tindih ini memungkinkan kueri ini untuk memperbarui agregat berdasarkan data apa pun yang kedatangannya sedikit tertunda, hingga 5 menit dalam contoh ini. Untuk memastikan kebenaran kueri yang terwujud, Timestream untuk LiveAnalytics memastikan bahwa kueri pada 2021-12-01 00:05:00 akan dilakukan hanya setelah kueri pada 2021-12-01 00:00:00 selesai dan hasil kueri terakhir dapat memperbarui agregat yang sebelumnya terwujud menggunakan jika nilai yang lebih baru dihasilkan. Misalnya, jika beberapa data pada stempel waktu 2021-11-30 23:59:00 tiba setelah kueri untuk 2021-12-01 00:00:00 dijalankan tetapi sebelum kueri untuk 2021-12-01 00:05:00, maka eksekusi pada 2021-12-01 00:05:00 akan menghitung ulang agregat untuk menit 2021-11-30 23:59:00 dan ini akan menghasilkan agregat sebelumnya diperbarui dengan nilai yang baru dihitung. Anda dapat mengandalkan semantik kueri terjadwal ini untuk mencapai trade-off antara seberapa cepat Anda memperbarui pra-perhitungan Anda versus bagaimana Anda dapat menangani beberapa data dengan anggun dengan kedatangan yang tertunda. Pertimbangan tambahan dibahas di bawah ini tentang bagaimana Anda menukar irama penyegaran ini dengan kesegaran data dan bagaimana

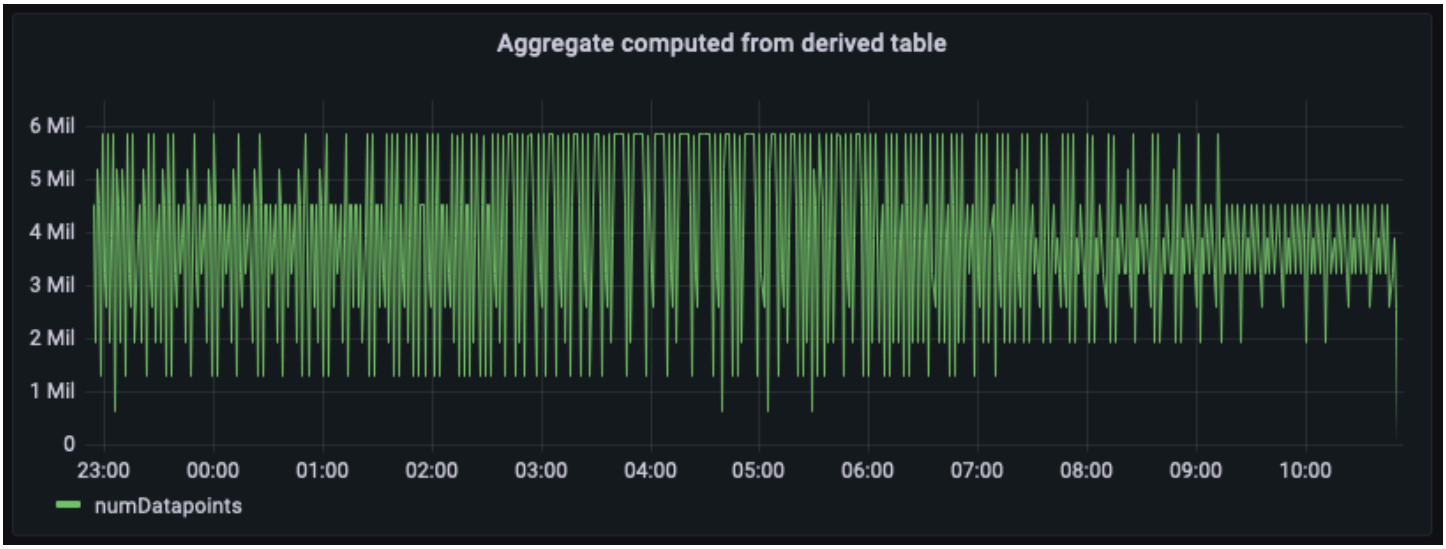
Anda menangani pembaruan agregat untuk data yang datang lebih tertunda atau jika sumber perhitungan terjadwal Anda telah memperbarui nilai yang akan memerlukan agregat untuk dihitung ulang.

Setiap perhitungan terjadwal memiliki konfigurasi notifikasi di mana Timestream untuk LiveAnalytics mengirim pemberitahuan setiap eksekusi konfigurasi terjadwal. Anda dapat mengonfigurasi SNS topik untuk menerima pemberitahuan untuk setiap pemanggilan. Selain status keberhasilan atau kegagalan instance tertentu, ia juga memiliki beberapa statistik seperti waktu yang diperlukan komputasi ini untuk mengeksekusi, jumlah byte komputasi yang dipindai, dan jumlah byte yang ditulis komputasi ke tabel tujuannya. Anda dapat menggunakan statistik ini untuk lebih menyetel kueri, menjadwalkan konfigurasi, atau melacak pengeluaran untuk kueri terjadwal Anda. Salah satu aspek yang perlu diperhatikan adalah waktu eksekusi untuk sebuah instance. Dalam contoh ini, perhitungan terjadwal dikonfigurasi untuk mengeksekusi setiap 5 menit. Waktu eksekusi akan menentukan penundaan dengan mana pra-komputasi akan tersedia, yang juga akan menentukan lag di dasbor Anda saat Anda menggunakan data yang telah dihitung sebelumnya di dasbor Anda. Selanjutnya, jika penundaan ini secara konsisten lebih tinggi daripada interval penyegaran, misalnya, jika waktu eksekusi lebih dari 5 menit untuk komputasi yang dikonfigurasi untuk disegarkan setiap 5 menit, penting untuk menyetel komputasi Anda agar berjalan lebih cepat untuk menghindari kelambatan lebih lanjut di dasbor Anda.

Agregat dari tabel turunan

Sekarang setelah Anda menyiapkan kueri terjadwal dan agregat telah dihitung sebelumnya dan diwujudkan ke Timestream lain untuk LiveAnalytics tabel yang ditentukan dalam konfigurasi target perhitungan terjadwal, Anda dapat menggunakan data dalam tabel tersebut untuk menulis kueri untuk memberi daya pada dasbor Anda. SQL Di bawah ini adalah setara dengan kueri yang menggunakan pra-agregat terwujud untuk menghasilkan agregat jumlah titik data per menit untuk us-east-1.

```
SELECT bin(time, 1m) as minute, SUM(numDataPoints) as numDatapoints
FROM "derived"."per_minute_aggs_pt5m"
WHERE time BETWEEN from_milliseconds(1636699996445) AND
 from_milliseconds(1636743196445)
    AND region = 'us-east-1'
GROUP BY bin(time, 1m)
ORDER BY 1 desc
```



Gambar sebelumnya memplot agregat yang dihitung dari tabel agregat. Membandingkan panel ini dengan panel yang dihitung dari data sumber mentah, Anda akan melihat bahwa mereka cocok persis, meskipun agregat ini tertunda beberapa menit, dikendalikan oleh interval penyegaran yang Anda konfigurasikan untuk perhitungan terjadwal ditambah waktu untuk menjalankannya.

Kueri atas data yang dihitung sebelumnya memindai beberapa kali lipat data yang lebih kecil dibandingkan dengan agregat yang dihitung melalui data sumber mentah. Bergantung pada perincian agregasi, pengurangan ini dapat dengan mudah menghasilkan biaya 100X lebih rendah dan latensi kueri. Ada biaya untuk melaksanakan perhitungan terjadwal ini. Namun, tergantung pada seberapa sering dasbor ini disegarkan dan berapa banyak pengguna bersamaan yang memuat dasbor ini, Anda akhirnya mengurangi biaya keseluruhan secara signifikan dengan menggunakan pra-perhitungan ini. Dan ini di atas waktu muat 10-100X lebih cepat untuk dasbor.

Agregat menggabungkan sumber dan tabel turunan

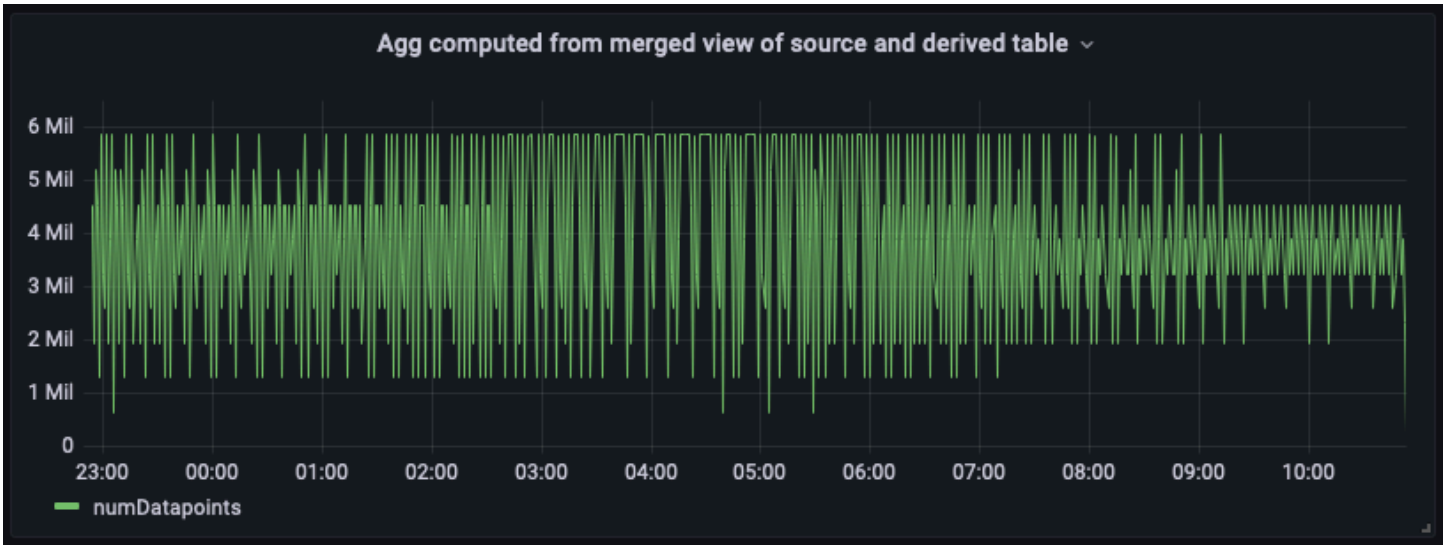
Dasbor yang dibuat menggunakan tabel turunan dapat memiliki lag. Jika skenario aplikasi Anda memerlukan dasbor untuk memiliki data terbaru, maka Anda dapat menggunakan kekuatan dan fleksibilitas SQL dukungan Timestream for untuk LiveAnalytics menggabungkan data terbaru dari tabel sumber dengan agregat historis dari tabel turunan untuk membentuk tampilan gabungan. Tampilan gabungan ini menggunakan semantik gabungan SQL dan rentang waktu yang tidak tumpang tindih dari sumber dan tabel turunan. Pada contoh di bawah ini, kita menggunakan “turunan”.” per_minute_aggs_pt5m” tabel turunan. Karena perhitungan terjadwal untuk tabel turunan tersebut menyegarkan setiap 5 menit sekali (sesuai spesifikasi ekspresi jadwal), kueri di bawah ini menggunakan 15 menit data terbaru dari tabel sumber, dan data apa pun yang lebih lama dari 15 menit dari tabel turunan dan kemudian menyatukan hasil untuk membuat tampilan gabungan yang memiliki yang terbaik dari kedua dunia: ekonomi dan latensi rendah dengan membaca agregat

pra-komputasi yang lebih lama dari tabel turunan dan kesegaran agregat dari tabel sumber untuk mendukung kasus penggunaan analitik waktu nyata Anda.

Perhatikan bahwa pendekatan gabungan ini akan memiliki latensi kueri yang sedikit lebih tinggi dibandingkan dengan hanya menanyakan tabel turunan dan juga memiliki data yang sedikit lebih tinggi yang dipindai, karena menggabungkan data mentah secara real time untuk mengisi interval waktu terbaru. Namun, tampilan gabungan ini masih akan jauh lebih cepat dan lebih murah dibandingkan dengan agregasi dengan cepat dari tabel sumber, terutama untuk dasbor yang merender data berhari-hari atau berminggu-minggu. Anda dapat menyetel rentang waktu untuk contoh ini untuk menyesuaikan kebutuhan penyegaran aplikasi Anda dan toleransi penundaan.

```
WITH aggregated_source_data AS (
    SELECT bin(time, 1m) as minute, SUM(CASE WHEN measure_name = 'metrics' THEN 20 ELSE
 5 END) as numDatapoints
    FROM "raw_data"."devops"
    WHERE time BETWEEN bin(from_milliseconds(1636743196439), 1m) - 15m AND
 from_milliseconds(1636743196439)
        AND region = 'us-east-1'
    GROUP BY bin(time, 1m)
), aggregated_derived_data AS (
    SELECT bin(time, 1m) as minute, SUM(numDataPoints) as numDatapoints
    FROM "derived"."per_minute_aggs_pt5m"
    WHERE time BETWEEN from_milliseconds(1636699996439) AND
 bin(from_milliseconds(1636743196439), 1m) - 15m
        AND region = 'us-east-1'
    GROUP BY bin(time, 1m)
)
SELECT minute, numDatapoints
FROM (
    (
    SELECT *
    FROM aggregated_derived_data
    )
    UNION
    (
    SELECT *
    FROM aggregated_source_data
    )
)
ORDER BY 1 desc
```

Di bawah ini adalah panel dasbor dengan tampilan gabungan terpadu ini. Seperti yang Anda lihat, dasbor terlihat hampir identik dengan tampilan yang dihitung dari tabel turunan, kecuali untuk itu akan memiliki up-to-date agregat paling banyak di ujung paling kanan.



Agregat dari komputasi terjadwal yang sering diperbarui

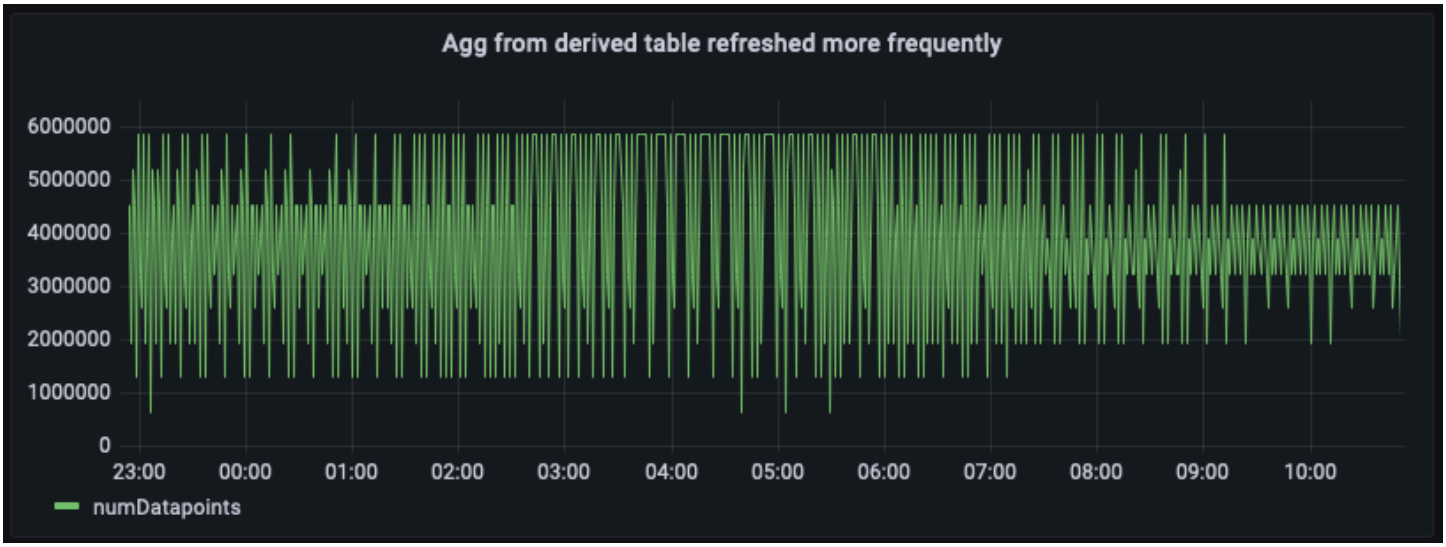
Bergantung pada seberapa sering dasbor Anda dimuat dan seberapa banyak latensi yang Anda inginkan untuk dasbor Anda, ada pendekatan lain untuk mendapatkan hasil yang lebih segar di dasbor Anda: membuat komputasi terjadwal menyegarkan agregat lebih sering. Misalnya, di bawah ini adalah konfigurasi dari perhitungan terjadwal yang sama, kecuali bahwa itu menyegarkan sekali setiap menit (perhatikan jadwal cron ekspres (0/1 * * * *? *)). Dengan pengaturan ini, tabel turunan per_minute_aggs_pt1m akan memiliki agregat yang jauh lebih baru dibandingkan dengan skenario di mana perhitungan menentukan jadwal penyegaran setiap 5 menit sekali.

```
{
    "Name": "MultiPT1mPerMinutePerRegionMeasureCount",
    "QueryString": "SELECT region, bin(time, 1m) as minute, SUM(CASE WHEN measure_name
 = 'metrics' THEN 20 ELSE 5 END) as numDataPoints FROM raw_data.devops WHERE time
 BETWEEN @scheduled_runtime - 10m AND @scheduled_runtime + 1m GROUP BY bin(time, 1m),
 region",
    "ScheduleConfiguration": {
        "ScheduleExpression": "cron(0/1 * * * ? *)"
    },
    "NotificationConfiguration": {
        "SnsConfiguration": {
            "TopicArn": "******"
        }
    },
```

```
    "TargetConfiguration": {
        "TimestreamConfiguration": {
            "DatabaseName": "derived",
            "TableName": "per_minute_aggs_pt1m",
            "TimeColumn": "minute",
            "DimensionMappings": [
                {
                    "Name": "region",
                    "DimensionValueType": "VARCHAR"
                }
            ],
            "MultiMeasureMappings": {
                "TargetMultiMeasureName": "numDataPoints",
                "MultiMeasureAttributeMappings": [
                    {
                        "SourceColumn": "numDataPoints",
                        "MeasureValueType": "BIGINT"
                    }
                ]
            }
        }
    },
    "ErrorReportConfiguration": {
        "S3Configuration" : {
            "BucketName" : "******",
            "ObjectKeyPrefix": "errors",
            "EncryptionOption": "SSE_S3"
        }
    },
    "ScheduledQueryExecutionRoleArn": "******"
}
```

```
SELECT bin(time, 1m) as minute, SUM(numDataPoints) as numDatapoints
FROM "derived"."per_minute_aggs_pt1m"
WHERE time BETWEEN from_milliseconds(1636699996446) AND
 from_milliseconds(1636743196446)
    AND region = 'us-east-1'
GROUP BY bin(time, 1m), region
ORDER BY 1 desc
```

Karena tabel turunan memiliki agregat yang lebih baru, Anda sekarang dapat langsung menanyakan tabel turunan per_minute_aggs_pt1m untuk mendapatkan agregat yang lebih segar, seperti yang dapat dilihat dari kueri sebelumnya dan snapshot dasbor di bawah ini.



Perhatikan bahwa menyegarkan perhitungan terjadwal pada jadwal yang lebih cepat (katakanlah 1 menit dibandingkan dengan 5 menit) akan meningkatkan biaya pemeliharaan untuk perhitungan terjadwal. Pesan notifikasi untuk setiap eksekusi komputasi menyediakan statistik untuk berapa banyak data yang dipindai dan berapa banyak yang ditulis ke tabel turunan. Demikian pula, jika Anda menggunakan tampilan gabungan untuk menyatukan tabel turunan, Anda meminta biaya pada tampilan gabungan dan latensi pemuatan dasbor akan lebih tinggi dibandingkan dengan hanya menanyakan tabel turunan. Oleh karena itu, pendekatan yang Anda pilih akan tergantung pada seberapa sering dasbor Anda diperbarui dan biaya pemeliharaan untuk kueri terjadwal. Jika Anda memiliki puluhan pengguna yang menyegarkan dasbor sekali setiap menit atau lebih, memiliki penyegaran yang lebih sering dari tabel turunan Anda kemungkinan akan menghasilkan biaya yang lebih rendah secara keseluruhan.

Poin terakhir dari setiap perangkat

Aplikasi Anda mungkin mengharuskan Anda membaca pengukuran terakhir yang dipancarkan oleh perangkat. Mungkin ada kasus penggunaan yang lebih umum untuk mendapatkan pengukuran terakhir untuk perangkat sebelum diberikandate/time or the first measurement for a device after a given date/time. Bila Anda memiliki jutaan perangkat dan data bertahun-tahun, pencarian ini mungkin memerlukan pemindaian data dalam jumlah besar.

Di bawah ini Anda akan melihat contoh bagaimana Anda dapat menggunakan kueri terjadwal untuk mengoptimalkan pencarian poin terakhir yang dipancarkan oleh perangkat. Anda dapat

menggunakan pola yang sama untuk mengoptimalkan kueri poin pertama juga jika aplikasi Anda membutuhkannya.

Topik

Dihitung dari tabel sumber

Di bawah ini adalah contoh kueri untuk menemukan pengukuran terakhir yang dipancarkan oleh layanan dalam penerapan tertentu (misalnya, server untuk layanan mikro tertentu dalam wilayah tertentu, sel, silo, dan availability_zone). Dalam aplikasi contoh, kueri ini akan mengembalikan pengukuran terakhir untuk ratusan server. Perhatikan juga bahwa kueri ini memiliki predikat waktu tak terbatas dan mencari data yang lebih lama dari stempel waktu yang diberikan.

> **Note**
>
> Untuk informasi tentang `max` dan `max_by` fungsi, lihatFungsi agregat.

```
SELECT instance_name, MAX(time) AS time, MAX_BY(gc_pause, time) AS last_measure
FROM "raw_data"."devops"
WHERE time < from_milliseconds(1636685271872)
    AND measure_name = 'events'
    AND region = 'us-east-1'
    AND cell = 'us-east-1-cell-10'
    AND silo = 'us-east-1-cell-10-silo-3'
    AND availability_zone = 'us-east-1-1'
    AND microservice_name = 'hercules'
GROUP BY region, cell, silo, availability_zone, microservice_name,
    instance_name, process_name, jdk_version
ORDER BY instance_name, time DESC
```

Tabel turunan untuk dihitung sebelumnya pada perincian harian

Anda dapat mengonversi kasus penggunaan sebelumnya menjadi perhitungan terjadwal. Jika persyaratan aplikasi Anda sedemikian rupa sehingga Anda mungkin perlu mendapatkan nilai

ini untuk seluruh armada Anda di beberapa wilayah, sel, silo, zona ketersediaan, dan layanan mikro, Anda dapat menggunakan satu perhitungan jadwal untuk menghitung nilai untuk seluruh armada Anda. Itulah kekuatan Timestream untuk kueri LiveAnalytics terjadwal tanpa server yang memungkinkan kueri ini disesuaikan dengan persyaratan penskalaan aplikasi Anda.

Di bawah ini adalah kueri untuk menghitung sebelumnya poin terakhir di semua server untuk hari tertentu. Perhatikan bahwa kueri hanya memiliki predikat waktu dan bukan predikat pada dimensi. Predikat waktu membatasi kueri ke hari terakhir dari saat perhitungan dipicu berdasarkan ekspresi jadwal yang ditentukan.

```
SELECT region, cell, silo, availability_zone, microservice_name,
    instance_name, process_name, jdk_version,
    MAX(time) AS time, MAX_BY(gc_pause, time) AS last_measure
FROM raw_data.devops
WHERE time BETWEEN bin(@scheduled_runtime, 1d) - 1d AND bin(@scheduled_runtime, 1d)
    AND measure_name = 'events'
GROUP BY region, cell, silo, availability_zone, microservice_name,
    instance_name, process_name, jdk_version
```

Di bawah ini adalah konfigurasi untuk perhitungan terjadwal menggunakan kueri sebelumnya yang mengeksekusi kueri itu pada pukul 01:00 UTC setiap hari untuk menghitung agregat untuk hari terakhir. Ekspresi jadwal cron (0 1 * *? *) mengontrol perilaku ini dan berjalan satu jam setelah hari berakhir untuk mempertimbangkan data yang tiba hingga satu hari terlambat.

```
{
    "Name": "PT1DPerInstanceLastpoint",
    "QueryString": "SELECT region, cell, silo, availability_zone, microservice_name,
 instance_name, process_name, jdk_version, MAX(time) AS time, MAX_BY(gc_pause, time)
 AS last_measure FROM raw_data.devops WHERE time BETWEEN bin(@scheduled_runtime, 1d) -
 1d AND bin(@scheduled_runtime, 1d) AND measure_name = 'events' GROUP BY region, cell,
 silo, availability_zone, microservice_name, instance_name, process_name, jdk_version",
    "ScheduleConfiguration": {
        "ScheduleExpression": "cron(0 1 * * ? *)"
    },
    "NotificationConfiguration": {
        "SnsConfiguration": {
            "TopicArn": "******"
        }
    },
    "TargetConfiguration": {
        "TimestreamConfiguration": {
```

```
            "DatabaseName": "derived",
            "TableName": "per_timeseries_lastpoint_pt1d",
            "TimeColumn": "time",
            "DimensionMappings": [
                {
                    "Name": "region",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "cell",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "silo",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "availability_zone",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "microservice_name",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "instance_name",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "process_name",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "jdk_version",
                    "DimensionValueType": "VARCHAR"
                }
            ],
            "MultiMeasureMappings": {
                "TargetMultiMeasureName": "last_measure",
                "MultiMeasureAttributeMappings": [
                    {
                        "SourceColumn": "last_measure",
                        "MeasureValueType": "DOUBLE"
                    }
```

```
                ]
            }
        }
    },
    "ErrorReportConfiguration": {
        "S3Configuration" : {
            "BucketName" : "******",
            "ObjectKeyPrefix": "errors",
            "EncryptionOption": "SSE_S3"
        }
    },
    "ScheduledQueryExecutionRoleArn": "******"
}
```

Dihitung dari tabel turunan

Setelah Anda menentukan tabel turunan menggunakan konfigurasi sebelumnya dan setidaknya satu contoh kueri terjadwal telah mewujudkan data ke dalam tabel turunan, Anda sekarang dapat menanyakan tabel turunan untuk mendapatkan pengukuran terbaru. Di bawah ini adalah contoh query pada tabel turunan.

```
SELECT instance_name, MAX(time) AS time, MAX_BY(last_measure, time) AS last_measure
FROM "derived"."per_timeseries_lastpoint_pt1d"
WHERE time < from_milliseconds(1636746715649)
    AND measure_name = 'last_measure'
    AND region = 'us-east-1'
    AND cell = 'us-east-1-cell-10'
    AND silo = 'us-east-1-cell-10-silo-3'
    AND availability_zone = 'us-east-1-1'
    AND microservice_name = 'hercules'
GROUP BY region, cell, silo, availability_zone, microservice_name,
    instance_name, process_name, jdk_version
ORDER BY instance_name, time DESC
```

Menggabungkan dari sumber dan tabel turunan

Mirip dengan contoh sebelumnya, data apa pun dari tabel turunan tidak akan memiliki tulisan terbaru. Oleh karena itu, Anda dapat kembali menggunakan pola yang sama seperti sebelumnya untuk menggabungkan data dari tabel turunan untuk data yang lebih lama dan menggunakan data sumber untuk tip yang tersisa. Di bawah ini adalah contoh kueri semacam itu menggunakan UNION pendekatan serupa. Karena persyaratan aplikasi adalah menemukan pengukuran terbaru

sebelum periode waktu, dan waktu mulai ini bisa di masa lalu, cara Anda menulis kueri ini adalah dengan menggunakan waktu yang disediakan, gunakan data sumber hingga satu hari dari waktu yang ditentukan, dan kemudian gunakan tabel turunan pada data yang lebih lama. Seperti yang Anda lihat dari contoh kueri di bawah ini, predikat waktu pada data sumber dibatasi. Itu memastikan pemrosesan yang efisien pada tabel sumber yang memiliki volume data yang jauh lebih tinggi, dan kemudian predikat waktu tak terbatas ada pada tabel turunan.

```
WITH last_point_derived AS (
    SELECT instance_name, MAX(time) AS time, MAX_BY(last_measure, time) AS last_measure
    FROM "derived"."per_timeseries_lastpoint_pt1d"
    WHERE time < from_milliseconds(1636746715649)
        AND measure_name = 'last_measure'
        AND region = 'us-east-1'
        AND cell = 'us-east-1-cell-10'
        AND silo = 'us-east-1-cell-10-silo-3'
        AND availability_zone = 'us-east-1-1'
        AND microservice_name = 'hercules'
    GROUP BY region, cell, silo, availability_zone, microservice_name,
        instance_name, process_name, jdk_version
), last_point_source AS (
    SELECT instance_name, MAX(time) AS time, MAX_BY(gc_pause, time) AS last_measure
    FROM "raw_data"."devops"
    WHERE time < from_milliseconds(1636746715649) AND time >
 from_milliseconds(1636746715649) - 26h
        AND measure_name = 'events'
        AND region = 'us-east-1'
        AND cell = 'us-east-1-cell-10'
        AND silo = 'us-east-1-cell-10-silo-3'
        AND availability_zone = 'us-east-1-1'
        AND microservice_name = 'hercules'
    GROUP BY region, cell, silo, availability_zone, microservice_name,
        instance_name, process_name, jdk_version
)
SELECT instance_name, MAX(time) AS time, MAX_BY(last_measure, time) AS last_measure
FROM (
    SELECT * FROM last_point_derived
    UNION
    SELECT * FROM last_point_source
)
GROUP BY instance_name
ORDER BY instance_name, time DESC
```

Yang sebelumnya hanyalah salah satu ilustrasi tentang bagaimana Anda dapat menyusun tabel turunan. Jika Anda memiliki data bertahun-tahun, Anda dapat menggunakan lebih banyak tingkat agregasi. Misalnya, Anda dapat memiliki agregat bulanan di atas agregat harian, dan Anda dapat memiliki agregat per jam sebelum harian. Jadi Anda dapat menggabungkan bersama yang terbaru untuk mengisi jam terakhir, per jam untuk mengisi hari terakhir, harian untuk mengisi bulan terakhir, dan bulanan untuk mengisi yang lebih lama. Jumlah level yang Anda atur vs. jadwal penyegaran akan tergantung pada kebutuhan Anda tentang seberapa sering pertanyaan ini menjadi masalah dan berapa banyak pengguna yang secara bersamaan mengeluarkan kueri ini.

Nilai dimensi unik

Anda mungkin memiliki kasus penggunaan di mana Anda memiliki dasbor yang ingin Anda gunakan nilai unik dimensi sebagai variabel untuk menelusuri metrik yang sesuai dengan potongan data tertentu. Cuplikan di bawah ini adalah contoh di mana dasbor mengisi nilai unik dari beberapa dimensi seperti wilayah, sel, silo, layanan mikro, dan availability_zone. Di sini kami menunjukkan contoh bagaimana Anda dapat menggunakan kueri terjadwal untuk secara signifikan mempercepat komputasi nilai-nilai berbeda dari variabel-variabel ini dari metrik yang Anda lacak.

Topik

Pada data mentah

Anda dapat menggunakan SELECT DISTINCT untuk menghitung nilai berbeda yang dilihat dari data Anda. Misalnya, jika Anda ingin mendapatkan nilai wilayah yang berbeda, Anda dapat menggunakan kueri formulir ini.

```
SELECT DISTINCT region
FROM "raw_data"."devops"
WHERE time > ago(1h)
ORDER BY 1
```

Anda mungkin melacak jutaan perangkat dan miliaran deret waktu. Namun, dalam banyak kasus, variabel menarik ini untuk dimensi kardinalitas yang lebih rendah, di mana Anda memiliki beberapa hingga puluhan nilai. Komputasi DISTINCT dari data mentah dapat memerlukan pemindaian volume data yang besar.

Pra-komputasi nilai dimensi unik

Anda ingin variabel-variabel ini dimuat dengan cepat sehingga dasbor Anda interaktif. Selain itu, variabel-variabel ini sering dihitung pada setiap beban dasbor, jadi Anda ingin mereka menjadi hemat biaya juga. Anda dapat mengoptimalkan menemukan variabel-variabel ini menggunakan kueri terjadwal dan mewujudkannya dalam tabel turunan.

Pertama, Anda perlu mengidentifikasi dimensi yang Anda butuhkan untuk menghitung DISTINCT nilai atau kolom yang akan Anda gunakan dalam predikat saat menghitung nilaiDISTINCT.

Dalam contoh ini, Anda dapat melihat bahwa dasbor mengisi nilai yang berbeda untuk wilayah dimensi, sel, silo, availability_zone, dan layanan mikro. Jadi, Anda dapat menggunakan kueri di bawah ini untuk menghitung nilai unik ini terlebih dahulu.

```
SELECT region, cell, silo, availability_zone, microservice_name,
    min(@scheduled_runtime) AS time, COUNT(*) as numDataPoints
FROM raw_data.devops
WHERE time BETWEEN @scheduled_runtime - 15m AND @scheduled_runtime
GROUP BY region, cell, silo, availability_zone, microservice_name
```

Ada beberapa hal penting yang perlu diperhatikan di sini.

- Anda dapat menggunakan satu komputasi terjadwal untuk pra-menghitung nilai untuk banyak kueri yang berbeda. Misalnya, Anda menggunakan query sebelumnya untuk pra-menghitung nilai untuk lima variabel yang berbeda. Jadi Anda tidak perlu satu untuk setiap variabel. Anda dapat menggunakan pola yang sama ini untuk mengidentifikasi komputasi bersama di beberapa panel untuk mengoptimalkan jumlah kueri terjadwal yang perlu Anda pertahankan.
- Nilai unik dari dimensi tidak secara inheren data deret waktu. Jadi Anda mengonversi ini ke deret waktu menggunakan @scheduled_runtime. Dengan mengaitkan data ini dengan parameter @scheduled_runtime, Anda juga dapat melacak nilai unik mana yang muncul pada titik waktu tertentu, sehingga membuat data deret waktu darinya.
- Pada contoh sebelumnya, Anda akan melihat nilai metrik yang dilacak. Contoh ini menggunakan COUNT (*). Anda dapat menghitung agregat bermakna lainnya jika Anda ingin melacaknya untuk dasbor Anda.

Di bawah ini adalah konfigurasi untuk perhitungan terjadwal menggunakan kueri sebelumnya. Dalam contoh ini, dikonfigurasi untuk menyegarkan setiap 15 menit sekali menggunakan cron ekspresi jadwal (0/15 * * *? *).

```
{
    "Name": "PT15mHighCardPerUniqueDimensions",
    "QueryString": "SELECT region, cell, silo, availability_zone, microservice_name,
 min(@scheduled_runtime) AS time, COUNT(*) as numDataPoints FROM raw_data.devops WHERE
 time BETWEEN @scheduled_runtime - 15m AND @scheduled_runtime GROUP BY region, cell,
 silo, availability_zone, microservice_name",
    "ScheduleConfiguration": {
        "ScheduleExpression": "cron(0/15 * * * ? *)"
    },
    "NotificationConfiguration": {
        "SnsConfiguration": {
            "TopicArn": "******"
        }
    },
    "TargetConfiguration": {
        "TimestreamConfiguration": {
            "DatabaseName": "derived",
            "TableName": "hc_unique_dimensions_pt15m",
            "TimeColumn": "time",
            "DimensionMappings": [
                {
                    "Name": "region",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "cell",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "silo",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "availability_zone",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "microservice_name",
                    "DimensionValueType": "VARCHAR"
                }
            ],
            "MultiMeasureMappings": {
                "TargetMultiMeasureName": "count_multi",
```

```
                "MultiMeasureAttributeMappings": [
                    {
                        "SourceColumn": "numDataPoints",
                        "MeasureValueType": "BIGINT"
                    }
                ]
            }
        }
    },
    "ErrorReportConfiguration": {
        "S3Configuration" : {
            "BucketName" : "******",
            "ObjectKeyPrefix": "errors",
            "EncryptionOption": "SSE_S3"
        }
    },
    "ScheduledQueryExecutionRoleArn": "******"
}
```

Menghitung variabel dari tabel turunan

Setelah perhitungan terjadwal pra-terwujud nilai unik dalam tabel turunan hc_unique_dimensions_pt15m, Anda dapat menggunakan tabel turunan untuk menghitung nilai unik dimensi secara efisien. Di bawah ini adalah contoh kueri tentang cara menghitung nilai unik, dan bagaimana Anda dapat menggunakan variabel lain sebagai predikat dalam kueri nilai unik ini.

Wilayah

```
SELECT DISTINCT region
FROM "derived"."hc_unique_dimensions_pt15m"
WHERE time > ago(1h)
ORDER BY 1
```

Sel

```
SELECT DISTINCT cell
FROM "derived"."hc_unique_dimensions_pt15m"
WHERE time > ago(1h)
  AND region = '${region}'
ORDER BY 1
```

Silo

```
SELECT DISTINCT silo
FROM "derived"."hc_unique_dimensions_pt15m"
WHERE time > ago(1h)
   AND region = '${region}' AND cell = '${cell}'
ORDER BY 1
```

Layanan Mikro

```
SELECT DISTINCT microservice_name
FROM "derived"."hc_unique_dimensions_pt15m"
WHERE time > ago(1h)
   AND region = '${region}' AND cell = '${cell}'
ORDER BY 1
```

Zona Ketersediaan

```
SELECT DISTINCT availability_zone
FROM "derived"."hc_unique_dimensions_pt15m"
WHERE time > ago(1h)
   AND region = '${region}' AND cell = '${cell}' AND silo = '${silo}'
ORDER BY 1
```

Menangani data yang datang terlambat

Anda mungkin memiliki skenario di mana Anda dapat memiliki data yang datang terlambat secara signifikan, misalnya, waktu ketika data dicerna ke Timestream untuk secara signifikan LiveAnalytics tertunda dibandingkan dengan stempel waktu yang terkait dengan baris yang dicerna. Dalam contoh sebelumnya, Anda telah melihat bagaimana Anda dapat menggunakan rentang waktu yang ditentukan oleh parameter @scheduled_runtime untuk memperhitungkan beberapa data yang terlambat tiba. Namun, jika Anda memiliki kasus penggunaan di mana data dapat ditunda oleh jam atau hari, Anda mungkin memerlukan pola yang berbeda untuk memastikan pra-perhitungan Anda dalam tabel turunan diperbarui dengan tepat untuk mencerminkan data yang datang terlambat tersebut. Untuk informasi umum tentang data yang datang terlambat, lihat. Menulis data (sisipan dan upserts)

Berikut ini Anda akan melihat dua cara berbeda untuk mengatasi data yang datang terlambat ini.

- Jika Anda memiliki penundaan yang dapat diprediksi dalam kedatangan data Anda, maka Anda dapat menggunakan perhitungan terjadwal “catch-up” lain untuk memperbarui agregat Anda untuk data yang terlambat tiba.

- Jika Anda memiliki penundaan yang tidak dapat diprediksi atau data kedatangan terlambat sesekali, Anda dapat menggunakan eksekusi manual untuk memperbarui tabel turunan.

Diskusi ini mencakup skenario untuk kedatangan data yang terlambat. Namun, prinsip yang sama berlaku untuk koreksi data, di mana Anda telah memodifikasi data dalam tabel sumber Anda dan Anda ingin memperbarui agregat dalam tabel turunan Anda.

Topik

Pertanyaan mengejar ketinggalan terjadwal

Kueri mengumpulkan data yang tiba tepat waktu

Di bawah ini adalah pola yang akan Anda lihat bagaimana Anda dapat menggunakan cara otomatis untuk memperbarui agregat Anda jika Anda memiliki penundaan yang dapat diprediksi dalam kedatangan data Anda. Pertimbangkan salah satu contoh sebelumnya dari perhitungan terjadwal pada data real-time di bawah ini. Perhitungan terjadwal ini menyegarkan tabel turunan setiap 30 menit sekali dan sudah memperhitungkan data hingga satu jam tertunda.

```
{
    "Name": "MultiPT30mPerHrPerTimeseriesDPCount",
    "QueryString": "SELECT region, cell, silo, availability_zone, microservice_name,
 instance_type, os_version, instance_name, process_name, jdk_version, bin(time,
 1h) as hour, SUM(CASE WHEN measure_name = 'metrics' THEN 20 ELSE 5 END) as
 numDataPoints FROM raw_data.devops WHERE time BETWEEN bin(@scheduled_runtime, 1h)
 - 1h AND @scheduled_runtime + 1h GROUP BY region, cell, silo, availability_zone,
 microservice_name, instance_type, os_version, instance_name, process_name,
 jdk_version, bin(time, 1h)",
    "ScheduleConfiguration": {
        "ScheduleExpression": "cron(0/30 * * * ? *)"
    },
    "NotificationConfiguration": {
        "SnsConfiguration": {
            "TopicArn": "******"
        }
    },
    "TargetConfiguration": {
        "TimestreamConfiguration": {
```

```
            "DatabaseName": "derived",
            "TableName": "dp_per_timeseries_per_hr",
            "TimeColumn": "hour",
            "DimensionMappings": [
                {
                    "Name": "region",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "cell",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "silo",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "availability_zone",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "microservice_name",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "instance_type",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "os_version",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "instance_name",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "process_name",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "jdk_version",
                    "DimensionValueType": "VARCHAR"
                }
```

```
            ],
            "MultiMeasureMappings": {
                "TargetMultiMeasureName": "numDataPoints",
                "MultiMeasureAttributeMappings": [
                    {
                        "SourceColumn": "numDataPoints",
                        "MeasureValueType": "BIGINT"
                    }
                ]
            }
        }
    },
    "ErrorReportConfiguration": {
        "S3Configuration" : {
            "BucketName" : "******",
            "ObjectKeyPrefix": "errors",
            "EncryptionOption": "SSE_S3"
        }
    },
    "ScheduledQueryExecutionRoleArn": "******"
}
```

Kueri catch-up memperbarui agregat untuk data yang datang terlambat

Sekarang jika Anda mempertimbangkan kasus bahwa data Anda dapat ditunda sekitar 12 jam. Di bawah ini adalah varian dari kueri yang sama. Namun, perbedaannya adalah bahwa ia menghitung agregat pada data yang tertunda hingga 12 jam dibandingkan dengan saat perhitungan terjadwal sedang dipicu. Misalnya, Anda melihat kueri pada contoh di bawah ini, rentang waktu yang ditargetkan kueri ini adalah antara 2 jam hingga 14 jam sebelum ketika kueri dipicu. Selain itu, jika Anda melihat ekspresi jadwal cron (0 0,12 * *? *), itu akan memicu perhitungan pada pukul 00:00 UTC dan UTC 12:00 setiap hari. Oleh karena itu, ketika kueri dipicu pada 2021-12-01 00:00:00, maka kueri memperbarui agregat dalam rentang waktu 2021-11-30 10:00:00 hingga 2021-11-30 22:00:00. Kueri terjadwal menggunakan semantik upsert yang mirip dengan Timestream untuk LiveAnalytics penulisan di mana kueri penangkapan ini akan memperbarui nilai agregat dengan nilai yang lebih baru jika ada data yang datang terlambat di jendela atau jika agregat yang lebih baru ditemukan (misalnya, pengelompokan baru muncul di agregat ini yang tidak ada saat perhitungan terjadwal asli dipicu), maka agregat baru akan dimasukkan ke dalam tabel turunan. Demikian pula, ketika instance berikutnya dipicu pada 2021-12-01 12:00:00, maka instance itu akan memperbarui agregat dalam kisaran 2021-11-30 22:00:00 hingga 2021-12-01 10:00:00.

```
      {
   "Name": "MultiPT12HPerHrPerTimeseriesDPCountCatchUp",
   "QueryString": "SELECT region, cell, silo, availability_zone, microservice_name,
instance_type, os_version, instance_name, process_name, jdk_version, bin(time, 1h)
as hour, SUM(CASE WHEN measure_name = 'metrics' THEN 20 ELSE 5 END) as numDataPoints
FROM raw_data.devops WHERE time BETWEEN bin(@scheduled_runtime, 1h) - 14h AND
bin(@scheduled_runtime, 1h) - 2h GROUP BY region, cell, silo, availability_zone,
microservice_name, instance_type, os_version, instance_name, process_name,
jdk_version, bin(time, 1h)",
   "ScheduleConfiguration": {
       "ScheduleExpression": "cron(0 0,12 * * ? *)"
   },
   "NotificationConfiguration": {
       "SnsConfiguration": {
           "TopicArn": "******"
       }
   },
   "TargetConfiguration": {
       "TimestreamConfiguration": {
           "DatabaseName": "derived",
           "TableName": "dp_per_timeseries_per_hr",
           "TimeColumn": "hour",
           "DimensionMappings": [
               {
                   "Name": "region",
                   "DimensionValueType": "VARCHAR"
               },
               {
                   "Name": "cell",
                   "DimensionValueType": "VARCHAR"
               },
               {
                   "Name": "silo",
                   "DimensionValueType": "VARCHAR"
               },
               {
                   "Name": "availability_zone",
                   "DimensionValueType": "VARCHAR"
               },
               {
                   "Name": "microservice_name",
                   "DimensionValueType": "VARCHAR"
               },
               {
```

```
                    "Name": "instance_type",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "os_version",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "instance_name",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "process_name",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "jdk_version",
                    "DimensionValueType": "VARCHAR"
                }
            ],
            "MultiMeasureMappings": {
                "TargetMultiMeasureName": "numDataPoints",
                "MultiMeasureAttributeMappings": [
                    {
                        "SourceColumn": "numDataPoints",
                        "MeasureValueType": "BIGINT"
                    }
                ]
            }
        }
    },
    "ErrorReportConfiguration": {
        "S3Configuration" : {
            "BucketName" : "******",
            "ObjectKeyPrefix": "errors",
            "EncryptionOption": "SSE_S3"
        }
    },
    "ScheduledQueryExecutionRoleArn": "******"
}
```

Contoh sebelumnya ini adalah ilustrasi dengan asumsi kedatangan terlambat Anda dibatasi hingga 12 jam dan tidak apa-apa untuk memperbarui tabel turunan setiap 12 jam sekali untuk data yang

tiba lebih lambat dari jendela waktu nyata. Anda dapat menyesuaikan pola ini untuk memperbarui tabel turunan Anda sekali setiap jam sehingga tabel turunan Anda mencerminkan data yang datang terlambat lebih cepat. Demikian pula, Anda dapat menyesuaikan rentang waktu menjadi lebih tua dari 12 jam, misalnya, sehari atau bahkan seminggu atau lebih, untuk menangani data kedatangan terlambat yang dapat diprediksi.

Eksekusi manual untuk data kedatangan terlambat yang tidak dapat diprediksi

Mungkin ada contoh di mana Anda memiliki data kedatangan terlambat yang tidak dapat diprediksi atau Anda membuat perubahan pada data sumber dan memperbarui beberapa nilai setelah fakta. Dalam semua kasus seperti itu, Anda dapat memicu kueri terjadwal secara manual untuk memperbarui tabel turunan. Di bawah ini adalah contoh bagaimana Anda dapat mencapai ini.

Asumsikan bahwa Anda memiliki kasus penggunaan di mana Anda memiliki perhitungan yang ditulis ke tabel turunan dp_per_timeseries_per_hr. Data dasar Anda dalam tabel devops diperbarui pada rentang waktu 2021-11-30 23:00:00 - 2021-12-01 00:00:00. Ada dua kueri terjadwal berbeda yang dapat digunakan untuk memperbarui tabel turunan ini: Multi PT3 0 mPerHr PerTimeseries DPCount dan MultiPT12HPerHrPerTimeseriesDPCountCatchUp. Setiap perhitungan terjadwal yang Anda buat di Timestream LiveAnalytics memiliki keunikan ARN yang Anda peroleh saat membuat perhitungan atau saat Anda melakukan operasi daftar. Anda dapat menggunakan ARN untuk perhitungan dan nilai untuk parameter @scheduled_runtime yang diambil oleh kueri untuk melakukan operasi ini.

Asumsikan bahwa perhitungan untuk Multi PT3 0 mPerHr PerTimeseries DPCount memiliki ARN arn_1 dan Anda ingin menggunakan perhitungan ini untuk memperbarui tabel turunan. Karena komputasi terjadwal sebelumnya memperbarui agregat 1 jam sebelum dan 1 jam setelah nilai @scheduled_runtime, Anda dapat mencakup rentang waktu untuk pembaruan (2021-11-30 23:00:00 - 2021-12-01 00:00:00) menggunakan nilai 2021-12-01 00:00:00 untuk parameter @scheduled_runtime. Anda dapat menggunakan ExecuteScheduledQuery API untuk meneruskan perhitungan ini dan nilai parameter waktu dalam epoch seconds (inUTC) untuk mencapai ini. ARN Di bawah ini adalah contoh menggunakan AWS CLI dan Anda dapat mengikuti pola yang sama menggunakan salah satu yang SDKs didukung oleh Timestream untuk LiveAnalytics.

```
aws timestream-query execute-scheduled-query --scheduled-query-arn arn_1 --invocation-
time 1638316800 --profile profile --region us-east-1
```

Pada contoh sebelumnya, profil adalah AWS profil yang memiliki hak istimewa yang sesuai untuk melakukan API panggilan ini dan 1638316800 sesuai dengan epoch second untuk 2021-12-01 00:00:00. Pemicu manual ini berperilaku hampir seperti pemicu otomatis dengan asumsi sistem memicu pemanggilan ini pada periode waktu yang diinginkan.

Jika Anda memiliki pembaruan dalam periode waktu yang lebih lama, katakanlah data dasar telah diperbarui untuk 2021-11-30 23:00:00 - 2021-12-01 11:00:00, maka Anda dapat memicu kueri sebelumnya beberapa kali untuk mencakup seluruh rentang waktu ini. Misalnya, Anda dapat melakukan enam eksekusi yang berbeda sebagai berikut.

```
aws timestream-query execute-scheduled-query --scheduled-query-arn arn_1 --invocation-
time 1638316800 --profile profile --region us-east-1

aws timestream-query execute-scheduled-query --scheduled-query-arn arn_1 --invocation-
time 1638324000 --profile profile --region us-east-1

aws timestream-query execute-scheduled-query --scheduled-query-arn arn_1 --invocation-
time 1638331200 --profile profile --region us-east-1

aws timestream-query execute-scheduled-query --scheduled-query-arn arn_1 --invocation-
time 1638338400 --profile profile --region us-east-1

aws timestream-query execute-scheduled-query --scheduled-query-arn arn_1 --invocation-
time 1638345600 --profile profile --region us-east-1

aws timestream-query execute-scheduled-query --scheduled-query-arn arn_1 --invocation-
time 1638352800 --profile profile --region us-east-1
```

Enam perintah sebelumnya sesuai dengan perhitungan terjadwal yang dipanggil pada 2021-12-01 00:00:00, 2021-12-01 02:00:00, 2021-12-01 04:0:00, 2021-12-01 06:00:00, 2021-12-01 08:00:00, dan 2021-12-01 10:00:

Atau, Anda dapat menggunakan komputasi Multi PT12HPerHrPerTimeseriesDPCountCatchUp dipicu pada 2021-12-01 13:00:00 untuk satu eksekusi guna memperbarui agregat untuk seluruh rentang waktu 12 jam. Misalnya, jika arn_2 adalah ARN untuk perhitungan itu, Anda dapat menjalankan perintah berikut dari. CLI

```
aws timestream-query execute-scheduled-query --scheduled-query-arn arn_2 --invocation-
time 1638363600 --profile profile --region us-east-1
```

Perlu dicatat bahwa untuk pemicu manual, Anda dapat menggunakan stempel waktu untuk parameter waktu pemanggilan yang tidak perlu disejajarkan dengan stempel waktu pemicu otomatis tersebut. Misalnya, pada contoh sebelumnya, Anda memicu perhitungan pada waktu 2021-12-01 13:00:00 meskipun jadwal otomatis hanya terpicu pada stempel waktu 2021-12-01 10:00:00, 2021-12-01 12:00:00, dan 2021-12-02 00:00:00. Timestream untuk LiveAnalytics memberi Anda

fleksibilitas untuk memicunya dengan nilai yang sesuai sesuai kebutuhan untuk operasi manual Anda.

Berikut ini adalah beberapa pertimbangan penting saat menggunakan. ExecuteScheduledQuery API

- Jika Anda memicu beberapa pemanggilan ini, Anda perlu memastikan bahwa pemanggilan ini tidak menghasilkan hasil dalam rentang waktu yang tumpang tindih. Misalnya, dalam contoh sebelumnya, ada enam doa. Setiap pemanggilan mencakup rentang waktu 2 jam, dan karenanya stempel waktu pemanggilan disebarkan masing-masing dua jam untuk menghindari tumpang tindih dalam pembaruan. Ini memastikan bahwa data dalam tabel turunan berakhir dalam keadaan yang cocok adalah agregat dari tabel sumber. Jika Anda tidak dapat memastikan rentang waktu yang tidak tumpang tindih, maka pastikan eksekusi ini dipicu secara berurutan satu demi satu. Jika Anda memicu beberapa eksekusi secara bersamaan yang tumpang tindih dalam rentang waktunya, maka Anda dapat melihat kegagalan pemicu di mana Anda mungkin melihat konflik versi dalam laporan kesalahan untuk eksekusi ini. Hasil yang dihasilkan oleh pemanggilan kueri terjadwal diberi versi berdasarkan kapan pemanggilan dipicu. Oleh karena itu, baris yang dihasilkan oleh pemanggilan yang lebih baru memiliki versi yang lebih tinggi. Catatan versi yang lebih tinggi dapat menimpa catatan versi yang lebih rendah. Untuk kueri terjadwal yang dipicu secara otomatis, Timestream untuk LiveAnalytics secara otomatis mengelola jadwal sehingga Anda tidak melihat masalah ini bahkan jika pemanggilan berikutnya memiliki rentang waktu yang tumpang tindih.
- disebutkan sebelumnya, Anda dapat memicu pemanggilan dengan nilai stempel waktu apa pun untuk @scheduled_runtime. Jadi, Anda bertanggung jawab untuk mengatur nilai dengan tepat sehingga rentang waktu yang sesuai diperbarui dalam tabel turunan yang sesuai dengan rentang di mana data diperbarui di tabel sumber.
- Anda juga dapat menggunakan pemicu manual ini untuk kueri terjadwal yang berada dalam DISABLED status. Ini memungkinkan Anda untuk menentukan kueri khusus yang tidak dieksekusi dalam jadwal otomatis, karena mereka berada dalam DISABLED keadaan. Sebaliknya, Anda dapat menggunakan pemicu manual pada mereka untuk mengelola koreksi data atau kasus penggunaan keterlambatan kedatangan.

Mengisi kembali pra-perhitungan historis

Saat Anda membuat perhitungan terjadwal, Timestream untuk LiveAnalytics mengelola eksekusi kueri yang bergerak maju di mana penyegaran diatur oleh ekspresi jadwal yang Anda berikan. Bergantung pada seberapa banyak data historis tabel sumber Anda, Anda mungkin ingin memperbarui tabel turunan Anda dengan agregat yang sesuai dengan data historis. Anda dapat menggunakan logika sebelumnya untuk pemicu manual untuk mengisi kembali agregat historis.

Misalnya, jika kita mempertimbangkan tabel turunan per_timeseries_lastpoint_pt1d, maka perhitungan terjadwal diperbarui sekali sehari selama satu hari terakhir. Jika tabel sumber Anda memiliki satu tahun data, Anda dapat menggunakan ARN untuk perhitungan terjadwal ini dan memicunya secara manual untuk setiap hari hingga satu tahun sehingga tabel turunan memiliki semua kueri historis yang terisi. Perhatikan bahwa semua peringatan untuk pemicu manual berlaku di sini. Selain itu, jika tabel turunan diatur sedemikian rupa sehingga konsumsi historis akan menulis ke penyimpanan magnetik pada tabel turunan, waspadai praktik dan batasan terbaik untuk menulis ke penyimpanan magnetik.

Contoh kueri terjadwal

Bagian ini berisi contoh bagaimana Anda dapat menggunakan Timestream untuk Kueri LiveAnalytics Terjadwal untuk mengoptimalkan biaya dan waktu muat dasbor saat memvisualisasikan statistik seluruh armada secara efektif memantau armada perangkat Anda. Kueri Terjadwal di Timestream untuk LiveAnalytics memungkinkan Anda mengekspresikan kueri Anda menggunakan luas SQL permukaan penuh Timestream untuk. LiveAnalytics Kueri Anda dapat menyertakan satu atau beberapa tabel sumber, melakukan agregasi atau kueri lain yang diizinkan oleh Timestream untuk LiveAnalytics SQL bahasa, dan kemudian menyimpan hasil kueri di tabel tujuan lain di Timestream for. LiveAnalytics

Bagian ini mengacu pada tabel target dari kueri terjadwal sebagai tabel turunan.

Sebagai contoh, kami akan menggunakan DevOps aplikasi tempat Anda memantau armada besar server yang digunakan di beberapa penerapan (seperti wilayah, sel, dan silo), beberapa layanan mikro, dan Anda melacak statistik seluruh armada menggunakan Timestream untuk. LiveAnalytics Contoh skema yang akan kita gunakan dijelaskan dalam Skema Sampel Kueri Terjadwal.

Skenario berikut akan dijelaskan.

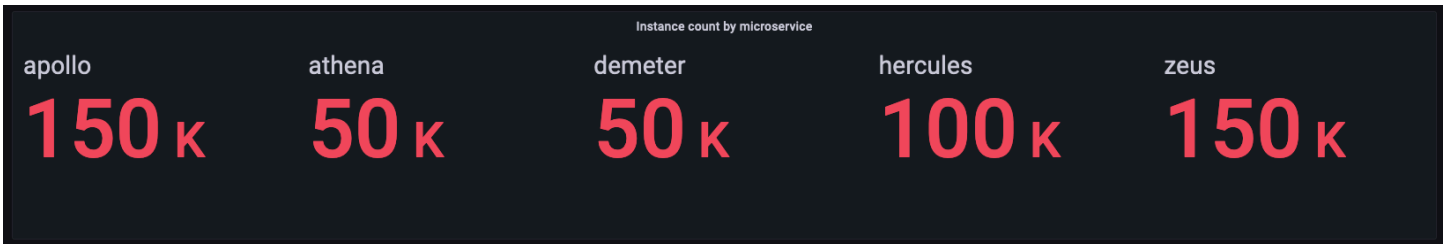
- Cara mengonversi dasbor, memplot statistik agregat dari data mentah yang Anda konsumsi ke Timestream menjadi kueri terjadwal dan kemudian cara menggunakan agregat yang telah dihitung sebelumnya untuk membuat dasbor baru yang menampilkan statistik agregat. LiveAnalytics
- Cara menggabungkan kueri terjadwal untuk mendapatkan tampilan agregat dan data granular mentah, untuk menelusuri detail. Ini memungkinkan Anda menyimpan dan menganalisis data mentah sambil mengoptimalkan operasi umum di seluruh armada menggunakan kueri terjadwal.
- Cara mengoptimalkan biaya menggunakan kueri terjadwal dengan menemukan agregat mana yang digunakan di beberapa dasbor dan memiliki kueri terjadwal yang sama mengisi beberapa panel di dasbor yang sama atau beberapa.

Topik

Mengonversi dasbor agregat menjadi kueri terjadwal

Asumsikan Anda menghitung statistik armada seperti jumlah host dalam armada oleh lima layanan mikro dan oleh enam wilayah tempat layanan Anda digunakan. Dari snapshot di bawah ini, Anda dapat melihat ada 500K server yang memancarkan metrik, dan beberapa wilayah yang lebih besar (misalnya, us-east-1) memiliki> 200K server.

Menghitung agregat ini, di mana Anda menghitung nama instance yang berbeda lebih dari ratusan gigabyte data dapat menghasilkan latensi kueri puluhan detik, selain biaya pemindaian data.



Kueri dasbor asli

Agregat yang ditampilkan di panel dasboard dihitung, dari data mentah, menggunakan kueri di bawah ini. Kueri menggunakan beberapa SQL konstruksi, seperti jumlah yang berbeda dan beberapa fungsi agregasi.

```
SELECT CASE WHEN microservice_name = 'apollo' THEN num_instances ELSE NULL END AS
 apollo,
    CASE WHEN microservice_name = 'athena' THEN num_instances ELSE NULL END AS athena,
    CASE WHEN microservice_name = 'demeter' THEN num_instances ELSE NULL END AS
 demeter,
    CASE WHEN microservice_name = 'hercules' THEN num_instances ELSE NULL END AS
 hercules,
    CASE WHEN microservice_name = 'zeus' THEN num_instances ELSE NULL END AS zeus
FROM (
    SELECT microservice_name, SUM(num_instances) AS num_instances
    FROM (
        SELECT microservice_name, COUNT(DISTINCT instance_name) as num_instances
```

```
        FROM "raw_data"."devops"
        WHERE time BETWEEN from_milliseconds(1636526171043) AND
 from_milliseconds(1636612571043)
            AND measure_name = 'metrics'
        GROUP BY region, cell, silo, availability_zone, microservice_name
    )
    GROUP BY microservice_name
)
```

Mengonversi ke kueri terjadwal

Query sebelumnya dapat dikonversi menjadi query terjadwal sebagai berikut. Pertama-tama Anda menghitung nama host yang berbeda dalam penerapan tertentu di wilayah, sel, silo, zona ketersediaan, dan layanan mikro. Kemudian Anda menambahkan host untuk menghitung per jam per jumlah host microservice. Dengan menggunakan `@scheduled_runtime` parameter yang didukung oleh kueri terjadwal, Anda dapat menghitung ulang selama satu jam terakhir saat kueri dipanggil. `WHERE`Klausa `bin(@scheduled_runtime, 1h)` dalam kueri dalam memastikan bahwa meskipun kueri dijadwalkan pada suatu waktu di tengah jam, Anda masih mendapatkan data selama satu jam penuh.

Meskipun kueri menghitung agregat per jam, seperti yang akan Anda lihat dalam konfigurasi komputasi terjadwal, kueri diatur untuk menyegarkan setiap setengah jam sehingga Anda mendapatkan pembaruan di tabel turunan lebih cepat. Anda dapat menyetelnya berdasarkan persyaratan kesegaran Anda, misalnya, menghitung ulang agregat setiap 15 menit atau menghitung ulang pada batas jam.

```
SELECT microservice_name, hour, SUM(num_instances) AS num_instances
FROM (
        SELECT microservice_name, bin(time, 1h) AS hour,
            COUNT(DISTINCT instance_name) as num_instances
       FROM raw_data.devops
       WHERE time BETWEEN bin(@scheduled_runtime, 1h) - 1h AND @scheduled_runtime

           AND measure_name = 'metrics'
       GROUP BY region, cell, silo, availability_zone, microservice_name, bin(time, 1h)

     )
GROUP BY microservice_name, hour
```

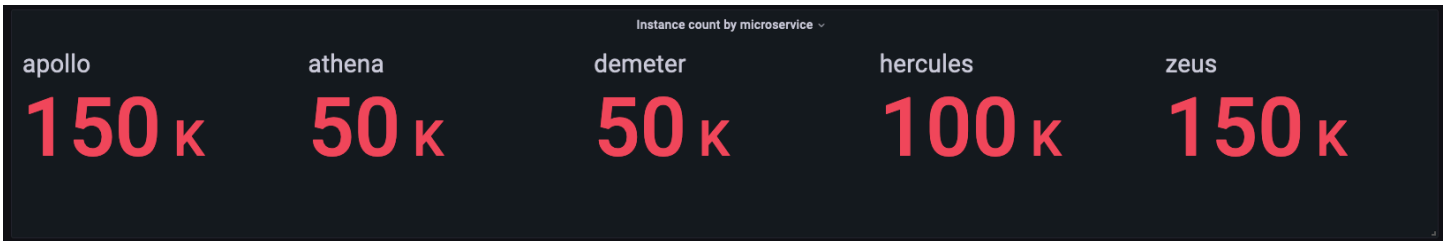
```
{
    "Name": "MultiPT30mHostCountMicroservicePerHr",
```

```
    "QueryString": "SELECT microservice_name, hour, SUM(num_instances) AS num_instances
    FROM (        SELECT microservice_name, bin(time, 1h) AS hour, COUNT(DISTINCT
instance_name) as num_instances        FROM raw_data.devops        WHERE time BETWEEN
bin(@scheduled_runtime, 1h) - 1h AND @scheduled_runtime            AND measure_name
= 'metrics'        GROUP BY region, cell, silo, availability_zone, microservice_name,
bin(time, 1h)    )    GROUP BY microservice_name, hour",
    "ScheduleConfiguration": {
        "ScheduleExpression": "cron(0/30 * * * ? *)"
    },
    "NotificationConfiguration": {
        "SnsConfiguration": {
            "TopicArn": "******"
        }
    },
    "TargetConfiguration": {
        "TimestreamConfiguration": {
            "DatabaseName": "derived",
            "TableName": "host_count_pt1h",
            "TimeColumn": "hour",
            "DimensionMappings": [
                {
                    "Name": "microservice_name",
                    "DimensionValueType": "VARCHAR"
                }
            ],
            "MultiMeasureMappings": {
                "TargetMultiMeasureName": "num_instances",
                "MultiMeasureAttributeMappings": [
                    {
                        "SourceColumn": "num_instances",
                        "MeasureValueType": "BIGINT"
                    }
                ]
            }
        }
    },
    "ErrorReportConfiguration": {
        "S3Configuration" : {
            "BucketName" : "******",
            "ObjectKeyPrefix": "errors",
            "EncryptionOption": "SSE_S3"
        }
    },
    "ScheduledQueryExecutionRoleArn": "******"
```

```
}
```

Menggunakan hasil yang telah dihitung sebelumnya di dasbor baru

Anda sekarang akan melihat cara membuat dasbor tampilan agregat Anda menggunakan tabel turunan dari kueri terjadwal yang Anda buat. Dari snapshot dasbor, Anda juga akan dapat memvalidasi bahwa agregat yang dihitung dari tabel turunan dan tabel dasar juga cocok. Setelah Anda membuat dasbor menggunakan tabel turunan, Anda akan melihat waktu muat yang jauh lebih cepat dan biaya yang lebih rendah untuk menggunakan tabel turunan dibandingkan dengan menghitung agregat ini dari data mentah. Di bawah ini adalah snapshot dasbor menggunakan data yang telah dihitung sebelumnya, dan kueri yang digunakan untuk merender panel ini menggunakan data yang telah dihitung sebelumnya yang disimpan dalam tabel “turunan”. host_count_pt1h”. Perhatikan bahwa struktur kueri sangat mirip dengan kueri yang digunakan di dasbor pada data mentah, kecuali itu menggunakan tabel turunan yang sudah menghitung jumlah berbeda yang digabungkan oleh kueri ini.




```
SELECT CASE WHEN microservice_name = 'apollo' THEN num_instances ELSE NULL END AS
 apollo,
    CASE WHEN microservice_name = 'athena' THEN num_instances ELSE NULL END AS athena,
    CASE WHEN microservice_name = 'demeter' THEN num_instances ELSE NULL END AS
 demeter,
    CASE WHEN microservice_name = 'hercules' THEN num_instances ELSE NULL END AS
 hercules,
    CASE WHEN microservice_name = 'zeus' THEN num_instances ELSE NULL END AS zeus
FROM (
    SELECT microservice_name, AVG(num_instances) AS num_instances
    FROM (
        SELECT microservice_name, bin(time, 1h), SUM(num_instances) as num_instances
        FROM "derived"."host_count_pt1h"
        WHERE time BETWEEN from_milliseconds(1636567785421) AND
 from_milliseconds(1636654185421)
            AND measure_name = 'num_instances'
        GROUP BY microservice_name, bin(time, 1h)
    )
    GROUP BY microservice_name
```

```
)
```

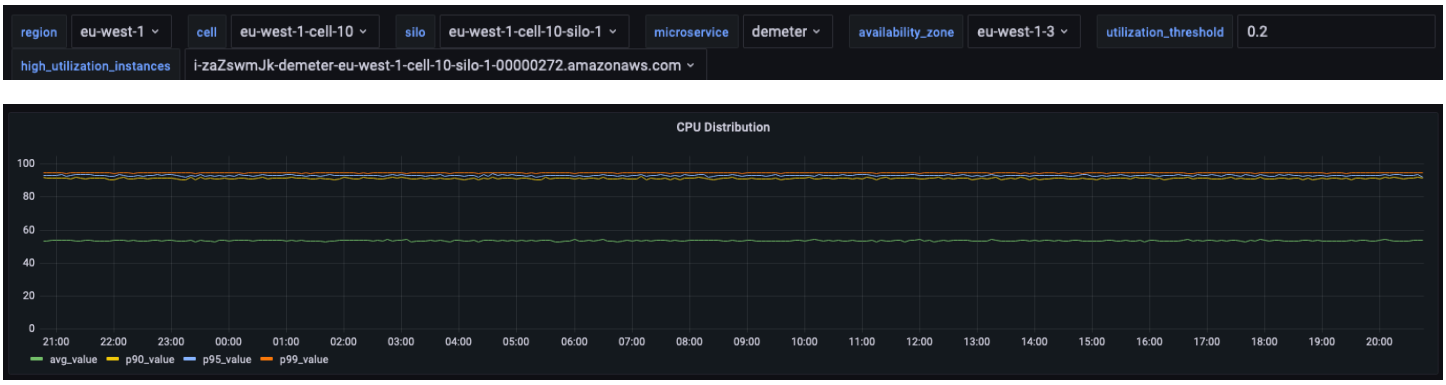
Menggunakan kueri terjadwal dan data mentah untuk menelusuri

Anda dapat menggunakan statistik agregat di seluruh armada Anda untuk mengidentifikasi area yang perlu ditelusuri dan kemudian menggunakan data mentah untuk menelusuri data terperinci untuk mendapatkan wawasan yang lebih dalam.

Dalam contoh ini, Anda akan melihat bagaimana Anda dapat menggunakan dasbor agregat untuk mengidentifikasi penerapan apa pun (penerapan adalah untuk layanan mikro tertentu dalam wilayah, sel, silo, dan zona ketersediaan tertentu) yang tampaknya memiliki CPU pemanfaatan yang lebih tinggi dibandingkan dengan penerapan lainnya. Anda kemudian dapat menelusuri untuk mendapatkan pemahaman yang lebih baik menggunakan data mentah. Karena penelusuran ini mungkin jarang terjadi dan hanya mengakses data yang relevan dengan penerapan, Anda dapat menggunakan data mentah untuk analisis ini dan tidak perlu menggunakan kueri terjadwal.

Per penyebaran menelusuri

Dasbor di bawah ini menyediakan penelusuran ke statistik yang lebih terperinci dan tingkat server dalam penerapan tertentu. Untuk membantu Anda menelusuri berbagai bagian armada Anda, dasbor ini menggunakan variabel seperti wilayah, sel, silo, layanan mikro, dan availability_zone. Ini kemudian menunjukkan beberapa statistik agregat untuk penerapan itu.



Dalam kueri di bawah ini, Anda dapat melihat bahwa nilai yang dipilih dalam drop-down variabel digunakan sebagai predikat dalam `WHERE` klausa kueri, yang memungkinkan Anda untuk hanya fokus pada data untuk penerapan. Dan kemudian panel memplot CPU metrik agregat untuk instance dalam penerapan itu. Anda dapat menggunakan data mentah untuk melakukan penelusuran ini dengan latensi kueri interaktif untuk mendapatkan wawasan yang lebih dalam.

```
SELECT bin(time, 5m) as minute,
    ROUND(AVG(cpu_user), 2) AS avg_value,
```

```
    ROUND(APPROX_PERCENTILE(cpu_user, 0.9), 2) AS p90_value,
    ROUND(APPROX_PERCENTILE(cpu_user, 0.95), 2) AS p95_value,
    ROUND(APPROX_PERCENTILE(cpu_user, 0.99), 2) AS p99_value
FROM "raw_data"."devops"
WHERE time BETWEEN from_milliseconds(1636527099476) AND
 from_milliseconds(1636613499476)
    AND region = 'eu-west-1'
    AND cell = 'eu-west-1-cell-10'
    AND silo = 'eu-west-1-cell-10-silo-1'
    AND microservice_name = 'demeter'
    AND availability_zone = 'eu-west-1-3'
    AND measure_name = 'metrics'
GROUP BY bin(time, 5m)
ORDER BY 1
```

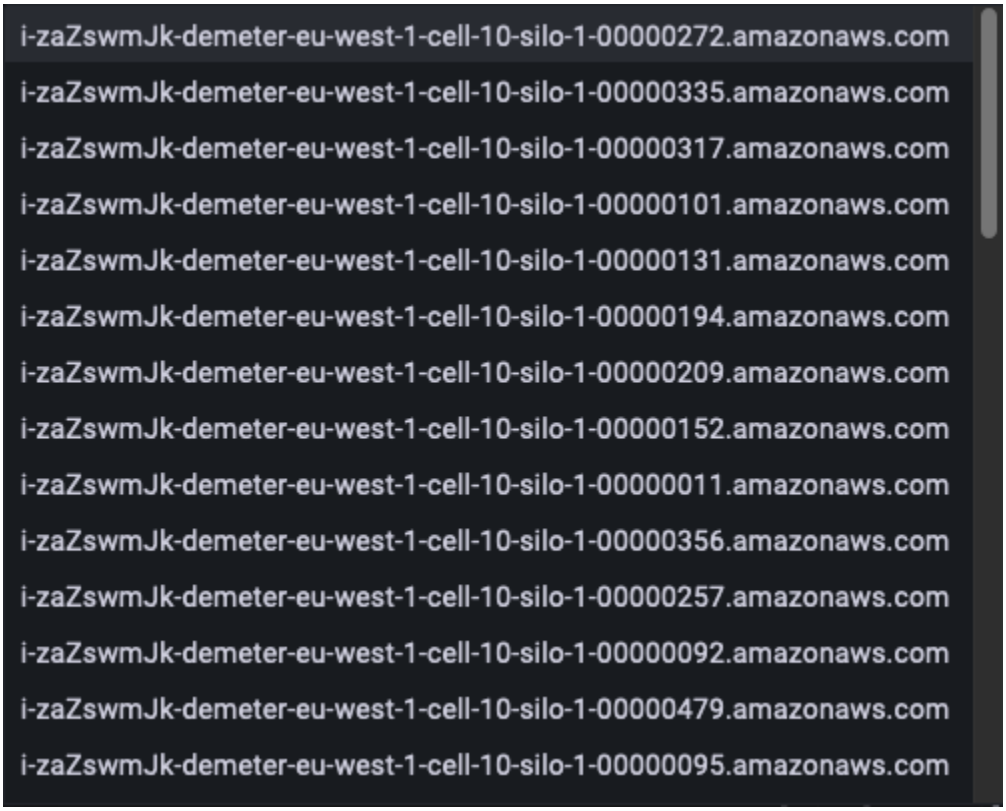
Statistik tingkat instans

Dasbor ini selanjutnya menghitung variabel lain yang juga mencantumkan server/instance dengan CPU pemanfaatan tinggi, diurutkan dalam urutan pemanfaatan yang menurun. Kueri yang digunakan untuk menghitung variabel ini ditampilkan di bawah ini.

```
WITH microservice_cell_avg AS (
    SELECT AVG(cpu_user) AS microservice_avg_metric
    FROM "raw_data"."devops"
    WHERE $__timeFilter
        AND measure_name = 'metrics'
        AND region = '${region}'
        AND cell = '${cell}'
        AND silo = '${silo}'
        AND availability_zone = '${availability_zone}'
        AND microservice_name = '${microservice}'
), instance_avg AS (
    SELECT instance_name,
        AVG(cpu_user) AS instance_avg_metric
    FROM "raw_data"."devops"
    WHERE $__timeFilter
        AND measure_name = 'metrics'
        AND region = '${region}'
        AND cell = '${cell}'
        AND silo = '${silo}'
        AND microservice_name = '${microservice}'
        AND availability_zone = '${availability_zone}'
    GROUP BY availability_zone, instance_name
```

```
)
SELECT i.instance_name
FROM instance_avg i CROSS JOIN microservice_cell_avg m
WHERE i.instance_avg_metric > (1 + ${utilization_threshold}) *
 m.microservice_avg_metric
ORDER BY i.instance_avg_metric DESC
```

Dalam kueri sebelumnya, variabel dihitung ulang secara dinamis tergantung pada nilai yang dipilih untuk variabel lainnya. Setelah variabel diisi untuk penerapan, Anda dapat memilih instance individual dari daftar untuk lebih memvisualisasikan metrik dari instance tersebut. Anda dapat memilih instance yang berbeda dari drop-down nama instance seperti yang terlihat dari snapshot di bawah ini.



Panel sebelumnya menunjukkan statistik untuk contoh yang dipilih dan di bawah ini adalah kueri yang digunakan untuk mengambil statistik ini.

```
SELECT BIN(time, 30m) AS time_bin,
    AVG(cpu_user) AS avg_cpu,
    ROUND(APPROX_PERCENTILE(cpu_user, 0.99), 2) as p99_cpu
FROM "raw_data"."devops"
WHERE time BETWEEN from_milliseconds(1636527099477) AND
 from_milliseconds(1636613499477)
    AND measure_name = 'metrics'
    AND region = 'eu-west-1' AND cell = 'eu-west-1-cell-10' AND silo = 'eu-west-1-
cell-10-silo-1'
    AND availability_zone = 'eu-west-1-3' AND microservice_name = 'demeter'
    AND instance_name = 'i-zaZswmJk-demeter-eu-west-1-cell-10-
silo-1-00000272.amazonaws.com'
GROUP BY BIN(time, 30m)
ORDER BY time_bin desc
```

```
SELECT BIN(time, 30m) AS time_bin,
    AVG(memory_used) AS avg_memory,
    ROUND(APPROX_PERCENTILE(memory_used, 0.99), 2) as p99_memory
FROM "raw_data"."devops"
WHERE time BETWEEN from_milliseconds(1636527099477) AND
 from_milliseconds(1636613499477)
    AND measure_name = 'metrics'
```

```
    AND region = 'eu-west-1' AND cell = 'eu-west-1-cell-10' AND silo = 'eu-west-1-
cell-10-silo-1'
    AND availability_zone = 'eu-west-1-3' AND microservice_name = 'demeter'
    AND instance_name = 'i-zaZswmJk-demeter-eu-west-1-cell-10-
silo-1-00000272.amazonaws.com'
GROUP BY BIN(time, 30m)
ORDER BY time_bin desc
```

```
SELECT COUNT(gc_pause)
FROM "raw_data"."devops"
WHERE time BETWEEN from_milliseconds(1636527099477) AND
 from_milliseconds(1636613499478)
    AND measure_name = 'events'
    AND region = 'eu-west-1' AND cell = 'eu-west-1-cell-10' AND silo = 'eu-west-1-
cell-10-silo-1'
    AND availability_zone = 'eu-west-1-3' AND microservice_name = 'demeter'
    AND instance_name = 'i-zaZswmJk-demeter-eu-west-1-cell-10-
silo-1-00000272.amazonaws.com'
```

```
SELECT avg(gc_pause) as avg, round(approx_percentile(gc_pause, 0.99), 2) as p99
FROM "raw_data"."devops"
WHERE time BETWEEN from_milliseconds(1636527099478) AND
 from_milliseconds(1636613499478)
    AND measure_name = 'events'
    AND region = 'eu-west-1' AND cell = 'eu-west-1-cell-10' AND silo = 'eu-west-1-
cell-10-silo-1'
    AND availability_zone = 'eu-west-1-3' AND microservice_name = 'demeter'
    AND instance_name = 'i-zaZswmJk-demeter-eu-west-1-cell-10-
silo-1-00000272.amazonaws.com'
```

```
SELECT BIN(time, 30m) AS time_bin,
    AVG(disk_io_reads) AS avg,
    ROUND(APPROX_PERCENTILE(disk_io_reads, 0.99), 2) as p99
FROM "raw_data"."devops"
WHERE time BETWEEN from_milliseconds(1636527099478) AND
 from_milliseconds(1636613499478)
    AND measure_name = 'metrics'
    AND region = 'eu-west-1' AND cell = 'eu-west-1-cell-10' AND silo = 'eu-west-1-
cell-10-silo-1'
    AND availability_zone = 'eu-west-1-3' AND microservice_name = 'demeter'
    AND instance_name = 'i-zaZswmJk-demeter-eu-west-1-cell-10-
silo-1-00000272.amazonaws.com'
```

```
GROUP BY BIN(time, 30m)
ORDER BY time_bin desc
```

Mengoptimalkan biaya dengan membagikan kueri terjadwal di seluruh dasbor

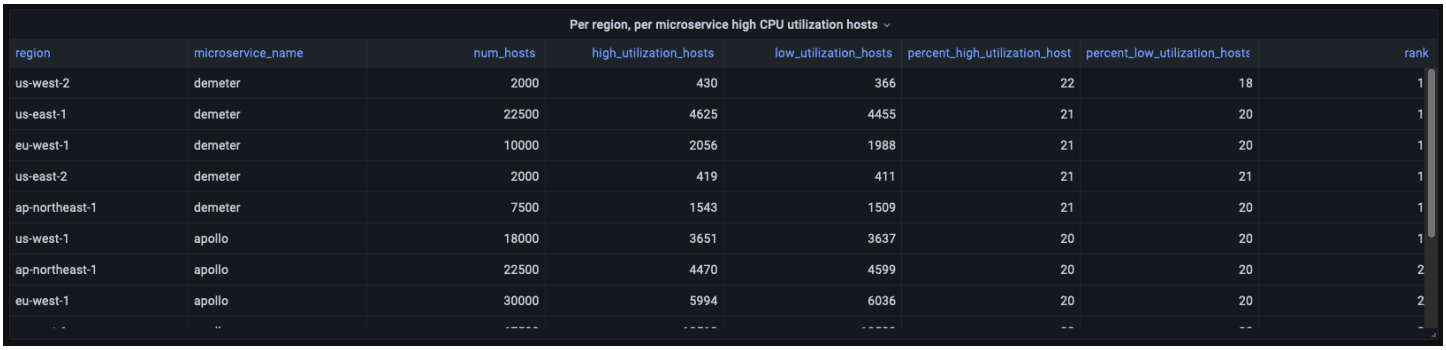
Dalam contoh ini, kita akan melihat skenario di mana beberapa panel dasbor menampilkan variasi informasi serupa (menemukan CPU host tinggi dan fraksi armada dengan CPU pemanfaatan tinggi) dan bagaimana Anda dapat menggunakan kueri terjadwal yang sama untuk pra-perhitungan hasil yang kemudian digunakan untuk mengisi beberapa panel. Penggunaan kembali ini lebih mengoptimalkan biaya Anda di mana alih-alih menggunakan kueri terjadwal yang berbeda, satu untuk setiap panel, Anda hanya menggunakan pemilik.

Panel dasbor dengan data mentah

CPUpemanfaatan per wilayah per microservice

Panel pertama menghitung instance yang CPU pemanfaatan rata-ratanya merupakan ambang batas di bawah atau di atas CPU pemanfaatan di atas untuk penerapan tertentu dalam wilayah, sel, silo, zona ketersediaan, dan layanan mikro. Kemudian menyortir wilayah dan layanan mikro yang memiliki persentase host tertinggi dengan pemanfaatan tinggi. Ini membantu mengidentifikasi seberapa panas server dari penyebaran tertentu berjalan, dan kemudian menelusuri untuk lebih memahami masalah.

Kueri untuk panel menunjukkan fleksibilitas SQL dukungan Timestream untuk LiveAnalytics melakukan tugas analitis yang kompleks dengan ekspresi tabel umum, fungsi jendela, gabungan, dan sebagainya.



Per region, per microservice high CPU utilization hosts

region	microservice_name	num_hosts	high_utilization_hosts	low_utilization_hosts	percent_high_utilization_host	percent_low_utilization_hosts	rank
us-west-2	demeter	2000	430	366	22	18	1
us-east-1	demeter	22500	4625	4455	21	20	1
eu-west-1	demeter	10000	2056	1988	21	20	1
us-east-2	demeter	2000	419	411	21	21	1
ap-northeast-1	demeter	7500	1543	1509	21	20	1
us-west-1	apollo	18000	3651	3637	20	20	1
ap-northeast-1	apollo	22500	4470	4599	20	20	2
eu-west-1	apollo	30000	5994	6036	20	20	2

Permintaan:

```
WITH microservice_cell_avg AS (
    SELECT region, cell, silo, availability_zone, microservice_name, AVG(cpu_user) AS
 microservice_avg_metric
    FROM "raw_data"."devops"
```

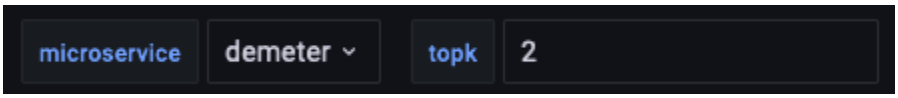
```
    WHERE time BETWEEN from_milliseconds(1636526593876) AND
 from_milliseconds(1636612993876)
        AND measure_name = 'metrics'
    GROUP BY region, cell, silo, availability_zone, microservice_name
), instance_avg AS (
    SELECT region, cell, silo, availability_zone, microservice_name, instance_name,
        AVG(cpu_user) AS instance_avg_metric
    FROM "raw_data"."devops"
    WHERE time BETWEEN from_milliseconds(1636526593876) AND
 from_milliseconds(1636612993876)
        AND measure_name = 'metrics'
    GROUP BY region, cell, silo, availability_zone, microservice_name, instance_name
), instances_above_threshold AS (
  SELECT i.*,
    CASE WHEN i.instance_avg_metric > (1 + 0.2) * m.microservice_avg_metric THEN 1 ELSE
 0 END AS high_utilization,
    CASE WHEN i.instance_avg_metric < (1 - 0.2) * m.microservice_avg_metric THEN 1 ELSE
 0 END AS low_utilization
  FROM instance_avg i INNER JOIN microservice_cell_avg m
    ON i.region = m.region AND i.cell = m.cell AND i.silo = m.silo AND
 i.availability_zone = m.availability_zone
      AND m.microservice_name = i.microservice_name
), per_deployment_high AS (
SELECT region, microservice_name, COUNT(*) AS num_hosts, SUM(high_utilization) AS
 high_utilization_hosts, SUM(low_utilization) AS low_utilization_hosts,
    ROUND(SUM(high_utilization) * 100.0 / COUNT(*), 0) AS
 percent_high_utilization_hosts,
    ROUND(SUM(low_utilization) * 100.0 / COUNT(*), 0) AS percent_low_utilization_hosts
FROM instances_above_threshold
GROUP BY region, microservice_name
), per_region_ranked AS (
    SELECT *,
        DENSE_RANK() OVER (PARTITION BY region ORDER BY percent_high_utilization_hosts
 DESC, high_utilization_hosts DESC) AS rank
    FROM per_deployment_high
)
SELECT *
FROM per_region_ranked
WHERE rank <= 2
ORDER BY percent_high_utilization_hosts desc, rank asc
```

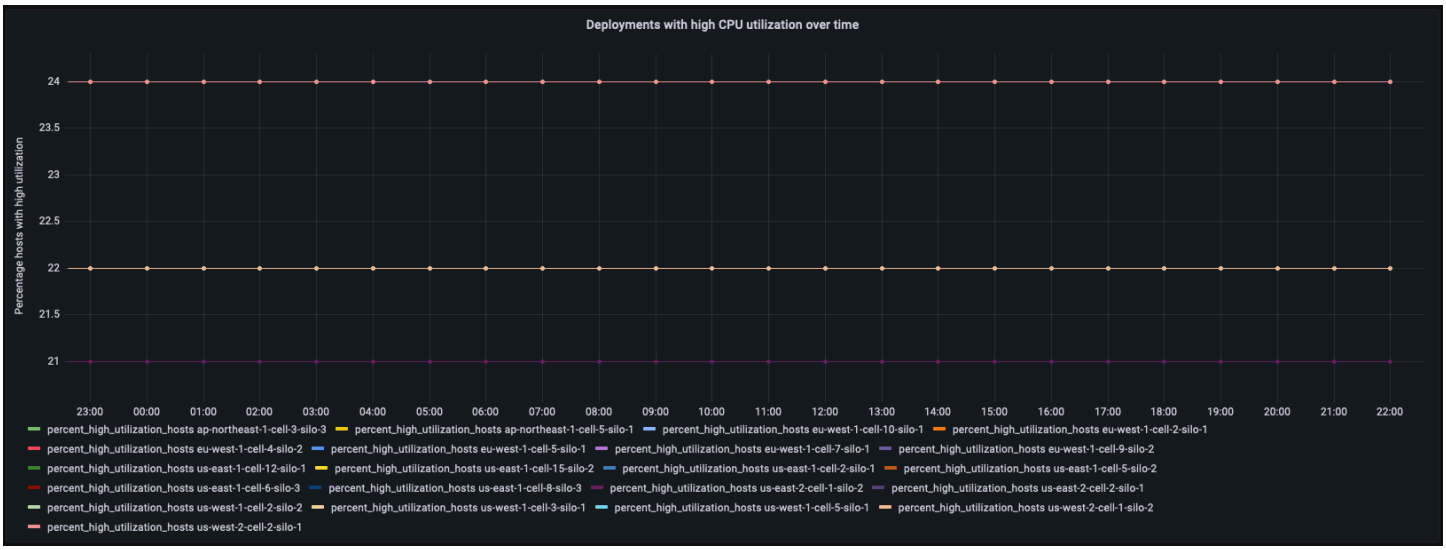
Telusuri ke layanan mikro untuk menemukan hot spot

Dasbor berikutnya memungkinkan Anda untuk mengebor lebih dalam ke salah satu layanan mikro untuk mengetahui wilayah, sel, dan silo tertentu untuk layanan mikro yang menjalankan fraksi fraksi armadanya pada pemanfaatan yang lebih tinggi. CPU Misalnya, di dasbor lebar armada Anda melihat demeter layanan mikro muncul di beberapa posisi peringkat teratas, jadi di dasbor ini, Anda ingin mengebor lebih dalam ke layanan mikro itu.

Dasbor ini menggunakan variabel untuk memilih layanan mikro untuk ditelusuri, dan nilai variabel diisi menggunakan nilai unik dimensi. Setelah Anda memilih layanan mikro, sisa dasbor akan diperbarui.

Seperti yang Anda lihat di bawah, panel pertama memplot persentase host dalam penerapan (wilayah, sel, dan silo untuk layanan mikro) dari waktu ke waktu, dan kueri terkait yang digunakan untuk memplot dasbor. Plot ini sendiri mengidentifikasi penyebaran tertentu yang memiliki persentase host yang lebih tinggi dengan tinggi. CPU





Permintaan:

```
WITH microservice_cell_avg AS (
    SELECT region, cell, silo, availability_zone, microservice_name, bin(time, 1h) as
 hour, AVG(cpu_user) AS microservice_avg_metric
    FROM "raw_data"."devops"
    WHERE time BETWEEN from_milliseconds(1636526898831) AND
 from_milliseconds(1636613298831)
        AND measure_name = 'metrics'
        AND microservice_name = 'demeter'
    GROUP BY region, cell, silo, availability_zone, microservice_name, bin(time, 1h)
), instance_avg AS (
```

```
    SELECT region, cell, silo, availability_zone, microservice_name, instance_name,
 bin(time, 1h) as hour,
        AVG(cpu_user) AS instance_avg_metric
    FROM "raw_data"."devops"
    WHERE time BETWEEN from_milliseconds(1636526898831) AND
 from_milliseconds(1636613298831)
        AND measure_name = 'metrics'
        AND microservice_name = 'demeter'
    GROUP BY region, cell, silo, availability_zone, microservice_name, instance_name,
 bin(time, 1h)
), instances_above_threshold AS (
  SELECT i.*,
    CASE WHEN i.instance_avg_metric > (1 + 0.2) * m.microservice_avg_metric THEN 1 ELSE
 0 END AS high_utilization
  FROM instance_avg i INNER JOIN microservice_cell_avg m
    ON i.region = m.region AND i.cell = m.cell AND i.silo = m.silo AND
 i.availability_zone = m.availability_zone
      AND m.microservice_name = i.microservice_name AND m.hour = i.hour
), high_utilization_percent AS (
    SELECT region, cell, silo, microservice_name, hour, COUNT(*) AS num_hosts,
 SUM(high_utilization) AS high_utilization_hosts,
        ROUND(SUM(high_utilization) * 100.0 / COUNT(*), 0) AS
 percent_high_utilization_hosts
    FROM instances_above_threshold
    GROUP BY region, cell, silo, microservice_name, hour
), high_utilization_ranked AS (
    SELECT region, cell, silo, microservice_name,
        DENSE_RANK() OVER (PARTITION BY region ORDER BY
 AVG(percent_high_utilization_hosts) desc, AVG(high_utilization_hosts) desc) AS rank
    FROM high_utilization_percent
    GROUP BY region, cell, silo, microservice_name
)
SELECT hup.silo, CREATE_TIME_SERIES(hour, hup.percent_high_utilization_hosts) AS
 percent_high_utilization_hosts
FROM high_utilization_percent hup INNER JOIN high_utilization_ranked hur
    ON hup.region = hur.region AND hup.cell = hur.cell AND hup.silo = hur.silo AND
 hup.microservice_name = hur.microservice_name
WHERE rank <= 2
GROUP BY hup.region, hup.cell, hup.silo
ORDER BY hup.silo
```

Mengonversi menjadi satu kueri terjadwal yang memungkinkan penggunaan kembali

Penting untuk dicatat bahwa perhitungan serupa dilakukan di berbagai panel di kedua dasbor. Anda dapat menentukan kueri terjadwal terpisah untuk setiap panel. Di sini Anda akan melihat bagaimana Anda dapat lebih mengoptimalkan biaya Anda dengan mendefinisikan satu kueri terjadwal yang hasilnya dapat digunakan untuk merender ketiga panel.

Berikut ini adalah query yang menangkap agregat yang dihitung dan digunakan untuk semua panel yang berbeda. Anda akan mengamati beberapa aspek penting dalam definisi kueri terjadwal ini.

- Fleksibilitas dan kekuatan area SQL permukaan yang didukung oleh kueri terjadwal, di mana Anda dapat menggunakan ekspresi tabel umum, gabungan, pernyataan kasus, dll.
- Anda dapat menggunakan satu kueri terjadwal untuk menghitung statistik pada perincian yang lebih halus daripada dasbor tertentu yang mungkin diperlukan, dan untuk semua nilai yang mungkin digunakan dasbor untuk variabel yang berbeda. Misalnya, Anda akan melihat agregat dihitung di seluruh wilayah, sel, silo, dan layanan mikro. Oleh karena itu, Anda dapat menggabungkan ini untuk membuat agregat tingkat wilayah, atau wilayah, dan tingkat layanan mikro. Demikian pula, kueri yang sama menghitung agregat untuk semua wilayah, sel, silo, dan layanan mikro. Ini memungkinkan Anda untuk menerapkan filter pada kolom ini untuk mendapatkan agregat untuk subset dari nilai. Misalnya, Anda dapat menghitung agregat untuk satu wilayah, katakanlah us-east-1, atau salah satu layanan mikro mengatakan demeter atau menelusuri penerapan tertentu dalam wilayah, sel, silo, dan layanan mikro. Pendekatan ini lebih mengoptimalkan biaya Anda untuk mempertahankan agregat yang telah dihitung sebelumnya.

```
WITH microservice_cell_avg AS (
    SELECT region, cell, silo, availability_zone, microservice_name, bin(time, 1h) as
 hour, AVG(cpu_user) AS microservice_avg_metric
    FROM raw_data.devops
    WHERE time BETWEEN bin(@scheduled_runtime, 1h) - 1h AND bin(@scheduled_runtime, 1h)
 + 1h
        AND measure_name = 'metrics'
    GROUP BY region, cell, silo, availability_zone, microservice_name, bin(time, 1h)

), instance_avg AS (
    SELECT region, cell, silo, availability_zone, microservice_name, instance_name,
 bin(time, 1h) as hour,
        AVG(cpu_user) AS instance_avg_metric
   FROM raw_data.devops
```

```
   WHERE time BETWEEN bin(@scheduled_runtime, 1h) - 1h AND bin(@scheduled_runtime, 1h)
 + 1h
       AND measure_name = 'metrics'
   GROUP BY region, cell, silo, availability_zone, microservice_name, instance_name,
 bin(time, 1h)
), instances_above_threshold AS (
    SELECT i.*,
        CASE WHEN i.instance_avg_metric > (1 + 0.2) * m.microservice_avg_metric THEN 1
 ELSE 0 END AS high_utilization,
        CASE WHEN i.instance_avg_metric < (1 - 0.2) * m.microservice_avg_metric THEN 1
 ELSE 0 END AS low_utilization
   FROM instance_avg i INNER JOIN microservice_cell_avg m
       ON i.region = m.region AND i.cell = m.cell AND i.silo = m.silo AND
 i.availability_zone = m.availability_zone
          AND m.microservice_name = i.microservice_name AND m.hour = i.hour
)
SELECT region, cell, silo, microservice_name, hour,
     COUNT(*) AS num_hosts, SUM(high_utilization) AS high_utilization_hosts,
 SUM(low_utilization) AS low_utilization_hosts
FROM instances_above_threshold GROUP BY region, cell, silo, microservice_name, hour
```

Berikut ini adalah definisi query terjadwal untuk query sebelumnya. Ekspresi jadwal, dikonfigurasi untuk menyegarkan setiap 30 menit, dan menyegarkan data hingga satu jam yang lalu, sekali lagi menggunakan konstruksi bin (@scheduled_runtime, 1h) untuk mendapatkan acara satu jam penuh. Bergantung pada persyaratan kesegaran aplikasi Anda, Anda dapat mengonfigurasinya untuk menyegarkan lebih atau lebih jarang. Dengan menggunakan WHERE time BETWEEN bin (@scheduled_runtime, 1h) - 1h AND bin (@scheduled_runtime, 1h) + 1h, kami dapat memastikan bahwa meskipun Anda menyegarkan setiap 15 menit sekali, Anda akan mendapatkan data satu jam penuh untuk jam saat ini dan jam sebelumnya.

Nanti, Anda akan melihat bagaimana ketiga panel menggunakan agregat ini yang ditulis ke tabel deployment_cpu_stats_per_hr untuk memvisualisasikan metrik yang relevan dengan panel.

```
{
    "Name": "MultiPT30mHighCpuDeploymentsPerHr",
    "QueryString": "WITH microservice_cell_avg AS (    SELECT region, cell,
 silo, availability_zone, microservice_name, bin(time, 1h) as hour, AVG(cpu_user)
 AS microservice_avg_metric    FROM raw_data.devops    WHERE time BETWEEN
 bin(@scheduled_runtime, 1h) - 1h AND bin(@scheduled_runtime, 1h) + 1h    AND
 measure_name = 'metrics'    GROUP BY region, cell, silo, availability_zone,
 microservice_name, bin(time, 1h)    ), instance_avg AS (    SELECT region,
 cell, silo, availability_zone, microservice_name, instance_name, bin(time, 1h)
```

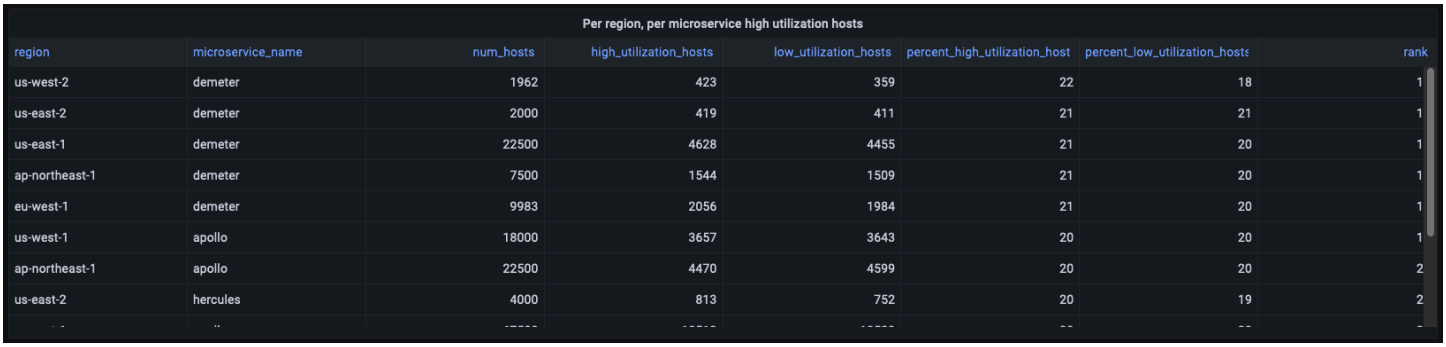
```
as hour,    AVG(cpu_user) AS instance_avg_metric    FROM raw_data.devops
WHERE time BETWEEN bin(@scheduled_runtime, 1h) - 1h AND bin(@scheduled_runtime,
1h) + 1h    AND measure_name = 'metrics'    GROUP BY region, cell, silo,
availability_zone, microservice_name, instance_name, bin(time, 1h)    ),
instances_above_threshold AS (    SELECT i.*,    CASE WHEN i.instance_avg_metric >
(1 + 0.2) * m.microservice_avg_metric THEN 1 ELSE 0 END AS high_utilization,    CASE
WHEN i.instance_avg_metric < (1 - 0.2) * m.microservice_avg_metric THEN 1 ELSE 0 END
AS low_utilization    FROM instance_avg i INNER JOIN microservice_cell_avg m    ON
i.region = m.region AND i.cell = m.cell AND i.silo = m.silo AND i.availability_zone
= m.availability_zone    AND m.microservice_name = i.microservice_name AND m.hour =
i.hour    )     SELECT region, cell, silo, microservice_name, hour,         COUNT(*)
AS num_hosts, SUM(high_utilization) AS high_utilization_hosts, SUM(low_utilization) AS
low_utilization_hosts    FROM instances_above_threshold GROUP BY region, cell, silo,
microservice_name, hour",
   "ScheduleConfiguration": {
       "ScheduleExpression": "cron(0/30 * * * ? *)"
   },
   "NotificationConfiguration": {
       "SnsConfiguration": {
           "TopicArn": "******"
       }
   },
   "TargetConfiguration": {
       "TimestreamConfiguration": {
           "DatabaseName": "derived",
           "TableName": "deployment_cpu_stats_per_hr",
           "TimeColumn": "hour",
           "DimensionMappings": [
               {
                   "Name": "region",
                   "DimensionValueType": "VARCHAR"
               },
               {
                   "Name": "cell",
                   "DimensionValueType": "VARCHAR"
               },
               {
                   "Name": "silo",
                   "DimensionValueType": "VARCHAR"
               },
               {
                   "Name": "microservice_name",
                   "DimensionValueType": "VARCHAR"
               }
```

```
            ],
            "MultiMeasureMappings": {
                "TargetMultiMeasureName": "cpu_user",
                "MultiMeasureAttributeMappings": [
                    {
                        "SourceColumn": "num_hosts",
                        "MeasureValueType": "BIGINT"
                    },
                    {
                        "SourceColumn": "high_utilization_hosts",
                        "MeasureValueType": "BIGINT"
                    },
                    {
                        "SourceColumn": "low_utilization_hosts",
                        "MeasureValueType": "BIGINT"
                    }
                ]
            }
        }
    },
    "ErrorReportConfiguration": {
        "S3Configuration" : {
            "BucketName" : "******",
            "ObjectKeyPrefix": "errors",
            "EncryptionOption": "SSE_S3"
        }
    },
    "ScheduledQueryExecutionRoleArn": "******"
}
```

Dasbor dari hasil yang dihitung sebelumnya

Host CPU pemanfaatan tinggi

Untuk host pemanfaatan tinggi, Anda akan melihat bagaimana panel yang berbeda menggunakan data dari deployment_cpu_stats_per_hr untuk menghitung agregat berbeda yang diperlukan untuk panel. Misalnya, panel ini menyediakan informasi tingkat wilayah, sehingga melaporkan agregat yang dikelompokkan berdasarkan wilayah dan layanan mikro, tanpa memfilter wilayah atau layanan mikro apa pun.



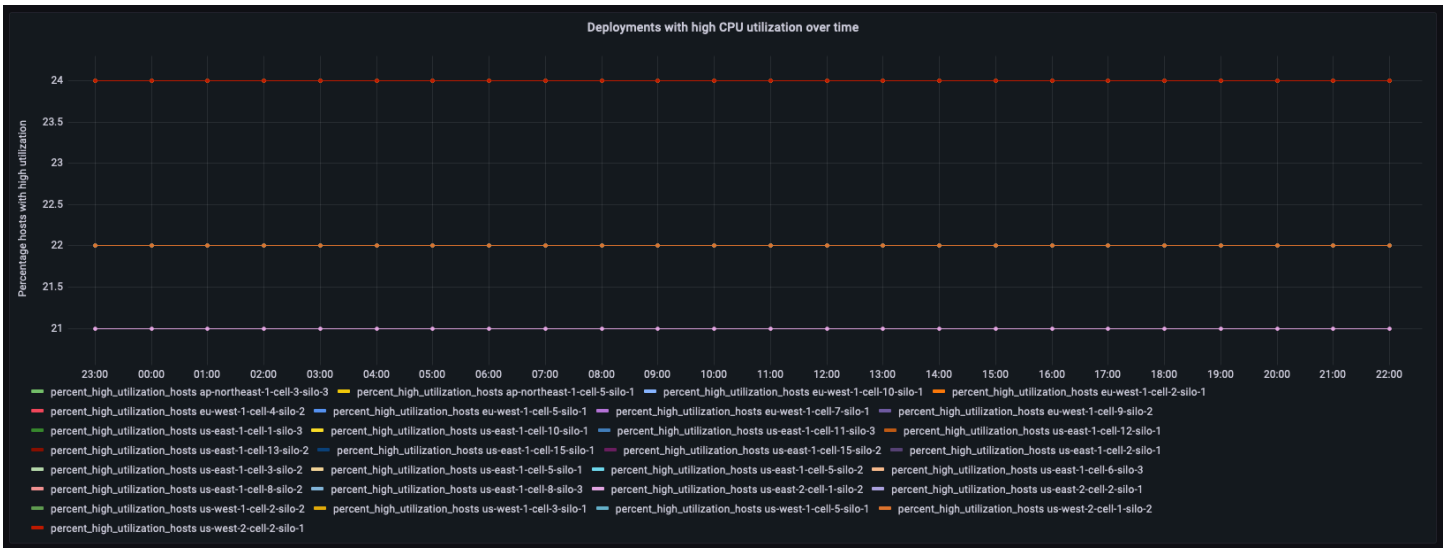
Per region, per microservice high utilization hosts

region	microservice_name	num_hosts	high_utilization_hosts	low_utilization_hosts	percent_high_utilization_host	percent_low_utilization_hosts	rank
us-west-2	demeter	1962	423	359	22	18	1
us-east-2	demeter	2000	419	411	21	21	1
us-east-1	demeter	22500	4628	4455	21	20	1
ap-northeast-1	demeter	7500	1544	1509	21	20	1
eu-west-1	demeter	9983	2056	1984	21	20	1
us-west-1	apollo	18000	3657	3643	20	20	1
ap-northeast-1	apollo	22500	4470	4599	20	20	2
us-east-2	hercules	4000	813	752	20	19	2

```
WITH per_deployment_hosts AS (
    SELECT region, cell, silo, microservice_name,
        AVG(num_hosts) AS num_hosts,
        AVG(high_utilization_hosts) AS high_utilization_hosts,
        AVG(low_utilization_hosts) AS low_utilization_hosts
    FROM "derived"."deployment_cpu_stats_per_hr"
    WHERE time BETWEEN from_milliseconds(1636567785437) AND
 from_milliseconds(1636654185437)
        AND measure_name = 'cpu_user'
    GROUP BY region, cell, silo, microservice_name
), per_deployment_high AS (
    SELECT region, microservice_name,
        SUM(num_hosts) AS num_hosts,
        ROUND(SUM(high_utilization_hosts), 0) AS high_utilization_hosts,
        ROUND(SUM(low_utilization_hosts),0) AS low_utilization_hosts,
        ROUND(SUM(high_utilization_hosts) * 100.0 / SUM(num_hosts)) AS
 percent_high_utilization_hosts,
        ROUND(SUM(low_utilization_hosts) * 100.0 / SUM(num_hosts)) AS
 percent_low_utilization_hosts
    FROM per_deployment_hosts
    GROUP BY region, microservice_name
),
per_region_ranked AS (
    SELECT *,
        DENSE_RANK() OVER (PARTITION BY region ORDER BY percent_high_utilization_hosts
 DESC, high_utilization_hosts DESC) AS rank
    FROM per_deployment_high
)
SELECT *
FROM per_region_ranked
WHERE rank <= 2
ORDER BY percent_high_utilization_hosts desc, rank asc
```

Telusuri ke layanan mikro untuk menemukan penerapan CPU penggunaan tinggi

Contoh berikutnya ini lagi menggunakan tabel turunan deployment_cpu_stats_per_hr, tetapi sekarang menerapkan filter untuk layanan mikro tertentu (demeter dalam contoh ini, karena melaporkan host pemanfaatan tinggi di dasbor agregat). Panel ini melacak persentase host CPU pemanfaatan tinggi dari waktu ke waktu.




```
WITH high_utilization_percent AS (
    SELECT region, cell, silo, microservice_name, bin(time, 1h) AS hour, MAX(num_hosts)
 AS num_hosts,
        MAX(high_utilization_hosts) AS high_utilization_hosts,
        ROUND(MAX(high_utilization_hosts) * 100.0 / MAX(num_hosts)) AS
 percent_high_utilization_hosts
    FROM "derived"."deployment_cpu_stats_per_hr"
    WHERE time BETWEEN from_milliseconds(1636525800000) AND
 from_milliseconds(1636612200000)
        AND measure_name = 'cpu_user'
        AND microservice_name = 'demeter'
    GROUP BY region, cell, silo, microservice_name, bin(time, 1h)
), high_utilization_ranked AS (
    SELECT region, cell, silo, microservice_name,
        DENSE_RANK() OVER (PARTITION BY region ORDER BY
 AVG(percent_high_utilization_hosts) desc, AVG(high_utilization_hosts) desc) AS rank
    FROM high_utilization_percent
    GROUP BY region, cell, silo, microservice_name
)
SELECT hup.silo, CREATE_TIME_SERIES(hour, hup.percent_high_utilization_hosts) AS
 percent_high_utilization_hosts
FROM high_utilization_percent hup INNER JOIN high_utilization_ranked hur
```

```
    ON hup.region = hur.region AND hup.cell = hur.cell AND hup.silo = hur.silo AND
 hup.microservice_name = hur.microservice_name
WHERE rank <= 2
GROUP BY hup.region, hup.cell, hup.silo
ORDER BY hup.silo
```

Membandingkan kueri pada tabel dasar dengan kueri hasil kueri terjadwal

Dalam contoh query Timestream ini, kita menggunakan skema berikut, contoh query, dan output untuk membandingkan query pada tabel dasar dengan query pada tabel turunan hasil query terjadwal. Dengan kueri terjadwal yang terencana dengan baik, Anda bisa mendapatkan tabel turunan dengan lebih sedikit baris dan karakteristik lain yang dapat menghasilkan kueri lebih cepat daripada yang mungkin terjadi pada tabel dasar asli.

Untuk video yang menjelaskan skenario ini, lihat Meningkatkan kinerja kueri dan mengurangi biaya menggunakan kueri terjadwal di Amazon LiveAnalytics Timestream untuk.

Untuk contoh ini, kami menggunakan skenario berikut:

- Wilayah — us-east-1
- Tabel dasar - `"clickstream"."shopping"`
- Tabel turunan — `"clickstream"."aggregate"`

Tabel dasar

Berikut ini menjelaskan skema untuk tabel dasar.

Kolom	Tipe	Timestream untuk jenis LiveAnalytics atribut
saluran	varchar	MULTI
deskripsi	varchar	MULTI
kejadian	varchar	DIMENSION
ip_alamat	varchar	DIMENSION
ukuran_nama	varchar	MEASURE_NAME

Kolom	Tipe	Timestream untuk jenis LiveAnalytics atribut
produk	varchar	MULTI
product_id	varchar	MULTI
kuantitas	double	MULTI
kueri	varchar	MULTI
session_id	varchar	DIMENSION
user_group	varchar	DIMENSION
user_id	varchar	DIMENSION

Berikut ini menjelaskan langkah-langkah untuk tabel dasar. Tabel dasar mengacu pada tabel di Timestream tempat kueri terjadwal dijalankan.

- ukuran_nama — `metrics`
- data — multi
- dimensi:

```
[ ( user_group, varchar ),( user_id, varchar ),( session_id, varchar ),( ip_address,
 varchar ),( event, varchar ) ]
```

Kueri pada tabel dasar

Berikut ini adalah kueri ad-hoc yang mengumpulkan hitungan dengan agregat 5 menit dalam rentang waktu tertentu.

```
SELECT BIN(time, 5m) as time,
channel,
product_id,
SUM(quantity) as product_quantity
FROM "clickstream"."shopping"
WHERE BIN(time, 5m) BETWEEN '2023-05-11 10:10:00.000000000' AND '2023-05-11
 10:30:00.000000000'
```

```
AND channel = 'Social media'
and product_id = '431412'
GROUP BY BIN(time, 5m),channel,product_id
```

Output:

```
duration:1.745 sec
Bytes scanned: 29.89 MB
Query Id: AEBQEANMHG7MHHBHCKJ3BSOE3QUGIDBGWCCP5I6J6YUW5CVJZ2M3JCJ27QRMM7A
Row count:5
```

Permintaan terjadwal

Berikut ini adalah kueri terjadwal yang berjalan setiap 5 menit.

```
SELECT BIN(time, 5m) as time, channel as measure_name, product_id, product,
SUM(quantity) as product_quantity
FROM "clickstream"."shopping"
WHERE time BETWEEN BIN(@scheduled_runtime, 5m) - 10m AND BIN(@scheduled_runtime, 5m) -
 5m
AND channel = 'Social media'
GROUP BY BIN(time, 5m), channel, product_id, product
```

Kueri pada tabel turunan

Berikut ini adalah query ad-hoc pada tabel turunan. Tabel turunan mengacu pada tabel Timestream yang berisi hasil kueri terjadwal.

```
SELECT time, measure_name, product_id,product_quantity
FROM "clickstream"."aggregate"
WHERE time BETWEEN '2023-05-11 10:10:00.000000000' AND '2023-05-11 10:30:00.000000000'
AND measure_name = 'Social media'
and product_id = '431412'
```

Output:

```
duration: 0.2960 sec
Bytes scanned: 235.00 B
QueryID: AEBQEANMHHAAQU4FFTT6CFM6UYXTL4SMLZV22MFP4KV2Z7IRVOPLOMLDD6BR33Q
Row count: 5
```

Perbandingan

Berikut ini adalah perbandingan hasil query pada tabel dasar dengan query pada tabel turunan. Kueri yang sama pada tabel turunan yang memiliki hasil agregat yang dilakukan melalui kueri terjadwal selesai lebih cepat dengan lebih sedikit byte yang dipindai.

Hasil ini menunjukkan nilai penggunaan kueri terjadwal untuk mengumpulkan data untuk kueri yang lebih cepat.

	Kueri pada tabel dasar	Kueri pada tabel turunan
Durasi	1,745 dtk	0.2960 dtk
Byte dipindai	29,89 MB	235 byte
Hitungan baris	5	5

Menggunakan UNLOAD untuk mengekspor hasil kueri ke S3 dari Timestream untuk LiveAnalytics

Amazon Timestream untuk LiveAnalytics saat ini memungkinkan Anda mengekspor hasil kueri ke Amazon S3 dengan cara yang hemat biaya dan aman menggunakan pernyataan tersebut. `UNLOAD` Dengan menggunakan `UNLOAD` pernyataan tersebut, Anda sekarang dapat mengekspor data deret waktu ke bucket S3 yang dipilih dalam format Apache Parquet atau Comma Separated Values (CSV), yang memberikan fleksibilitas untuk menyimpan, menggabungkan, dan menganalisis data deret waktu Anda dengan layanan lain. `UNLOAD`Pernyataan ini memungkinkan Anda untuk mengekspor data dengan cara terkompresi, yang mengurangi data yang ditransfer dan ruang penyimpanan yang diperlukan. `UNLOAD`juga mendukung partisi berdasarkan atribut yang dipilih saat mengekspor data, meningkatkan kinerja dan mengurangi waktu pemrosesan layanan hilir yang mengakses data. Selain itu, Anda dapat menggunakan kunci terkelola Amazon S3 (SSE-S3) atau AWS Key Management Service () kunci terkelola (SSE-AWS KMSKMS) untuk mengenkripsi data yang diekspor.

Manfaat UNLOAD dari Timestream untuk LiveAnalytics

Manfaat utama menggunakan `UNLOAD` pernyataan tersebut adalah sebagai berikut.

- Kemudahan operasional — Dengan `UNLOAD` pernyataan tersebut, Anda dapat mengekspor gigabyte data dalam satu permintaan kueri baik dalam Apache Parquet atau CSV format,

memberikan fleksibilitas untuk memilih format yang paling sesuai untuk kebutuhan pemrosesan hilir Anda dan membuatnya lebih mudah untuk membangun data lake.

- Aman dan Hemat biaya — `UNLOAD` pernyataan menyediakan kemampuan untuk mengekspor data Anda ke bucket S3 secara terkompresi dan mengenkripsi (SSE- KMS atau SSE _S3) data Anda menggunakan kunci yang dikelola pelanggan, mengurangi biaya penyimpanan data, dan melindungi dari akses yang tidak sah.
- Kinerja — Menggunakan `UNLOAD` pernyataan, Anda dapat mempartisi data saat mengekspor ke bucket S3. Mempartisi data memungkinkan layanan hilir untuk memproses data secara paralel, mengurangi waktu pemrosesan mereka. Selain itu, layanan hilir hanya dapat memproses data yang mereka butuhkan, mengurangi sumber daya pemrosesan yang diperlukan dan dengan demikian biaya yang terkait.

Gunakan kasus untuk UNLOAD dari Timestream untuk LiveAnalytics

Anda dapat menggunakan `UNLOAD` pernyataan untuk menulis data ke bucket S3 Anda ke yang berikut ini.

- Build Data Warehouse — Anda dapat mengekspor gigabyte hasil kueri ke dalam bucket S3 dan lebih mudah menambahkan data deret waktu ke data lake Anda. Anda dapat menggunakan layanan seperti Amazon Athena dan Amazon Redshift untuk menggabungkan data deret waktu Anda dengan data relevan lainnya untuk mendapatkan wawasan bisnis yang kompleks.
- Membangun jaringan data AI dan ML — `UNLOAD` Pernyataan ini memungkinkan Anda membangun jalur data dengan mudah untuk model pembelajaran mesin Anda yang mengakses data deret waktu, sehingga memudahkan penggunaan data deret waktu dengan layanan seperti Amazon dan SageMaker Amazon. EMR
- Menyederhanakan ETL Pemrosesan — Mengekspor data ke dalam bucket S3 dapat menyederhanakan proses melakukan operasi Extract, Transform, Load (ETL) pada data, memungkinkan Anda menggunakan alat atau layanan pihak ketiga AWS seperti AWS Glue untuk memproses dan mengubah data dengan mulus.

Konsep UNLOAD

Sintaks

```
UNLOAD (SELECT statement)
 TO 's3://bucket-name/folder'
```

```
 WITH ( option = expression [, ...] )
```

`option`dimana

```
{ partitioned_by = ARRAY[ col_name[,…] ]
 | format = [ '{ CSV | PARQUET }' ]
 | compression = [ '{ GZIP | NONE }' ]
 | encryption = [ '{ SSE_KMS | SSE_S3 }' ]
 | kms_key = '<string>'
 | field_delimiter ='<character>'
 | escaped_by = '<character>'
 | include_header = ['{true, false}']
 | max_file_size = '<value>'
 | }
```

Parameter

SELECTpernyataan

Pernyataan query yang digunakan untuk memilih dan mengambil data dari satu atau lebih Timestream untuk LiveAnalytics tabel.

```
(SELECT column 1, column 2, column 3 from database.table
      where measure_name = "ABC" and timestamp between ago (1d) and now() )
```

Klausul TO

```
TO 's3://bucket-name/folder'
```

atau

```
TO 's3://access-point-alias/folder'
```

`TO`Klausa dalam `UNLOAD` pernyataan menentukan tujuan untuk output dari hasil query. Anda perlu menyediakan jalur lengkap, termasuk nama ember Amazon S3 atau Amazon S3 dengan lokasi folder di Amazon S3 access-point-alias tempat Timestream untuk menulis objek file output. LiveAnalytics Bucket S3 harus dimiliki oleh akun yang sama dan di wilayah yang sama. Selain set hasil kueri, Timestream untuk LiveAnalytics menulis file manifes dan metadata ke folder tujuan tertentu.

PARTITIONED_BY klausa

```
partitioned_by = ARRAY [col_name[,…] , (default: none)
```

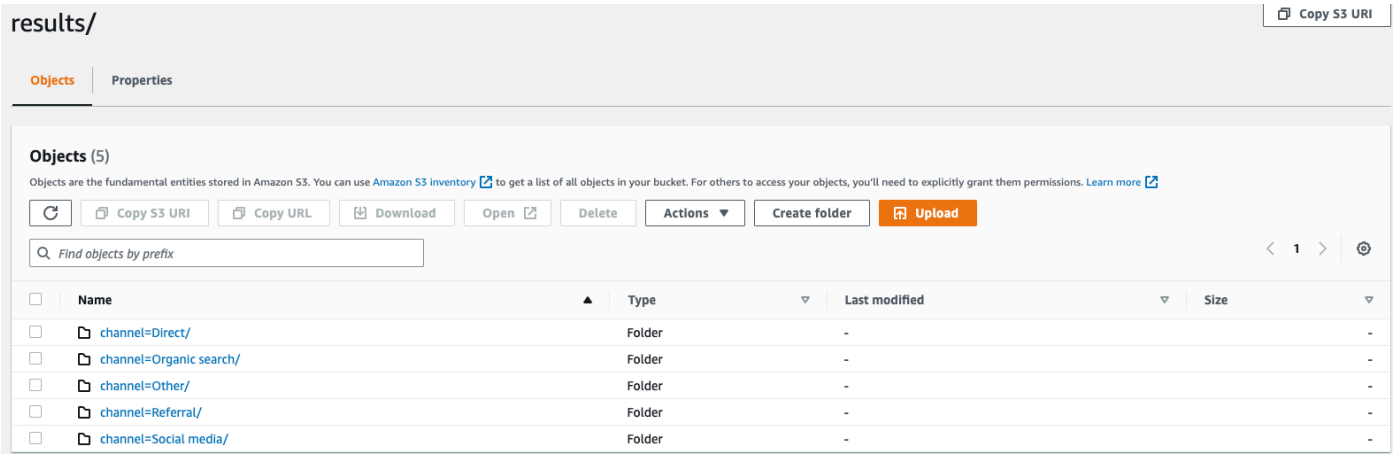
`partitioned_by`Klausa ini digunakan dalam kueri untuk mengelompokkan dan menganalisis data pada tingkat granular. Saat mengekspor hasil kueri ke bucket S3, Anda dapat memilih untuk mempartisi data berdasarkan satu atau beberapa kolom dalam kueri pilih. Saat mempartisi data, data yang diekspor dibagi menjadi himpunan bagian berdasarkan kolom partisi dan setiap subset disimpan dalam folder terpisah. Dalam folder hasil yang berisi data yang diekspor, sub-folder dibuat `folder/results/partition column = partition value/` secara otomatis. Namun, perhatikan bahwa kolom yang dipartisi tidak termasuk dalam file output.

`partitioned_by`bukan klausa wajib dalam sintaks. Jika Anda memilih untuk mengekspor data tanpa partisi apa pun, Anda dapat mengecualikan klausa dalam sintaks.

Example

Dengan asumsi Anda memantau data clickstream situs web Anda dan memiliki 5 saluran lalu lintas yaitu`direct`,,, `Social Media`, `Organic Search`, `Other` dan. `Referral` Saat mengekspor data, Anda dapat memilih untuk mempartisi data menggunakan kolom`Channel`. Dalam folder data Anda`s3://bucketname/results`,, Anda akan memiliki lima folder masing-masing dengan nama saluran masing-masing, misalnya, `s3://bucketname/results/channel=Social Media/`. Dalam folder ini Anda akan menemukan data semua pelanggan yang mendarat di situs web Anda melalui `Social Media` saluran. Demikian pula, Anda akan memiliki folder lain untuk saluran yang tersisa.

Data yang diekspor dipartisi oleh kolom Channel



FORMAT

```
format = [ '{ CSV | PARQUET }' , default: CSV
```

Kata kunci untuk menentukan format hasil kueri yang ditulis ke bucket S3 Anda. Anda dapat mengekspor data baik sebagai nilai dipisahkan koma (CSV) menggunakan koma (,) sebagai pembatas default atau dalam format Apache Parquet, format penyimpanan kolom terbuka yang efisien untuk analitik.

COMPRESSION

```
compression = [ '{ GZIP | NONE }' ], default: GZIP
```

Anda dapat mengompres data yang diekspor menggunakan algoritma kompresi GZIP atau membuatnya tidak dikompresi dengan menentukan opsi. `NONE`

ENCRYPTION

```
encryption = [ '{ SSE_KMS | SSE_S3 }' ], default: SSE_S3
```

File output di Amazon S3 dienkripsi menggunakan opsi enkripsi yang Anda pilih. Selain data Anda, file manifes dan metadata juga dienkripsi berdasarkan opsi enkripsi yang Anda pilih. Saat ini kami mendukung SSE enkripsi _S3 dan SSE _KMS. SSE_S3 adalah enkripsi sisi server dengan Amazon S3 mengenkripsi data menggunakan enkripsi standar enkripsi lanjutan () 256-bit. AES SSE_ KMS adalah enkripsi sisi server untuk mengenkripsi data menggunakan kunci yang dikelola pelanggan.

KMS_KEY

```
kms_key = '<string>'
```

KMSKunci adalah kunci yang ditentukan pelanggan untuk mengenkripsi hasil kueri yang diekspor. KMSKey dikelola dengan aman oleh AWS Key Management Service (AWS KMS) dan digunakan untuk mengenkripsi file data di Amazon S3.

FIELD_DELIMITER

```
field_delimiter ='<character>' , default: (,)
```

Saat mengekspor data dalam CSV format, bidang ini menentukan satu ASCII karakter yang digunakan untuk memisahkan bidang dalam file output, seperti karakter pipa (|), koma (,), atau

tab (/t). Pembatas default untuk CSV file adalah karakter koma. Jika nilai dalam data Anda berisi pembatas yang dipilih, pembatas akan dikutip dengan karakter kutipan. Misalnya, jika nilai dalam data Anda berisi`Time,stream`, maka nilai ini akan dikutip seperti `"Time,stream"` pada data yang diekspor. Karakter kutipan yang digunakan oleh Timestream untuk LiveAnalytics adalah tanda kutip ganda (“).

Hindari menentukan karakter carriage return (ASCII13, hex`0D`, text '\ r') atau karakter line break (ASCII10, hex 0A, text'\n') sebagai `FIELD_DELIMITER` jika Anda ingin memasukkan header dalamCSV, karena itu akan mencegah banyak parser dari dapat mengurai header dengan benar dalam output yang dihasilkan. CSV

ESCAPED_OLEH

```
escaped_by = '<character>', default: (\)
```

Saat mengekspor data dalam CSV format, bidang ini menentukan karakter yang harus diperlakukan sebagai karakter escape dalam file data yang ditulis ke bucket S3. Melarikan diri terjadi dalam skenario berikut:

1. Jika nilai itu sendiri berisi karakter kutipan (“) maka itu akan diloloskan menggunakan karakter escape. Misalnya, jika nilainya`Time"stream`, di mana (\) adalah karakter escape yang dikonfigurasi, maka itu akan lolos sebagai`Time\"stream`.
2. Jika nilai berisi karakter escape dikonfigurasi, itu akan lolos. Misalnya, jika nilainya`Time \stream`, maka itu akan lolos sebagai`Time\\stream`.

> **Note**
>
> Jika output yang diekspor berisi tipe data yang kompleks seperti Array, Rows atau Timeseries, itu akan diserialisasi sebagai string. JSON Berikut adalah contohnya.

Tipe data	Nilai aktual	Bagaimana nilai diloloskan dalam CSV format [serial stringJSON]
Array	`[ 23,24,25 ]`	`"[23,24,25]"`
Baris	`( x=23.0, y=hello )`	`"{\"x\":23.0,\"y\": \"hello\"}"`

Tipe data	Nilai aktual	Bagaimana nilai diloloskan dalam CSV format [serial stringJSON]
Timeseries	`[ ( time=1970-01-01 00:00:00.000000010 , value=100.0 ), ( time=1970-01-01 00:00:00.000000012, value=120.0 ) ]`	`"[{\"time\":\"1970-01-01 00:00:00.000000010Z\",\"value\":100.0},{\"time\":\"1970-01-01 00:00:00.000000012Z\",\"value\":120.0}]"`

INCLUDE_HEADER

```
include_header = 'true' , default: 'false'
```

Saat mengekspor data dalam CSV format, bidang ini memungkinkan Anda menyertakan nama kolom sebagai baris pertama dari file CSV data yang diekspor.

Nilai yang diterima adalah 'true' dan 'false' dan nilai default adalah 'false'. Opsi transformasi teks seperti `escaped_by` dan `field_delimiter` berlaku untuk header juga.

> **Note**
>
> Saat menyertakan header, penting bahwa Anda tidak memilih karakter carriage return (ASCII13, hex 0D, text '\ r') atau karakter pemisah baris (ASCII10, hex 0A, teks'\n') sebagai`FIELD_DELIMITER`, karena itu akan mencegah banyak parser untuk dapat mengurai header dengan benar dalam output yang dihasilkan. CSV

MAX_FILE_SIZE

```
max_file_size = 'X[MB|GB]' , default: '78GB'
```

Bidang ini menentukan ukuran maksimum file yang dibuat `UNLOAD` pernyataan di Amazon S3. `UNLOAD`Pernyataan tersebut dapat membuat beberapa file tetapi ukuran maksimum setiap file yang ditulis ke Amazon S3 akan kira-kira apa yang ditentukan dalam bidang ini.

Nilai bidang harus antara 16 MB dan 78 GB, inklusif. Anda dapat menentukannya dalam bilangan bulat seperti`12GB`, atau dalam desimal seperti atau. `0.5GB` `24.7MB` Nilai default adalah 78 GB.

Ukuran file sebenarnya diperkirakan saat file sedang ditulis, sehingga ukuran maksimum sebenarnya mungkin tidak persis sama dengan angka yang Anda tentukan.

Apa yang ditulis ke ember S3 saya?

Untuk setiap UNLOAD kueri yang berhasil dijalankan, Timestream untuk LiveAnalytics menulis hasil kueri, file metadata, dan file manifes Anda ke dalam bucket S3. Jika Anda telah mempartisi data, Anda memiliki semua folder partisi di folder hasil. File manifes berisi daftar file yang ditulis oleh UNLOAD perintah. File metadata berisi informasi yang menjelaskan karakteristik, properti, dan atribut data tertulis.

Apa nama file yang diekspor?

Nama file yang diekspor berisi dua komponen, komponen pertama adalah QueryID dan komponen kedua adalah pengidentifikasi unik.

CSVberkas

```
S3://bucket_name/results/<queryid>_<UUID>.csv
S3://bucket_name/results/<partitioncolumn>=<partitionvalue>/<queryid>_<UUID>.csv
```

File terkompresi CSV

```
S3://bucket_name/results/<partitioncolumn>=<partitionvalue>/<queryid>_<UUID>.gz
```

File parket

```
S3://bucket_name/results/<partitioncolumn>=<partitionvalue>/<queryid>_<UUID>.parquet
```

File metadata dan Manifest

```
S3://bucket_name/<queryid>_<UUID>_manifest.json
S3://bucket_name/<queryid>_<UUID>_metadata.json
```

Karena data dalam CSV format disimpan pada tingkat file, saat Anda mengompres data saat mengekspor ke S3, file tersebut akan memiliki ekstensi ".gz". Namun, data di Parket dikompresi pada tingkat kolom sehingga bahkan ketika Anda mengompres data saat mengekspor, file tersebut masih akan memiliki ekstensi.parquet.

Informasi apa yang terkandung dalam setiap file?

File manifes

File manifes memberikan informasi tentang daftar file yang diekspor dengan UNLOAD eksekusi. File manifes tersedia di bucket S3 yang disediakan dengan nama file:`s3://<bucket_name>/<queryid>_<UUID>_manifest.json`. File manifes akan berisi url file di folder hasil, jumlah catatan dan ukuran file masing-masing, dan metadata kueri (yang merupakan total byte dan total baris yang diekspor ke S3 untuk kueri).

```
{
  "result_files": [
    {
        "url":"s3://my_timestream_unloads/ec2_metrics/
AEDAGANLHLBH4OLISD3CVOZZRWPX5GV2XCXRBKCVD554N6GWPWWXBP7LSG74V2Q_1448466917_szCL4YgVYzGXj2lS.gz"
        "file_metadata":
            {
                "content_length_in_bytes": 32295,
                "row_count": 10
            }
    },
    {
        "url":"s3://my_timestream_unloads/ec2_metrics/
AEDAGANLHLBH4OLISD3CVOZZRWPX5GV2XCXRBKCVD554N6GWPWWXBP7LSG74V2Q_1448466917_szCL4YgVYzGXj2lS.gz"
        "file_metadata":
            {
                "content_length_in_bytes": 62295,
                "row_count": 20
            }
    },
  ],
  "query_metadata":
    {
      "content_length_in_bytes": 94590,
      "total_row_count": 30,
      "result_format": "CSV",
      "result_version": "Amazon Timestream version 1.0.0"
```

```
    },
  "author": {
        "name": "Amazon Timestream",
        "manifest_file_version": "1.0"
  }
}
```

Metadata

File metadata memberikan informasi tambahan tentang kumpulan data seperti nama kolom, jenis kolom, dan skema. <queryid>File metadata tersedia di bucket S3 yang disediakan dengan nama file: S3: //bucket_name/ _< >_metadata.json UUID

Berikut ini adalah contoh dari file metadata.

```
{
    "ColumnInfo": [
        {
            "Name": "hostname",
            "Type": {
                "ScalarType": "VARCHAR"
            }
        },
        {
            "Name": "region",
            "Type": {
                "ScalarType": "VARCHAR"
            }
        },
        {
            "Name": "measure_name",
            "Type": {
                "ScalarType": "VARCHAR"
            }
        },
        {
            "Name": "cpu_utilization",
            "Type": {
                "TimeSeriesMeasureValueColumnInfo": {
                    "Type": {
                        "ScalarType": "DOUBLE"
                    }
                }
```

```
            }
        }
  ],
  "Author": {
        "Name": "Amazon Timestream",
        "MetadataFileVersion": "1.0"
  }
}
```

Informasi kolom yang dibagikan dalam file metadata memiliki struktur yang sama seperti yang `ColumnInfo` dikirim dalam API respons Kueri untuk `SELECT` kueri.

Hasil

Folder hasil berisi data yang diekspor dalam Apache Parquet atau format. CSV

Contoh

Saat Anda mengirimkan `UNLOAD` kueri seperti di bawah ini melalui QueryAPI,

```
UNLOAD(SELECT user_id, ip_address, event, session_id, measure_name, time, query,
 quantity, product_id, channel
                    FROM sample_clickstream.sample_shopping WHERE time BETWEEN ago(2d)
 AND now())
                TO 's3://my_timestream_unloads/withoutpartition/' WITH ( format='CSV',
 compression='GZIP')
```

`UNLOAD`respon query akan memiliki 1 baris* 3 kolom. 3 kolom tersebut adalah:

- baris tipe BIGINT - menunjukkan jumlah baris yang diekspor
- metadataFile tipe VARCHAR - yang merupakan S3 dari file URI metadata yang diekspor
- manifestFile tipe VARCHAR - yang merupakan S3 dari file manifes URI yang diekspor

Anda akan mendapatkan jawaban berikut dari QueryAPI:

```
{
    "Rows": [
        {
            "Data": [
                {
                    "ScalarValue": "20" # No of rows in output across all files
```

```
                },
                {
                    "ScalarValue": "s3://my_timestream_unloads/withoutpartition/
AEDAAANGH3D7FYHOBQGQQMEAISCJ45B42OWWJMOT4N6RRJICZUA7R25VYVOHJIY_<UUID>_metadata.json"
 #Metadata file
                },
                {
                    "ScalarValue": "s3://my_timestream_unloads/withoutpartition/
AEDAAANGH3D7FYHOBQGQQMEAISCJ45B42OWWJMOT4N6RRJICZUA7R25VYVOHJIY_<UUID>_manifest.json"
 #Manifest file
                }
            ]
        }
    ],
    "ColumnInfo": [
        {
            "Name": "rows",
            "Type": {
                "ScalarType": "BIGINT"
            }
        },
        {
            "Name": "metadataFile",
            "Type": {
                "ScalarType": "VARCHAR"
            }
        },
        {
            "Name": "manifestFile",
            "Type": {
                "ScalarType": "VARCHAR"
            }
        }
    ],
    "QueryId": "AEDAAANGH3D7FYHOBQGQQMEAISCJ45B42OWWJMOT4N6RRJICZUA7R25VYVOHJIY",
    "QueryStatus": {
        "ProgressPercentage": 100.0,
        "CumulativeBytesScanned": 1000,
        "CumulativeBytesMetered": 10000000
    }
}
```

Jenis data

`UNLOAD`Pernyataan ini mendukung semua tipe data Timestream untuk bahasa LiveAnalytics kueri yang dijelaskan dalam Jenis data yang didukung kecuali `time` dan`unknown`.

Prasyarat untuk dari Timestream untuk UNLOAD LiveAnalytics

Berikut ini adalah prasyarat untuk menulis data ke S3 menggunakan dari Timestream untuk. `UNLOAD` LiveAnalytics

- Anda harus memiliki izin untuk membaca data dari Timestream untuk LiveAnalytics tabel yang akan digunakan dalam `UNLOAD` perintah.
- Anda harus memiliki bucket Amazon S3 di AWS Wilayah yang sama dengan Timestream Anda untuk sumber daya. LiveAnalytics
- Untuk bucket S3 yang dipilih, pastikan kebijakan bucket S3 juga memiliki izin untuk mengizinkan Timestream LiveAnalytics mengekspor data.
- Kredensi yang digunakan untuk mengeksekusi `UNLOAD` kueri harus memiliki izin AWS Identity and Access Management (IAM) yang diperlukan yang memungkinkan Timestream untuk menulis data LiveAnalytics ke S3. Contoh kebijakan adalah sebagai berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "timestream:Select",
                "timestream:ListMeasures",
                "timestream:WriteRecords",
                "timestream:Unload"
            ],
            "Resource": "arn:aws:timestream:<region>:<account_id>:database/
<database_name>/table/<table_name>"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketAcl",
                "s3:PutObject",
                "s3:GetObjectMetadata",
                "s3:AbortMultipartUpload"
```

```
            ],
            "Resource": [
                "arn:aws:s3:::<S3_Bucket_Created>",
                "arn:aws:s3:::<S3_Bucket_Created>/*"
            ]
        }
    ]
}
```

Untuk konteks tambahan tentang izin menulis S3 ini, lihat panduan Layanan Penyimpanan Sederhana Amazon. Jika Anda menggunakan KMS kunci untuk mengenkripsi data yang diekspor, lihat berikut ini untuk kebijakan tambahan IAM yang diperlukan.

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
            "kms:DescribeKey",
            "kms:Decrypt",
            "kms:GenerateDataKey*"
        ],
        "Resource": "<account_id>-arn:aws:kms:<region>:<account_id>:key/*",
        "Condition": {
            "ForAnyValue:StringLike": {
                "kms:ResourceAliases": "alias/<Alias_For_Generated_Key>"
            }
        }
    }, {
        "Effect": "Allow",
        "Action": [
            "kms:CreateGrant"
        ],
        "Resource": "<account_id>-arn:aws:kms:<region>:<account_id>:key/*",
        "Condition": {
            "ForAnyValue:StringEquals": {
                "kms:EncryptionContextKeys": "aws:timestream:<database_name>"
            },
            "Bool": {
                "kms:GrantIsForAWSResource": true
            },
            "StringLike": {
```

```
                "kms:ViaService": "timestream.<region>.amazonaws.com"
            },
            "ForAnyValue:StringLike": {
                "kms:ResourceAliases": "alias/<Alias_For_Generated_Key>"
            }
        }
    }
]
}
```

Praktik terbaik untuk UNLOAD dari Timestream untuk LiveAnalytics

Berikut ini adalah praktik terbaik yang terkait dengan UNLOAD perintah.

- Jumlah data yang dapat diekspor ke bucket S3 menggunakan UNLOAD perintah tidak dibatasi. Namun, waktu kueri habis dalam 60 menit dan kami sarankan untuk mengekspor tidak lebih dari 60GB data dalam satu kueri. Jika Anda perlu mengekspor lebih dari 60GB data, pisahkan pekerjaan di beberapa kueri.
- Meskipun Anda dapat mengirim ribuan permintaan ke S3 untuk mengunggah data, disarankan untuk memparalelkan operasi penulisan ke beberapa awalan S3. Lihat dokumentasi di sini. Tingkat API panggilan S3 dapat dibatasi ketika beberapa pembaca/penulis mengakses folder yang sama.
- Mengingat batas panjang kunci S3 untuk menentukan awalan, kami sarankan memiliki nama bucket dan folder dalam 10-15 karakter, terutama saat menggunakan klausa. `partitioned_by`
- Ketika Anda menerima 4XX atau 5XX untuk kueri yang berisi UNLOAD pernyataan, ada kemungkinan bahwa sebagian hasil ditulis ke dalam ember S3. Timestream for LiveAnalytics tidak menghapus data apa pun dari bucket Anda. Sebelum menjalankan UNLOAD kueri lain dengan tujuan S3 yang sama, kami sarankan untuk menghapus file yang dibuat oleh kueri yang gagal secara manual. Anda dapat mengidentifikasi file yang ditulis oleh kueri yang gagal dengan yang sesuai`QueryExecutionId`. Untuk kueri yang gagal, Timestream for LiveAnalytics tidak mengekspor file manifes ke bucket S3.
- Timestream untuk LiveAnalytics menggunakan unggahan multi-bagian untuk mengekspor hasil kueri ke S3. Ketika Anda menerima 4XX atau 5XX dari Timestream LiveAnalytics untuk kueri yang berisi UNLOAD pernyataan, Timestream for LiveAnalytics melakukan aborsi upaya terbaik untuk unggahan multi-bagian tetapi ada kemungkinan bahwa beberapa bagian yang tidak lengkap tertinggal. Oleh karena itu, kami merekomendasikan untuk menyiapkan pembersihan otomatis dari unggahan multi-bagian yang tidak lengkap di bucket S3 Anda dengan mengikuti panduan di sini.

Rekomendasi untuk mengakses data dalam CSV format menggunakan parser CSV

- CSVparser tidak memungkinkan Anda untuk memiliki karakter yang sama dalam karakter pembatas, escape, dan quote.
- Beberapa CSV parser tidak dapat menafsirkan tipe data yang kompleks seperti Array, kami sarankan menafsirkannya melalui deserializer. JSON

Rekomendasi untuk mengakses data dalam format Parket

1. Jika kasus penggunaan Anda memerlukan dukungan karakter UTF -8 dalam skema alias nama kolom, sebaiknya gunakan pustaka Parquet-MR.
2. Stempel waktu dalam hasil Anda direpresentasikan sebagai bilangan bulat 12 byte () INT96
3. Timeseries akan direpresentasikan sebagai`array<row<time, value>>`, struktur bersarang lainnya akan menggunakan tipe data yang sesuai yang didukung dalam format Parket

Menggunakan klausa partition_by

- Kolom yang digunakan di `partitioned_by` bidang harus menjadi kolom terakhir dalam kueri pilih. Jika lebih dari satu kolom digunakan di `partitioned_by` bidang, kolom harus menjadi kolom terakhir dalam kueri pilih dan dalam urutan yang sama seperti yang digunakan di `partition_by` bidang.
- Nilai kolom yang digunakan untuk mempartisi data (`partitioned_by`bidang) hanya dapat berisi ASCII karakter. Sementara Timestream untuk LiveAnalytics memungkinkan UTF -8 karakter dalam nilai, S3 hanya mendukung ASCII karakter sebagai kunci objek.

Contoh kasus penggunaan untuk UNLOAD dari Timestream untuk LiveAnalytics

Asumsikan Anda memantau metrik sesi pengguna, sumber lalu lintas, dan pembelian produk situs web e-commerce Anda. Anda menggunakan Timestream LiveAnalytics untuk memperoleh wawasan real-time tentang perilaku pengguna, penjualan produk, dan melakukan analisis pemasaran pada saluran lalu lintas (pencarian organik, media sosial, lalu lintas langsung, kampanye berbayar, dan lainnya) yang mengarahkan pelanggan ke situs web.

Topik

- Mengekspor data tanpa partisi
- Mempartisi data berdasarkan saluran
- Mempartisi data berdasarkan acara
- Mempartisi data berdasarkan saluran dan acara
- File manifes dan metadata
- Menggunakan Glue crawler untuk membangun Glue Data Catalog

Mengekspor data tanpa partisi

Anda ingin mengekspor dua hari terakhir data Anda dalam CSV format.

```
UNLOAD(SELECT user_id, ip_address, event, session_id, measure_name, time,
query, quantity, product_id, channel
FROM sample_clickstream.sample_shopping
WHERE time BETWEEN ago(2d) AND now())
TO 's3://<bucket_name>/withoutpartition'
WITH (  format='CSV',
compression='GZIP')
```

Mempartisi data berdasarkan saluran

Anda ingin mengekspor dua hari terakhir data dalam CSV format tetapi ingin memiliki data dari setiap saluran lalu lintas di folder terpisah. Untuk melakukan ini, Anda perlu mempartisi data menggunakan `channel` kolom seperti yang ditunjukkan pada berikut ini.

```
UNLOAD(SELECT user_id, ip_address, event, session_id, measure_name, time,
query, quantity, product_id, channel
FROM sample_clickstream.sample_shopping
WHERE time BETWEEN ago(2d) AND now())
TO 's3://<bucket_name>/partitionbychannel/'
WITH (
partitioned_by = ARRAY ['channel'],
format='CSV',
compression='GZIP')
```

Mempartisi data berdasarkan acara

Anda ingin mengekspor dua hari terakhir data dalam CSV format tetapi ingin memiliki data untuk setiap acara dalam folder terpisah. Untuk melakukan ini, Anda perlu mempartisi data menggunakan `event` kolom seperti yang ditunjukkan pada berikut ini.

```
UNLOAD(SELECT user_id, ip_address, channel, session_id, measure_name, time,
query, quantity, product_id, event
FROM sample_clickstream.sample_shopping
WHERE time BETWEEN ago(2d) AND now())
TO 's3://<bucket_name>/partitionbyevent/'
WITH (
partitioned_by = ARRAY ['event'],
format='CSV',
compression='GZIP')
```

Mempartisi data berdasarkan saluran dan acara

Anda ingin mengekspor dua hari terakhir data dalam CSV format tetapi ingin memiliki data untuk setiap saluran dan di dalam saluran menyimpan setiap acara dalam folder terpisah. Untuk melakukan ini, Anda perlu mempartisi data menggunakan keduanya `channel` dan `event` kolom seperti yang ditunjukkan pada berikut ini.

```
UNLOAD(SELECT user_id, ip_address, session_id, measure_name, time,
query, quantity, product_id, channel,event
FROM sample_clickstream.sample_shopping
WHERE time BETWEEN ago(2d) AND now())
TO 's3://<bucket_name>/partitionbychannelevent/'
WITH (
partitioned_by = ARRAY ['channel','event'],
format='CSV',
compression='GZIP')
```

File manifes dan metadata

File manifes

File manifes memberikan informasi tentang daftar file yang diekspor dengan UNLOAD eksekusi. File manifes tersedia di bucket S3 yang disediakan dengan nama file:`S3://bucket_name/<queryid>_<UUID>_manifest.json`. File manifes akan berisi url file di folder hasil, jumlah

catatan dan ukuran file masing-masing, dan metadata kueri (yang merupakan total byte dan total baris yang diekspor ke S3 untuk kueri).


```
{
  "result_files": [
    {
        "url":"s3://my_timestream_unloads/ec2_metrics/
AEDAGANLHLBH4OLISD3CVOZZRWPX5GV2XCXRBKCVD554N6GWPWWXBP7LSG74V2Q_1448466917_szCL4YgVYzGXj2lS.gz"
        "file_metadata":
            {
                "content_length_in_bytes": 32295,
                "row_count": 10
            }
    },
    {
        "url":"s3://my_timestream_unloads/ec2_metrics/
AEDAGANLHLBH4OLISD3CVOZZRWPX5GV2XCXRBKCVD554N6GWPWWXBP7LSG74V2Q_1448466917_szCL4YgVYzGXj2lS.gz"
        "file_metadata":
            {
                "content_length_in_bytes": 62295,
                "row_count": 20
            }
    },
  ],
  "query_metadata":
    {
      "content_length_in_bytes": 94590,
      "total_row_count": 30,
      "result_format": "CSV",
      "result_version": "Amazon Timestream version 1.0.0"
    },
  "author": {
        "name": "Amazon Timestream",
        "manifest_file_version": "1.0"
  }
}
```


Metadata

File metadata memberikan informasi tambahan tentang kumpulan data seperti nama kolom, jenis kolom, dan skema. <queryid>File metadata tersedia di bucket S3 yang disediakan dengan nama file: S3: //bucket_name/ _< >_metadata.json UUID

Berikut ini adalah contoh dari file metadata.

```
{
    "ColumnInfo": [
        {
            "Name": "hostname",
            "Type": {
                "ScalarType": "VARCHAR"
            }
        },
        {
            "Name": "region",
            "Type": {
                "ScalarType": "VARCHAR"
            }
        },
        {
            "Name": "measure_name",
            "Type": {
                "ScalarType": "VARCHAR"
            }
        },
        {
            "Name": "cpu_utilization",
            "Type": {
                "TimeSeriesMeasureValueColumnInfo": {
                    "Type": {
                        "ScalarType": "DOUBLE"
                    }
                }
            }
        }
  ],
  "Author": {
        "Name": "Amazon Timestream",
        "MetadataFileVersion": "1.0"
  }
}
```

Informasi kolom yang dibagikan dalam file metadata memiliki struktur yang sama seperti yang `ColumnInfo` dikirim dalam API respons Kueri untuk `SELECT` kueri.

Menggunakan Glue crawler untuk membangun Glue Data Catalog

1. Login ke akun Anda dengan kredensi Admin untuk validasi berikut.
2. Buat Crawler untuk Glue Database menggunakan pedoman yang disediakan di sini. Harap dicatat bahwa folder S3 yang akan disediakan di sumber data harus berupa folder UNLOAD hasil seperti. `s3://my_timestream_unloads/results`
3. Jalankan crawler mengikuti panduan di sini.
4. Lihat tabel Glue.
   - Pergi ke AWS Glue → Tabel.
   - Anda akan melihat tabel baru yang dibuat dengan awalan tabel yang disediakan saat membuat crawler.
   - Anda dapat melihat skema dan informasi partisi dengan mengklik tampilan detail tabel.

Berikut ini adalah AWS layanan lain dan proyek sumber terbuka yang menggunakan Katalog Data AWS Glue.

- Amazon Athena — Untuk informasi selengkapnya, lihat Memahami tabel, database, dan katalog data di Panduan Pengguna Amazon Athena.
- Amazon Redshift Spectrum — Untuk informasi selengkapnya, lihat Mengkueri data eksternal menggunakan Amazon Redshift Spectrum di Panduan Pengembang Database Amazon Redshift.
- Amazon EMR - Untuk informasi selengkapnya, lihat Menggunakan kebijakan berbasis sumber daya untuk akses Amazon EMR ke Katalog Data AWS Glue di Panduan Manajemen Amazon. EMR
- AWS Klien Glue Data Catalog untuk Apache Hive metastore — Untuk informasi lebih lanjut tentang proyek ini GitHub, lihat Klien Katalog Data AWS Glue untuk Apache Hive Metastore.

Batas untuk UNLOAD dari Timestream untuk LiveAnalytics

Berikut ini adalah batasan yang terkait dengan UNLOAD perintah.

- Konkurensi untuk kueri menggunakan UNLOAD pernyataan adalah 1 kueri per detik ()QPS. Melebihi tingkat kueri dapat mengakibatkan pelambatan.
- Kueri yang berisi UNLOAD pernyataan dapat mengekspor paling banyak 100 partisi per kueri. Kami merekomendasikan untuk memeriksa jumlah yang berbeda dari kolom yang dipilih sebelum menggunakannya untuk mempartisi data yang diekspor.

- Kueri yang berisi waktu keluar `UNLOAD` pernyataan setelah 60 menit.
- Ukuran maksimum file yang dibuat `UNLOAD` pernyataan di Amazon S3 adalah 78 GB.

Untuk batasan lain untuk Timestream LiveAnalytics, lihat Kuota

Menggunakan wawasan kueri untuk mengoptimalkan kueri di Amazon Timestream

Wawasan kueri adalah fitur penyetelan kinerja yang membantu Anda mengoptimalkan kueri, meningkatkan kinerjanya, dan mengurangi biaya. Dengan wawasan kueri, Anda dapat menilai efisiensi pemangkasan berbasis kunci partisi temporal, berbasis waktu, dan spasial dari kueri Anda. Menggunakan wawasan kueri, Anda juga dapat mengidentifikasi area untuk perbaikan guna meningkatkan kinerja kueri. Selain itu, dengan wawasan kueri, Anda dapat mengevaluasi seberapa efektif kueri Anda menggunakan pengindeksan berbasis kunci berbasis waktu dan partisi untuk mengoptimalkan pengambilan data. Untuk mengoptimalkan kinerja kueri, penting untuk menyempurnakan parameter temporal dan spasial yang mengatur eksekusi kueri.

Topik

- Manfaat wawasan kueri
- Mengoptimalkan akses data di Amazon Timestream
- Mengaktifkan wawasan kueri di Amazon Timestream
- Mengoptimalkan kueri menggunakan respons wawasan kueri

Manfaat wawasan kueri

Berikut ini adalah manfaat utama menggunakan wawasan kueri:

- Mengidentifikasi kueri yang tidak efisien — Wawasan kueri memberikan informasi tentang pemangkasan tabel yang diakses oleh kueri berbasis waktu dan berbasis atribut. Informasi ini membantu Anda mengidentifikasi tabel yang diakses secara sub-optimal.
- Mengoptimalkan model data dan partisi — Anda dapat menggunakan informasi wawasan kueri untuk mengakses dan menyempurnakan model data dan strategi partisi Anda.
- Kueri penyetelan — Wawasan kueri menyoroti peluang untuk menggunakan indeks secara lebih efektif.

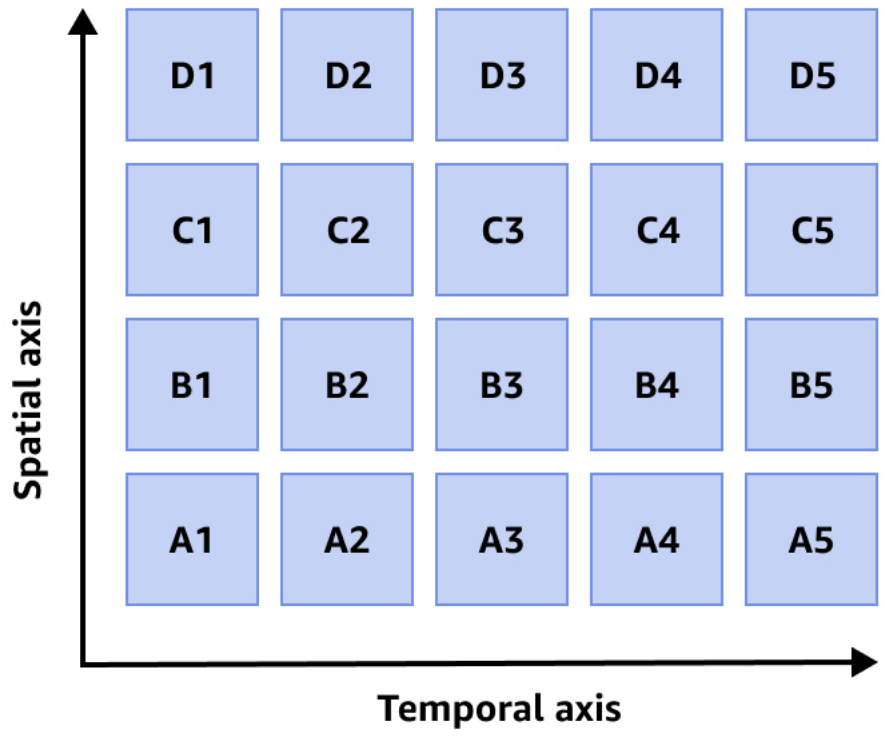
Mengoptimalkan akses data di Amazon Timestream

Anda dapat mengoptimalkan pola akses data di Amazon Timestream menggunakan skema partisi Timestream atau teknik organisasi data.

Topik

Skema partisi Timestream

Amazon Timestream menggunakan skema partisi yang sangat skalabel di mana setiap tabel Timestream dapat memiliki ratusan, ribuan, atau bahkan jutaan partisi independen. Layanan pelacakan dan pengindeksan partisi yang sangat tersedia mengelola partisi, meminimalkan dampak kegagalan dan membuat sistem lebih tangguh.



Organisasi data

Timestream menyimpan setiap titik data yang dicerna dalam satu partisi. Saat Anda menyerap data ke dalam tabel Timestream, Timestream secara otomatis membuat partisi berdasarkan stempel waktu, kunci partisi, dan atribut konteks lainnya dalam data. Selain mempartisi data tepat waktu (partisi temporal), Timestream juga mempartisi data berdasarkan kunci partisi yang dipilih dan dimensi lainnya (partisi spasial). Pendekatan ini dirancang untuk mendistribusikan lalu lintas tulis dan memungkinkan pemangkasan data yang efektif untuk kueri.

Fitur wawasan kueri memberikan wawasan berharga tentang efisiensi pemangkasan kueri, yang mencakup cakupan spasial kueri dan cakupan temporal kueri.

Topik

- QuerySpatialCoverage
- QueryTemporalCoverage

QuerySpatialCoverage

QuerySpatialCoverageMetrik memberikan wawasan tentang cakupan spasial dari kueri yang dieksekusi dan tabel dengan pemangkasan spasial yang paling tidak efisien. Informasi ini dapat membantu Anda mengidentifikasi area perbaikan dalam strategi partisi untuk meningkatkan pemangkasan spasial. Nilai untuk rentang `QuerySpatialCoverage` metrik antara 0 dan 1. Semakin rendah nilai metrik, semakin optimal pemangkasan kueri pada sumbu spasial. Misalnya, nilai 0,1 menunjukkan bahwa kueri memindai 10% sumbu spasial. Nilai 1 menunjukkan bahwa kueri memindai 100% sumbu spasial.

Example Menggunakan wawasan kueri untuk menganalisis cakupan spasial kueri

Katakanlah Anda memiliki database Timestream yang menyimpan data cuaca. Asumsikan bahwa suhu dicatat setiap jam dari stasiun cuaca yang terletak di berbagai negara bagian di Amerika Serikat. Bayangkan Anda memilih `State` sebagai kunci partisi yang ditentukan pelanggan (CDPK) untuk mempartisi data berdasarkan status.

Misalkan Anda menjalankan kueri untuk mengambil suhu rata-rata untuk semua stasiun cuaca di California antara jam 2 siang dan 4 sore pada hari tertentu. Contoh berikut menunjukkan query untuk skenario ini.

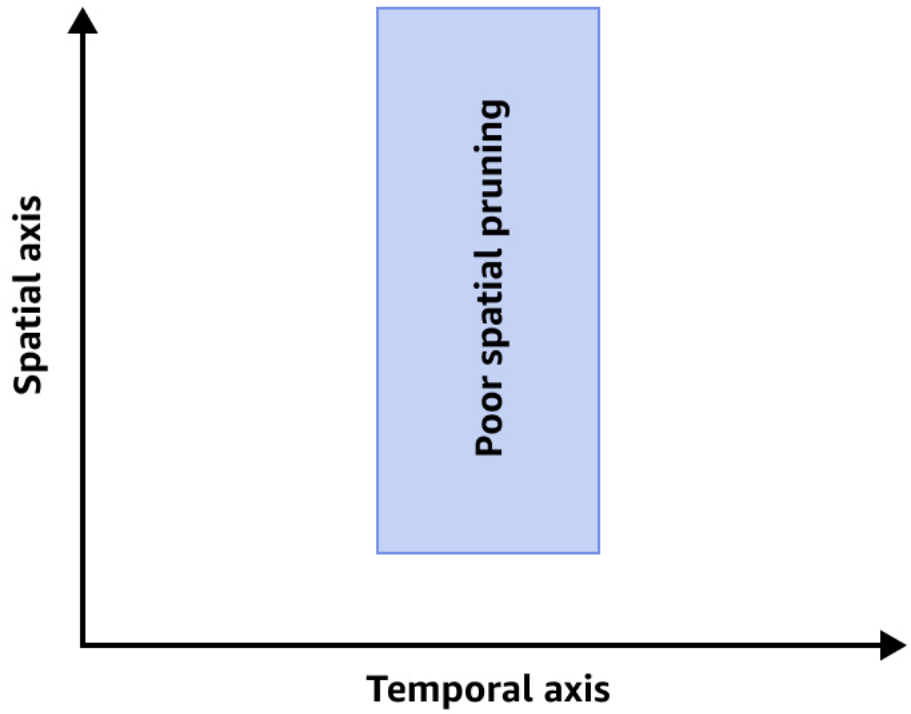
```
SELECT AVG(temperature)
```

```
FROM "weather_data"."hourly_weather"
WHERE time >= '2024-10-01 14:00:00' AND time < '2024-10-01 16:00:00'
  AND state = 'CA';
```

Dengan menggunakan fitur wawasan kueri, Anda dapat menganalisis cakupan spasial kueri. Bayangkan bahwa `QuerySpatialCoverage` metrik mengembalikan nilai 0,02. Ini berarti bahwa kueri hanya memindai 2% sumbu spasial, yang efisien. Dalam hal ini, kueri dapat secara efektif memangkas rentang spasial, hanya mengambil data dari California dan mengabaikan data dari negara bagian lain.

Sebaliknya, jika `QuerySpatialCoverage` metrik mengembalikan nilai 0,8, itu akan menunjukkan bahwa kueri memindai 80% sumbu spasial, yang kurang efisien. Ini mungkin menunjukkan bahwa strategi partisi perlu disempurnakan untuk meningkatkan pemangkasan spasial. Misalnya, Anda dapat memilih kunci partisi sebagai kota atau wilayah, bukan negara bagian. Dengan menganalisis `QuerySpatialCoverage` metrik, Anda dapat mengidentifikasi peluang untuk mengoptimalkan strategi partisi Anda dan meningkatkan kinerja kueri Anda.

Gambar berikut menunjukkan pemangkasan spasial yang buruk.



Untuk meningkatkan efisiensi pemangkasan spasial, Anda dapat melakukan salah satu atau kedua hal berikut:

- Tambahkan`measure_name`, kunci paritioning default, atau gunakan CDPK predikat dalam kueri Anda.
- Jika Anda telah menambahkan atribut yang disebutkan di poin sebelumnya, hapus fungsi di sekitar atribut atau klausa ini, seperti. `LIKE`

QueryTemporalCoverage

`QueryTemporalCoverage`Metrik memberikan wawasan tentang rentang temporal yang dipindai oleh kueri yang dieksekusi, termasuk tabel dengan rentang waktu terbesar yang dipindai. Nilai untuk `QueryTemporalCoverage` metrik adalah rentang waktu yang diwakili dalam nanodetik. Semakin rendah nilai metrik ini, semakin optimal pemangkasan kueri pada rentang temporal. Misalnya, pemindaian kueri beberapa menit terakhir data lebih berkinerja daripada kueri yang memindai seluruh rentang waktu tabel.

Example

Katakanlah Anda memiliki database Timestream yang menyimpan data sensor IoT, dengan pengukuran yang dilakukan setiap menit dari perangkat yang berlokasi di pabrik. Asumsikan bahwa Anda telah mempartisi data Anda dengan. `device_ID`

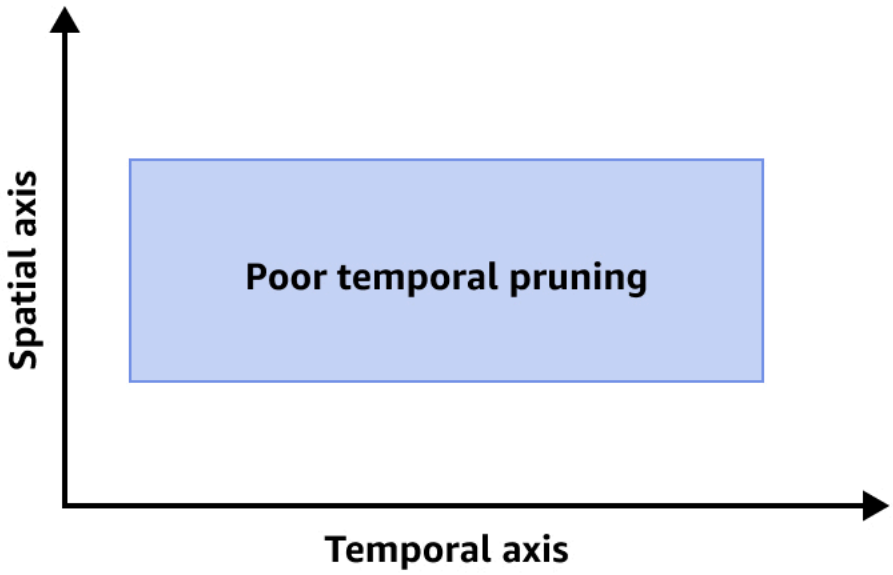
Misalkan Anda menjalankan kueri untuk mengambil pembacaan sensor rata-rata untuk perangkat tertentu selama 30 menit terakhir. Contoh berikut menunjukkan query untuk skenario ini.

```
SELECT AVG(sensor_reading)
FROM "sensor_data"."factory_1"
WHERE device_id = 'DEV_123'
  AND time >= NOW() - INTERVAL 30 MINUTE and time < NOW();
```

Dengan menggunakan fitur wawasan kueri, Anda dapat menganalisis rentang temporal yang dipindai oleh kueri. Bayangkan `QueryTemporalCoverage` metrik mengembalikan nilai 1800000000000 nanodetik (30 menit). Ini berarti bahwa kueri hanya memindai 30 menit terakhir data, yang merupakan rentang temporal yang relatif sempit. Ini adalah pertanda baik karena ini menunjukkan bahwa kueri dapat secara efektif memangkas partisi temporal dan hanya mengambil data yang diminta.

Sebaliknya, jika `QueryTemporalCoverage` metrik mengembalikan nilai 1 tahun dalam nanodetik, ini menunjukkan bahwa kueri memindai rentang waktu satu tahun dalam tabel, yang kurang efisien. Ini mungkin menunjukkan bahwa kueri tidak dioptimalkan untuk pemangkasan temporal, dan Anda dapat memperbaikinya dengan menambahkan filter waktu.

Gambar berikut menunjukkan pemangkasan temporal yang buruk.



Untuk meningkatkan pemangkasan temporal, kami sarankan Anda melakukan satu atau semua hal berikut:

- Tambahkan predikat waktu yang hilang dalam kueri dan pastikan predikat waktu memangkas jendela waktu yang diinginkan.
- Hapus fungsi, seperti`MAX()`, sekitar predikat waktu.
- Tambahkan predikat waktu ke semua sub kueri. Ini penting jika sub kueri Anda bergabung dengan tabel besar atau melakukan operasi yang kompleks.

Mengaktifkan wawasan kueri di Amazon Timestream

Anda dapat mengaktifkan wawasan kueri untuk kueri Anda dengan wawasan yang dikirimkan langsung melalui respons kueri. Mengaktifkan wawasan kueri tidak memerlukan infrastruktur tambahan atau menimbulkan biaya tambahan. Saat Anda mengaktifkan wawasan kueri, ia menampilkan bidang metadata terkait kinerja kueri selain hasil kueri sebagai bagian dari respons

kueri Anda. Anda dapat menggunakan informasi ini untuk menyetel kueri untuk meningkatkan kinerja kueri dan mengurangi biaya kueri.

Untuk informasi tentang mengaktifkan wawasan kueri, lihat. Jalankankueri

Untuk melihat contoh tanggapan yang ditampilkan dengan mengaktifkan wawasan kueri, lihat Contoh untuk kueri terjadwal.

> **Note**
>
> - Saat Anda mengaktifkan wawasan kueri, nilainya membatasi kueri menjadi 1 kueri per detik (QPS). Untuk menghindari dampak kinerja, kami sangat menyarankan agar Anda mengaktifkan wawasan kueri hanya selama fase evaluasi kueri Anda, sebelum menerapkannya ke produksi.
> - Wawasan yang diberikan dalam wawasan kueri pada akhirnya konsisten, yang berarti mereka mungkin berubah karena data baru terus dicerna ke dalam tabel.

Mengoptimalkan kueri menggunakan respons wawasan kueri

Katakanlah Anda menggunakan Amazon Timestream LiveAnalytics untuk memantau konsumsi energi di berbagai lokasi. Bayangkan bahwa Anda memiliki dua tabel dalam database Anda bernama `raw-metrics` dan`aggregate-metrics`.

`raw-metrics`Tabel menyimpan data energi terperinci di tingkat perangkat dan berisi kolom berikut:

- Stempel Waktu
- Negara, misalnya, Washington
- ID Perangkat
- Konsumsi energi

Data untuk tabel ini dikumpulkan dan disimpan pada minute-by-minute perincian. Tabel menggunakan `State` sebagaiCDPK.

`aggregate-metrics`Tabel menyimpan hasil kueri terjadwal untuk mengumpulkan data konsumsi energi di semua perangkat setiap jam. Tabel ini berisi kolom berikut:

- Stempel Waktu

- Negara, misalnya, Washington
- Konsumsi energi total

`aggregate-metrics`Tabel menyimpan data ini pada perincian per jam. Tabel menggunakan `State` sebagaiCDPK.

Topik

- Meminta konsumsi energi selama 24 jam terakhir
- Mengoptimalkan kueri untuk rentang temporal
- Mengoptimalkan kueri untuk cakupan spasial
- Peningkatan kinerja kueri

Meminta konsumsi energi selama 24 jam terakhir

Katakanlah Anda ingin mengekstrak total energi yang dikonsumsi di Washington selama 24 jam terakhir. Untuk menemukan data ini, Anda dapat memanfaatkan kekuatan kedua tabel: `raw-metrics` dan`aggregate-metrics`. `aggregate-metrics`Tabel menyediakan data konsumsi energi per jam selama 23 jam terakhir, sedangkan `raw-metrics` tabel menawarkan data menit-granular selama satu jam terakhir. Dengan menanyakan di kedua tabel, Anda bisa mendapatkan gambaran konsumsi energi yang lengkap dan akurat di Washington selama 24 jam terakhir.

```
SELECT am.time, am.state, am.total_energy_consumption,
rm.time, rm.state, rm.device_id, rm.energy_consumption
FROM
 "metrics"."aggregate-metrics" am
 LEFT JOIN "metrics"."raw-metrics" rm ON am.state = rm.state
WHERE rm.time >= ago(1h) and rm.time < now()
```

Contoh kueri ini disediakan untuk tujuan ilustrasi saja dan mungkin tidak berfungsi apa adanya. Ini dimaksudkan untuk mendemonstrasikan konsep, tetapi Anda mungkin perlu memodifikasinya agar sesuai dengan kasus penggunaan atau lingkungan khusus Anda.

Setelah menjalankan kueri ini, Anda mungkin memperhatikan bahwa waktu respons kueri lebih lambat dari yang diharapkan. Untuk mengidentifikasi akar penyebab masalah kinerja ini, Anda dapat menggunakan fitur wawasan kueri untuk menganalisis kinerja kueri dan mengoptimalkan pelaksanaannya.

Contoh berikut menunjukkan respons wawasan kueri.

```
queryInsightsResponse={
                QuerySpatialCoverage: {
                    Max: {
                        Value: 1.0,
                        TableArn: arn:aws:timestream:us-east-1:123456789012:database/
metrics/table/raw-metrics,
                        PartitionKey: [State]
                    }
                },
                QueryTemporalRange: {
                    Max: {
                        Value:31540000000000000 //365 days,
                        TableArn: arn:aws:timestream:us-east-1:123456789012:database/
metrics/table/aggregate-metrics
                    }
                },
                QueryTableCount: 2,
                OutputRows: 83,
                OutputBytes: 590
```

Respons wawasan kueri memberikan informasi berikut:

- Rentang temporal: Kueri memindai rentang temporal 365 hari yang berlebihan untuk tabel. `aggregate-metrics` Ini menunjukkan penggunaan penyaringan temporal yang tidak efisien.
- Cakupan spasial: Kueri memindai seluruh rentang spasial (100%) `raw-metrics` tabel. Ini menunjukkan bahwa penyaringan spasial tidak digunakan secara efektif.

Jika kueri Anda mengakses lebih dari satu tabel, wawasan kueri menyediakan metrik untuk tabel dengan pola akses paling sub-optimal.

Mengoptimalkan kueri untuk rentang temporal

Berdasarkan respons wawasan kueri, Anda dapat mengoptimalkan kueri untuk rentang temporal seperti yang ditunjukkan pada contoh berikut.

```
SELECT am.time, am.state, am.total_energy_consumption,
rm.time, rm.state, rm.device_id, rm.energy_consumption
FROM
  "metrics"."aggregate-metrics" am
```

```
  LEFT JOIN "metrics"."raw-metrics" rm ON am.state = rm.state
WHERE
  am.time >=  ago(23h) and am.time < now()
  AND rm.time >=  ago(1h) and rm.time < now()
  AND rm.state = 'Washington'
```

Jika Anda menjalankan `QueryInsights` perintah lagi, ia mengembalikan respons berikut.

```
queryInsightsResponse={
                QuerySpatialCoverage: {
                    Max: {
                        Value: 1.0,
                        TableArn: arn:aws:timestream:us-east-1:123456789012:database/
metrics/table/aggregate-metrics,
                        PartitionKey: [State]
                    }
                },
                QueryTemporalRange: {
                    Max: {
                        Value: 82800000000000 //23 hours,
                        TableArn: arn:aws:timestream:us-east-1:123456789012:database/
metrics/table/aggregate-metrics
                    }
                },
                QueryTableCount: 2,
                OutputRows: 83,
                OutputBytes: 590
```

Tanggapan ini menunjukkan bahwa cakupan spasial untuk `aggregate-metrics` tabel masih 100%, yang tidak efisien. Bagian berikut menunjukkan cara mengoptimalkan kueri untuk cakupan spasial.

Mengoptimalkan kueri untuk cakupan spasial

Berdasarkan respons wawasan kueri, Anda dapat mengoptimalkan kueri untuk cakupan spasial seperti yang ditunjukkan pada contoh berikut.

```
SELECT am.time, am.state, am.total_energy_consumption,
rm.time, rm.state, rm.device_id, rm.energy_consumption
FROM
  "metrics"."aggregate-metrics" am
  LEFT JOIN "metrics"."raw-metrics" rm ON am.state = rm.state
WHERE
```

```
  am.time >=  ago(23h) and am.time < now()
  AND am.state ='Washington'
  AND rm.time >=  ago(1h) and rm.time < now()
  AND rm.state = 'Washington'
```

Jika Anda menjalankan `QueryInsights` perintah lagi, ia mengembalikan respons berikut.

```
queryInsightsResponse={
                QuerySpatialCoverage: {
                    Max: {
                        Value: 0.02,
                        TableArn: arn:aws:timestream:us-east-1:123456789012:database/
metrics/table/aggregate-metrics,
                        PartitionKey: [State]
                    }
                },
                QueryTemporalRange: {
                    Max: {
                        Value: 82800000000000 //23 hours,
                        TableArn: arn:aws:timestream:us-east-1:123456789012:database/
metrics/table/aggregate-metrics
                    }
                },
                QueryTableCount: 2,
                OutputRows: 83,
                OutputBytes: 590
```

Peningkatan kinerja kueri

Setelah mengoptimalkan kueri, wawasan kueri memberikan informasi berikut:

- Pemangkasan temporal untuk `aggregate-metrics` meja adalah 23 jam. Ini menunjukkan bahwa hanya 23 jam dari rentang temporal yang dipindai.
- Pemangkasan spasial untuk `aggregate-metrics` tabel adalah 0,02. Ini menunjukkan bahwa hanya 2% dari data rentang spasial tabel yang dipindai. Kueri memindai sebagian kecil tabel yang mengarah ke kinerja yang cepat dan pemanfaatan sumber daya yang berkurang. Efisiensi pemangkasan yang ditingkatkan menunjukkan bahwa kueri sekarang dioptimalkan untuk kinerja.

Bekerja dengan AWS Backup

Fungsionalitas perlindungan data di Amazon Timestream for LiveAnalytics adalah solusi yang dikelola sepenuhnya untuk membantu Anda memenuhi kepatuhan peraturan dan persyaratan kelangsungan bisnis Anda. Fungsionalitas ini diaktifkan melalui integrasi asli dengan AWS Backup, layanan pencadangan terpadu yang dirancang untuk menyederhanakan pembuatan, migrasi, pemulihan, dan penghapusan cadangan, sambil memberikan pelaporan dan audit yang lebih baik. Melalui integrasi dengan AWS Backup, Anda dapat menggunakan solusi perlindungan data terpusat yang dikelola sepenuhnya dan didorong oleh kebijakan untuk membuat cadangan yang tidak dapat diubah dan mengelola perlindungan data secara terpusat dari data aplikasi Anda yang mencakup Timestream dan layanan lain yang didukung oleh. AWS AWS Backup

Untuk menggunakan fungsionalitas, Anda harus ikut serta untuk memungkinkan AWS Backup untuk melindungi sumber daya Timestream Anda. Pilihan keikutsertaan berlaku untuk akun dan AWS Wilayah tertentu, jadi Anda mungkin harus memilih beberapa Wilayah menggunakan akun yang sama. Untuk informasi selengkapnya tentang AWS Backup, lihat Panduan AWS Backup Pengembang.

Fungsionalitas Perlindungan Data yang tersedia AWS Backup termasuk yang berikut ini.

Pencadangan terjadwal —Anda dapat mengatur cadangan Timestream yang dijadwalkan secara teratur untuk tabel menggunakan paket cadangan. LiveAnalytics

Penyalinan lintas akun dan lintas wilayah —Anda dapat secara otomatis menyalin cadangan ke brankas cadangan lain di AWS Wilayah atau akun lain, yang memungkinkan Anda untuk mendukung persyaratan perlindungan data Anda.

Tingkat penyimpanan dingin —Anda dapat mengonfigurasi cadangan Anda untuk menerapkan aturan siklus hidup untuk menghapus atau mentransisikan cadangan ke penyimpanan yang lebih dingin. Hal ini dapat membantu Anda mengoptimalkan biaya cadangan Anda.

Tag —Anda dapat secara otomatis menandai cadangan Anda untuk tujuan penagihan dan alokasi biaya.

Enkripsi —Data cadangan Anda disimpan di AWS Backup brankas. Ini memungkinkan Anda untuk mengenkripsi dan mengamankan cadangan Anda dengan menggunakan AWS KMS kunci yang independen dari Timestream Anda untuk LiveAnalytics kunci enkripsi tabel.

Cadangan aman menggunakan WORM model —Anda dapat menggunakan AWS Backup Vault Lock untuk mengaktifkan pengaturan write-once-read-many (WORM) untuk cadangan Anda. Dengan

AWS Backup Vault Lock, Anda dapat menambahkan lapisan pertahanan tambahan yang melindungi cadangan dari operasi penghapusan yang tidak disengaja atau berbahaya, perubahan periode retensi cadangan, dan pembaruan ke pengaturan siklus hidup. Untuk mempelajari lebih lanjut, lihat AWS Backup Vault Lock.

Fungsionalitas perlindungan data tersedia di semua wilayah Untuk mempelajari fungsionalitas lebih lanjut, lihat Panduan AWS Backup Pengembang.

Mencadangkan dan memulihkan tabel Timestream: Cara kerjanya

Anda dapat membuat cadangan tabel Amazon Timestream Anda. Bagian ini memberikan gambaran umum tentang apa yang terjadi selama proses pencadangan dan pemulihan.

Topik

- Pencadangan
- Memulihkan

Pencadangan

Anda dapat menggunakan fitur pencadangan sesuai permintaan untuk membuat cadangan lengkap Amazon Timestream Anda untuk tabel. LiveAnalytics Bagian ini memberikan gambaran umum tentang apa yang terjadi selama proses pencadangan dan pemulihan.

Anda dapat membuat cadangan data Timestream Anda pada perincian tabel. Anda dapat memulai pencadangan tabel yang dipilih menggunakan konsol Timestream, atau AWS Backup konsol,SDK, atau. CLI Cadangan dibuat secara asinkron dan semua data dalam tabel hingga waktu inisiasi cadangan disertakan dalam cadangan. Namun, ada kemungkinan bahwa beberapa data yang tertelan ke dalam tabel saat cadangan sedang berlangsung mungkin juga disertakan dalam cadangan. Untuk melindungi data Anda, Anda dapat membuat cadangan sesuai permintaan satu kali atau menjadwalkan pencadangan berulang tabel Anda.

Saat cadangan sedang berlangsung, Anda tidak dapat melakukan hal berikut.

- Menjeda atau membatalkan operasi pencadangan.
- Menghapus tabel sumber cadangan.
- Nonaktifkan pencadangan pada tabel jika pencadangan untuk tabel tersebut sedang berlangsung.

Setelah dikonfigurasi, AWS Backup menyediakan jadwal pencadangan otomatis, manajemen retensi, dan manajemen siklus hidup, menghilangkan kebutuhan akan skrip khusus dan proses manual. Untuk informasi selengkapnya, lihat Panduan AWS Backup Pengembang

Semua Timestream untuk LiveAnalytics cadangan bersifat inkremental, menyiratkan bahwa cadangan pertama tabel adalah cadangan penuh dan setiap cadangan berikutnya dari tabel yang sama adalah cadangan tambahan, hanya menyalin perubahan ke data sejak cadangan terakhir. Karena data di Timestream for LiveAnalytics disimpan dalam kumpulan partisi, semua partisi yang berubah baik karena menelan data baru atau pembaruan ke data yang ada sejak cadangan terakhir disalin selama pencadangan berikutnya.

Jika Anda menggunakan Timestream untuk LiveAnalytics konsol, cadangan yang dibuat untuk semua sumber daya di akun tercantum di tab Cadangan. Selain itu, cadangan juga tercantum dalam rincian Tabel.

Memulihkan

Anda dapat memulihkan tabel dari Timestream untuk LiveAnalytics konsol, atau AWS Backup konsol,SDK, atau AWS CLI. Anda dapat memulihkan seluruh data dari cadangan Anda, atau mengonfigurasi pengaturan retensi tabel untuk memulihkan data tertentu. Ketika Anda memulai pemulihan, Anda dapat mengkonfigurasi pengaturan tabel berikut.

- Nama Basis Data
- Nama Tabel
- Retensi penyimpanan memori
- Retensi toko magnetik
- Aktifkan penulisan penyimpanan Magnetik
- Lokasi log kesalahan S3 (opsional)
- IAMperan yang AWS Backup akan diasumsikan saat memulihkan cadangan

Konfigurasi sebelumnya tidak tergantung pada tabel sumber. Untuk memulihkan semua data dalam cadangan Anda, kami sarankan Anda mengonfigurasi pengaturan tabel baru sehingga jumlah periode penyimpanan penyimpanan memori dan periode penyimpanan penyimpanan magnetik lebih besar daripada perbedaan antara stempel waktu tertua dan sekarang. Saat Anda memilih cadangan yang bersifat inkremental untuk dipulihkan, semua data (inkremental+data lengkap yang mendasari) dipulihkan. Setelah pemulihan berhasil, tabel dalam keadaan aktif dan Anda dapat melakukan operasi konsumsi dan/atau kueri pada tabel yang dipulihkan. Namun, Anda tidak dapat melakukan

operasi ini saat pemulihan sedang berlangsung. Setelah dipulihkan, tabelnya mirip dengan tabel lain di akun Anda.

Example Kembalikan semua data dari cadangan

Contoh ini memiliki asumsi berikut.

Stempel waktu tertua — `August 1, 2021 0:00:00`

- Sekarang - `November 9, 2022 0:00:00`

Untuk mengembalikan semua data dari cadangan, masukkan dan bandingkan nilai sebagai berikut.

1. Masukkan Retensi penyimpanan memori dan retensi penyimpanan Magnetik. Misalnya, asumsikan nilai-nilai ini.
   - Retensi penyimpanan memori —12 jam
   - Retensi toko magnetik —500 hari
2. Temukan jumlah retensi penyimpanan Memori dan retensi penyimpanan Magnetik.

```
12 hours + (500 * 24 hours) =
12 hours + 12,000 hours =
12,012 hours
```

3. Temukan perbedaan antara stempel waktu tertua dan sekarang.

```
November 9, 2022 0:00:00 - August 1, 2021 0:00:00 =
465 days =
465 * 24 hours =
11,160 hours
```

4. Pastikan jumlah nilai retensi pada langkah kedua lebih besar dari perbedaan waktu pada langkah ketiga. Sesuaikan waktu retensi jika perlu.

```
12,012 > 11,160
true
```

Example Pulihkan data tertentu dari cadangan

Contoh ini memiliki asumsi sebagai berikut.

- Sekarang - `November 9, 2022 0:00:00`

Untuk mengembalikan hanya pilih data dari cadangan, masukkan dan bandingkan nilai sebagai berikut.

1. Tentukan stempel waktu paling awal yang diperlukan. Misalnya, asumsikan`December 4, 2021 0:00:00`.
2. Temukan perbedaan antara stempel waktu paling awal yang diperlukan dan sekarang.

```
November 9, 2022 0:00:00 - December 4, 2021 0:00:00 =
340 days =
340 * 24 hours =
8,160 hours
```

3. Masukkan nilai yang diinginkan untuk retensi penyimpanan memori. Misalnya, masukkan 12 jam.
4. Kurangi nilai dari perbedaan pada langkah kedua.

```
8,160 hours - 12 hours =
8148 hours
```

5. Masukkan nilai itu untuk retensi penyimpanan Magnetik.

Anda dapat menyalin cadangan Timestream Anda untuk data LiveAnalytics tabel ke AWS Wilayah lain dan kemudian mengembalikannya di Wilayah baru tersebut. Anda dapat menyalin dan kemudian memulihkan cadangan antara Wilayah AWS komersial, dan Wilayah AWS GovCloud (AS). Anda hanya membayar data yang Anda salin dari Wilayah sumber dan data yang Anda pulihkan ke tabel baru di Wilayah tujuan.

Setelah tabel dipulihkan, Anda harus mengatur yang berikut secara manual pada tabel yang dipulihkan.

- AWS Kebijakan Identity and Access Management (IAM)
- Tanda
- Pertanyaan Terjadwal

Waktu pemulihan terkait langsung dengan konfigurasi tabel Anda. Ini termasuk ukuran tabel Anda, jumlah partisi yang mendasarinya, jumlah data yang dikembalikan ke penyimpanan memori, dan

variabel lainnya. Praktik terbaik saat merencanakan pemulihan bencana adalah dengan secara teratur mendokumentasikan waktu penyelesaian pemulihan rata-rata dan menetapkan bagaimana waktu-waktu ini memengaruhi Tujuan Waktu Pemulihan Anda secara keseluruhan (RTO).

Semua konsol dan API tindakan pencadangan dan pemulihan ditangkap dan direkam AWS CloudTrail untuk pencatatan, pemantauan berkelanjutan, dan audit.

Membuat cadangan tabel Amazon Timestream

Bagian ini menjelaskan cara mengaktifkan AWS Backup dan membuat cadangan sesuai permintaan dan terjadwal untuk Amazon Timestream.

Topik

Mengaktifkan AWS Backup untuk melindungi Timestream untuk data LiveAnalytics

Anda harus mengaktifkan AWS Backup untuk menggunakannya dengan Timestream untuk LiveAnalytics.

Untuk mengaktifkan AWS Backup di Timestream untuk LiveAnalytics konsol, lakukan langkah-langkah berikut.

1. Masuk ke Konsol AWS Manajemen.
2. Spanduk pop-up muncul di bagian atas Timestream Anda untuk halaman LiveAnalytics dasbor agar memungkinkan untuk mendukung Timestream AWS Backup untuk data. LiveAnalytics Jika tidak, dari panel navigasi, pilih Backup.
3. Di jendela Backup, Anda akan melihat spanduk untuk mengaktifkan AWS Backup. Pilih Aktifkan.

   Perlindungan Data melalui sekarang AWS Backup tersedia untuk Timestream Anda untuk LiveAnalytics tabel.

Untuk mengaktifkan melalui AWS Backup, lihat AWS Backup dokumentasi untuk mengaktifkan melalui konsol dan secara terprogram.

Jika Anda memilih untuk menonaktifkan AWS Backup dari perlindungan Timestream Anda untuk LiveAnalytics data setelah diaktifkan, masuk melalui AWS Backup konsol dan pindahkan sakelar ke kiri.

Jika Anda tidak dapat mengaktifkan atau menonaktifkan AWS Backup fitur, AWS admin Anda mungkin perlu melakukan tindakan tersebut.

Membuat cadangan sesuai permintaan

Untuk membuat cadangan Timestream sesuai permintaan untuk LiveAnalytics tabel, ikuti langkah-langkah ini.

1. Masuk ke Konsol AWS Manajemen.
2. Di panel navigasi pada sisi kiri konsol, pilih Cadangan.
3. Pilih Buat cadangan sesuai permintaan.
4. Lanjutkan untuk memilih pengaturan di jendela cadangan.
5. Anda dapat membuat cadangan sekarang, segera memulai pencadangan, atau memilih jendela cadangan untuk memulai pencadangan.
6. Pilih kebijakan manajemen siklus hidup cadangan Anda. Anda dapat mentransisikan data cadangan Anda ke penyimpanan dingin di mana Anda harus menyimpan cadangan selama minimal 90 hari. Anda dapat mengatur periode penyimpanan yang diperlukan untuk pencadangan Anda dapat memilih vault yang ada atau atau memilih buat brankas cadangan baru untuk menavigasi ke AWS Backup konsol dan membuat brankas cadangan baru **<**documentation link on creating a new backup vault here**>**
7. Pilih IAM peran yang sesuai.
8. Jika Anda ingin menetapkan satu atau beberapa tag ke cadangan sesuai permintaan, masukkan kunci dan nilai opsional, lalu pilih Tambah tag.
9. Pilih untuk membuat cadangan sesuai permintaan. Ini membawa Anda ke halaman Backup, di mana Anda akan melihat daftar pekerjaan.
10. Pilih ID pekerjaan Backup untuk sumber daya yang Anda pilih untuk dicadangkan untuk melihat detail pekerjaan itu.

Pencadangan terjadwal

Untuk menjadwalkan pencadangan, lihat Membuat cadangan terjadwal.

Memulihkan cadangan tabel Amazon Timestream

Bagian ini menjelaskan cara memulihkan cadangan tabel Amazon Timestream.

Topik

Memulihkan Timestream untuk LiveAnalytics tabel dari AWS Backup

Untuk memulihkan Timestream untuk LiveAnalytics tabel agar tidak AWS Backup menggunakan Timestream untuk LiveAnalytics konsol, ikuti langkah-langkah berikut.

1. Masuk ke Konsol AWS Manajemen.
2. Di panel navigasi pada sisi kiri konsol, pilih Cadangan.
3. Untuk memulihkan sumber daya, pilih tombol radio di sebelah ID titik pemulihan sumber daya. Di sudut kanan atas panel, pilih Pulihkan.
4. Masukkan pengaturan konfigurasi tabel, yaitu nama Database dan Nama Tabel. Harap dicatat, nama tabel yang dipulihkan harus berbeda dari nama tabel sumber asli.
5. Konfigurasikan pengaturan penyimpanan memori dan penyimpanan magnetik.
6. Untuk peran Pulihkan, pilih IAM peran yang AWS Backup akan diambil untuk pemulihan ini.
7. Pilih Pulihkan cadangan. Pesan di bagian atas halaman memberikan informasi tentang pekerjaan pemulihan.

> **Note**
>
> Anda dikenakan biaya untuk memulihkan seluruh cadangan terlepas dari memori yang dikonfigurasi dan periode penyimpanan penyimpanan magnetik. Namun, setelah pemulihan selesai, tabel yang dipulihkan hanya akan berisi data dalam periode retensi yang dikonfigurasi.

Memulihkan Timestream untuk LiveAnalytics tabel ke Wilayah atau akun lain

Untuk mengembalikan Timestream untuk LiveAnalytics tabel ke Wilayah atau akun lain, Anda harus terlebih dahulu menyalin cadangan ke Wilayah atau akun baru tersebut. Untuk menyalin ke akun

lain, akun tersebut harus memberi Anda izin terlebih dahulu. Setelah Anda menyalin Timestream Anda untuk LiveAnalytics cadangan ke Wilayah atau akun baru, itu dapat dipulihkan dengan proses di bagian sebelumnya.

Menyalin cadangan tabel Timestream Amazon

Anda dapat membuat salinan dari cadangan saat ini. Anda dapat menyalin cadangan ke beberapa AWS akun atau AWS Wilayah sesuai permintaan atau secara otomatis sebagai bagian dari rencana pencadangan terjadwal. Replikasi Lintas Wilayah sangat berharga jika Anda memiliki kelangsungan bisnis atau persyaratan kepatuhan untuk menyimpan cadangan dengan jarak minimum dari data produksi Anda.

Pencadangan lintas akun berguna untuk menyalin cadangan Anda dengan aman ke satu atau beberapa akun AWS di organisasi Anda untuk alasan operasional atau keamanan. Jika cadangan asli Anda terhapus, Anda dapat menyalin cadangan dari akun tujuannya ke akun sumbernya, lalu memulai pemulihan. Sebelum Anda dapat melakukan ini, Anda harus memiliki dua akun milik organisasi yang sama di layanan Organizations dan izin yang diperlukan untuk akun tersebut. Saat Anda menyalin cadangan tambahan ke akun atau Wilayah lain, cadangan lengkap terkait juga disalin.

Salinan mewarisi konfigurasi cadangan sumber kecuali Anda menentukan sebaliknya. Ada satu pengecualian. Jika Anda menentukan salinan baru Anda ke “Tidak Pernah” kedaluwarsa. Dengan pengaturan ini, salinan baru masih mewarisi tanggal kedaluwarsa sumbernya. Jika Anda ingin salinan cadangan baru Anda bersifat permanen, atur cadangan sumber Anda agar tidak pernah kedaluwarsa, atau tentukan salinan baru Anda untuk kedaluwarsa 100 tahun setelah pembuatannya.

Untuk menyalin cadangan dari konsol Timestream, ikuti langkah-langkah ini.

1. Masuk ke Konsol AWS Manajemen.
2. Di panel navigasi pada sisi kiri konsol, pilih Cadangan.
3. Pilih tombol radio di sebelah ID titik pemulihan sumber daya. Di sudut kanan atas panel, pilih Tindakan dan pilih Salin.
4. Pilih Lanjutkan ke AWS Cadangan dan ikuti langkah-langkah untuk pencadangan Cross account.

Menyalin cadangan sesuai permintaan dan terjadwal di seluruh akun dan Wilayah tidak didukung secara asli di Timestream untuk LiveAnalytics konsol saat ini dan Anda harus menavigasi ke untuk melakukan operasi. AWS Backup

Menghapus cadangan

Bagian ini menjelaskan cara menghapus cadangan Timestream untuk LiveAnalytics tabel.

Untuk menghapus cadangan dari konsol Timestream, ikuti langkah-langkah ini.

1. Masuk ke Konsol AWS Manajemen.
2. Di panel navigasi pada sisi kiri konsol, pilih Cadangan.
3. Pilih tombol radio di sebelah ID titik pemulihan sumber daya. Di sudut kanan atas panel, pilih Tindakan dan pilih Hapus.
4. Pilih Lanjutkan ke AWS Cadangan dan ikuti langkah-langkah untuk menghapus cadangan di Menghapus cadangan.

> **Note**
>
> Saat Anda menghapus cadangan yang bersifat inkremental, hanya cadangan tambahan yang dihapus dan cadangan penuh yang mendasarinya tidak dihapus.

Kuota dan batas

AWS Backup membatasi cadangan menjadi satu cadangan bersamaan per sumber daya. Oleh karena itu, permintaan cadangan terjadwal atau sesuai permintaan tambahan untuk sumber daya diantrian dan akan dimulai hanya setelah pekerjaan pencadangan yang ada selesai. Jika pekerjaan cadangan tidak dimulai atau diselesaikan dalam jendela cadangan, permintaan gagal. Untuk informasi selengkapnya tentang AWS Backup batasan, lihat Batas AWS Cadangan di Panduan Pengembang AWS Cadangan.

Saat membuat cadangan, Anda dapat menjalankan hingga empat cadangan bersamaan per akun. Demikian pula, Anda dapat menjalankan satu pemulihan bersamaan per akun. Ketika Anda memulai lebih dari empat pekerjaan cadangan secara bersamaan, hanya empat pekerjaan cadangan yang dimulai dan pekerjaan yang tersisa akan dicoba ulang secara berkala. Setelah dimulai, jika pekerjaan pencadangan tidak selesai dalam durasi jendela cadangan yang dikonfigurasi, pekerjaan pencadangan gagal. Jika pekerjaan pencadangan yang gagal adalah cadangan sesuai permintaan, Anda dapat mencoba kembali cadangan dan untuk pencadangan terjadwal, pekerjaan dicoba dalam jadwal berikut.

Kunci partisi yang ditentukan pelanggan

Amazon Timestream untuk kunci partisi LiveAnalytics yang ditentukan pelanggan adalah fitur di Timestream LiveAnalytics yang memungkinkan pelanggan menentukan kunci partisi mereka sendiri untuk tabel mereka. Partisi adalah teknik yang digunakan untuk mendistribusikan data di beberapa unit penyimpanan fisik, memungkinkan pengambilan data yang lebih cepat dan lebih efisien. Dengan kunci partisi yang ditentukan pelanggan, pelanggan dapat membuat skema partisi yang lebih selaras dengan pola kueri dan kasus penggunaan mereka.

Dengan Timestream untuk kunci partisi yang LiveAnalytics ditentukan pelanggan, pelanggan dapat memilih satu nama dimensi sebagai kunci partisi untuk tabel mereka. Hal ini memungkinkan lebih banyak fleksibilitas dalam mendefinisikan skema partisi untuk data mereka. Dengan memilih kunci partisi yang tepat, pelanggan dapat mengoptimalkan model data mereka, meningkatkan kinerja kueri mereka, dan mengurangi latensi kueri.

Topik

Menggunakan tombol partisi yang ditentukan pelanggan

Jika Anda memiliki pola kueri yang terdefinisi dengan baik dengan dimensi kardinalitas tinggi dan memerlukan latensi kueri rendah, Timestream untuk kunci partisi yang LiveAnalytics ditentukan pelanggan dapat menjadi alat yang berguna untuk menyempurnakan model data Anda. Misalnya, jika Anda adalah perusahaan ritel yang melacak interaksi pelanggan di situs web Anda, pola akses utama kemungkinan adalah dengan ID pelanggan dan stempel waktu. Dengan mendefinisikan ID pelanggan sebagai kunci partisi, data Anda dapat didistribusikan secara merata, memungkinkan latensi berkurang, yang pada akhirnya meningkatkan pengalaman pengguna.

Contoh lain adalah di industri perawatan kesehatan, di mana perangkat yang dapat dikenakan mengumpulkan data sensor untuk melacak tanda-tanda vital pasien. Pola akses utama adalah dengan ID Perangkat dan stempel waktu, dengan kardinalitas tinggi pada kedua dimensi. Dengan mendefinisikan ID Perangkat sebagai kunci partisi, dapat mengoptimalkan eksekusi kueri Anda dan memastikan kinerja kueri jangka panjang yang berkelanjutan.

Singkatnya, Timestream untuk kunci partisi LiveAnalytics yang ditentukan pelanggan paling berguna ketika Anda memiliki pola kueri yang jelas, dimensi kardinalitas tinggi, dan membutuhkan latensi rendah untuk kueri Anda. Dengan mendefinisikan kunci partisi yang selaras dengan pola kueri, Anda dapat mengoptimalkan eksekusi kueri dan memastikan kinerja kueri kinerja jangka panjang yang berkelanjutan.

Memulai dengan kunci partisi yang ditentukan pelanggan

Dari konsol, pilih Tabel dan buat tabel baru. Anda juga dapat menggunakan SDK untuk mengakses `CreateTable` tindakan untuk membuat tabel baru yang dapat menyertakan kunci partisi yang ditentukan pelanggan.

Buat tabel dengan kunci partisi tipe dimensi

Anda dapat menggunakan cuplikan kode berikut untuk membuat tabel dengan kunci partisi tipe dimensi.

Java

```
public void createTableWithDimensionTypePartitionKeyExample() {
        System.out.println("Creating table");
        CreateTableRequest createTableRequest = new CreateTableRequest();
        createTableRequest.setDatabaseName(DATABASE_NAME);
        createTableRequest.setTableName(TABLE_NAME);
        final RetentionProperties retentionProperties = new RetentionProperties()
                .withMemoryStoreRetentionPeriodInHours(HT_TTL_HOURS)
                .withMagneticStoreRetentionPeriodInDays(CT_TTL_DAYS);
        createTableRequest.setRetentionProperties(retentionProperties);

        // Can specify enforcement level with OPTIONAL or REQUIRED
        final List<PartitionKey> partitionKeyWithDimensionAndOptionalEnforcement =
 Collections.singletonList(new PartitionKey()
            .withName(COMPOSITE_PARTITION_KEY_DIM_NAME)
            .withType(PartitionKeyType.DIMENSION)
```

```
            .withEnforcementInRecord(PartitionKeyEnforcementLevel.OPTIONAL));
        Schema schema = new Schema();

 schema.setCompositePartitionKey(partitionKeyWithDimensionAndOptionalEnforcement);
        createTableRequest.setSchema(schema);

        try {
            writeClient.createTable(createTableRequest);
            System.out.println("Table [" + TABLE_NAME + "] successfully created.");
        } catch (ConflictException e) {
            System.out.println("Table [" + TABLE_NAME + "] exists on database [" +
 DATABASE_NAME + "] . Skipping database creation");
        }
    }
```

Java v2

```
public void createTableWithDimensionTypePartitionKeyExample() {
        System.out.println("Creating table");
        final RetentionProperties retentionProperties =
 RetentionProperties.builder()
                .memoryStoreRetentionPeriodInHours(HT_TTL_HOURS)
                .magneticStoreRetentionPeriodInDays(CT_TTL_DAYS)
                .build();
        // Can specify enforcement level with OPTIONAL or REQUIRED
        final List<PartitionKey> partitionKeyWithDimensionAndOptionalEnforcement =
 Collections.singletonList(PartitionKey
                .builder()
                .name(COMPOSITE_PARTITION_KEY_DIM_NAME)
                .type(PartitionKeyType.DIMENSION)
                .enforcementInRecord(PartitionKeyEnforcementLevel.OPTIONAL)
                .build());
        final Schema schema = Schema.builder()

 .compositePartitionKey(partitionKeyWithDimensionAndOptionalEnforcement).build();
        final CreateTableRequest createTableRequest = CreateTableRequest.builder()
                .databaseName(DATABASE_NAME)
                .tableName(TABLE_NAME)
                .retentionProperties(retentionProperties)
                .schema(schema)
                .build();

        try {
```

```
            writeClient.createTable(createTableRequest);
            System.out.println("Table [" + TABLE_NAME + "] successfully created.");
        } catch (ConflictException e) {
            System.out.println("Table [" + TABLE_NAME + "] exists on database [" +
 DATABASE_NAME + "] . Skipping database creation");
        }
    }
```

Go v1

```
func createTableWithDimensionTypePartitionKeyExample(){
        // Can specify enforcement level with OPTIONAL or REQUIRED
        partitionKeyWithDimensionAndOptionalEnforcement :=
 []*timestreamwrite.PartitionKey{
                {
                    Name:                aws.String(CompositePartitionKeyDimName),
                    EnforcementInRecord: aws.String("OPTIONAL"),
                    Type:                aws.String("DIMENSION"),
                },
        }
        createTableInput := &timestreamwrite.CreateTableInput{
             DatabaseName: aws.String(*databaseName),
             TableName:    aws.String(*tableName),
             // Enable MagneticStoreWrite for Table
             MagneticStoreWriteProperties:
 &timestreamwrite.MagneticStoreWriteProperties{
                 EnableMagneticStoreWrites: aws.Bool(true),
                 // Persist MagneticStoreWrite rejected records in S3
                 MagneticStoreRejectedDataLocation:
 &timestreamwrite.MagneticStoreRejectedDataLocation{
                     S3Configuration: &timestreamwrite.S3Configuration{
                         BucketName:       aws.String("timestream-sample-bucket"),
                         ObjectKeyPrefix:  aws.String("TimeStreamCustomerSampleGo"),
                         EncryptionOption: aws.String("SSE_S3"),
                     },
                 },
             },
             Schema: &timestreamwrite.Schema{
                 CompositePartitionKey:
 partitionKeyWithDimensionAndOptionalEnforcement,
             }
         }
         _, err := writeSvc.CreateTable(createTableInput)
```

```
    }
```

Go v2

```
 func (timestreamBuilder TimestreamBuilder)
CreateTableWithDimensionTypePartitionKeyExample() error {
        partitionKeyWithDimensionAndOptionalEnforcement := []types.PartitionKey{
            {
                Name:                aws.String(CompositePartitionKeyDimName),
                EnforcementInRecord: types.PartitionKeyEnforcementLevelOptional,
                Type:                types.PartitionKeyTypeDimension,
            },
        }
        _, err := timestreamBuilder.WriteSvc.CreateTable(context.TODO(),
&timestreamwrite.CreateTableInput{
            DatabaseName: aws.String(databaseName),
            TableName:    aws.String(tableName),
            MagneticStoreWriteProperties: &types.MagneticStoreWriteProperties{
                EnableMagneticStoreWrites: aws.Bool(true),
                // Persist MagneticStoreWrite rejected records in S3
                MagneticStoreRejectedDataLocation:
&types.MagneticStoreRejectedDataLocation{
                    S3Configuration: &types.S3Configuration{
                        BucketName:       aws.String(s3BucketName),
                        EncryptionOption: "SSE_S3",
                    },
                },
            },
            Schema: &types.Schema{
                CompositePartitionKey:
partitionKeyWithDimensionAndOptionalEnforcement,
            },
        })

        if err != nil {
            fmt.Println("Error:")
            fmt.Println(err)
        } else {
            fmt.Println("Create table is successful")
        }
        return err
    }
```

Python

```
def create_table_with_measure_name_type_partition_key(self):
        print("Creating table")
        retention_properties = {
            'MemoryStoreRetentionPeriodInHours': HT_TTL_HOURS,
            'MagneticStoreRetentionPeriodInDays': CT_TTL_DAYS
        }
        partitionKey_with_measure_name = [
            {'Type': 'MEASURE'}
        ]
        schema = {
            'CompositePartitionKey': partitionKey_with_measure_name
        }
        try:
            self.client.create_table(DatabaseName=DATABASE_NAME,
 TableName=TABLE_NAME,
                                     RetentionProperties=retention_properties,
 Schema=schema)
            print("Table [%s] successfully created." % TABLE_NAME)
        except self.client.exceptions.ConflictException:
            print("Table [%s] exists on database [%s]. Skipping table creation" % (
                TABLE_NAME, DATABASE_NAME))
        except Exception as err:
            print("Create table failed:", err)
```

Memeriksa konfigurasi skema partisi

Anda dapat memeriksa bagaimana konfigurasi tabel untuk skema partisi dalam beberapa cara. Dari konsol, pilih Database dan pilih tabel untuk diperiksa. Anda juga dapat menggunakan sebuah SDK untuk mengakses `DescribeTable` tindakan.

Jelaskan tabel dengan kunci partisi

Anda dapat menggunakan cuplikan kode berikut untuk menggambarkan tabel dengan kunci partisi.

Java

```
    public void describeTable() {
        System.out.println("Describing table");
        final DescribeTableRequest describeTableRequest = new
 DescribeTableRequest();
```

```
        describeTableRequest.setDatabaseName(DATABASE_NAME);
        describeTableRequest.setTableName(TABLE_NAME);
        try {
            DescribeTableResult result =
amazonTimestreamWrite.describeTable(describeTableRequest);
            String tableId = result.getTable().getArn();
            System.out.println("Table " + TABLE_NAME + " has id " + tableId);
            // If table is created with composite partition key, it can be described
with
            //
System.out.println(result.getTable().getSchema().getCompositePartitionKey());
        } catch (final Exception e) {
            System.out.println("Table " + TABLE_NAME + " doesn't exist = " + e);
            throw e;
        }
    }
```

Berikut ini adalah output contoh.

1. Tabel memiliki kunci partisi tipe dimensi

```
[{Type: DIMENSION,Name: hostId,EnforcementInRecord: OPTIONAL}]
```

2. Tabel memiliki kunci partisi tipe nama ukuran

```
[{Type: MEASURE,}]
```

3. Mendapatkan kunci partisi komposit dari tabel yang dibuat tanpa menentukan kunci partisi komposit

```
[{Type: MEASURE,}]
```

Java v2

```
 public void describeTable() {
        System.out.println("Describing table");
        final DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
                .databaseName(DATABASE_NAME).tableName(TABLE_NAME).build();
        try {
```

```
            DescribeTableResponse response =
writeClient.describeTable(describeTableRequest);
            String tableId = response.table().arn();
            System.out.println("Table " + TABLE_NAME + " has id " + tableId);
            // If table is created with composite partition key, it can be described
with
            //
System.out.println(response.table().schema().compositePartitionKey());
        } catch (final Exception e) {
            System.out.println("Table " + TABLE_NAME + " doesn't exist = " + e);
            throw e;
        }
    }
```

Berikut ini adalah output contoh.

1. Tabel memiliki kunci partisi tipe dimensi

```
[PartitionKey(Type=DIMENSION, Name=hostId, EnforcementInRecord=OPTIONAL)]
```

2. Tabel memiliki kunci partisi tipe nama ukuran

```
[PartitionKey(Type=MEASURE)]
```

3. Mendapatkan kunci partisi komposit dari tabel yang dibuat tanpa menentukan kunci partisi komposit akan kembali

```
[PartitionKey(Type=MEASURE)]
```

Go v1

```
    <tablistentry>
     <tabname> Go </tabname>
     <tabcontent>
      <programlisting language="go"></programlisting>
     </tabcontent>
    </tablistentry>
```

Berikut ini adalah sebuah contoh output.

```
{
```

```
  Table: {
    Arn: "arn:aws:timestream:us-west-2:533139590831:database/devops/table/
host_metrics_dim_pk_1",
    CreationTime: 2023-05-31 01:52:00.511 +0000 UTC,
    DatabaseName: "devops",
    LastUpdatedTime: 2023-05-31 01:52:00.511 +0000 UTC,
    MagneticStoreWriteProperties: {
      EnableMagneticStoreWrites: true,
      MagneticStoreRejectedDataLocation: {
        S3Configuration: {
          BucketName: "timestream-sample-bucket-west",
          EncryptionOption: "SSE_S3",
          ObjectKeyPrefix: "TimeStreamCustomerSampleGo"
        }
      }
    },
    RetentionProperties: {
      MagneticStoreRetentionPeriodInDays: 73000,
      MemoryStoreRetentionPeriodInHours: 6
    },
    Schema: {
      CompositePartitionKey: [{
          EnforcementInRecord: "OPTIONAL",
          Name: "hostId",
          Type: "DIMENSION"
        }]
    },
    TableName: "host_metrics_dim_pk_1",
    TableStatus: "ACTIVE"
  }
}
```

Go v2

```
 func (timestreamBuilder TimestreamBuilder) DescribeTable()
 (*timestreamwrite.DescribeTableOutput, error) {
         describeTableInput := &timestreamwrite.DescribeTableInput{
             DatabaseName: aws.String(databaseName),
             TableName:    aws.String(tableName),
         }
        describeTableOutput, err :=
 timestreamBuilder.WriteSvc.DescribeTable(context.TODO(), describeTableInput)
```

```
        if err != nil {
            fmt.Printf("Failed to describe table with Error: %s", err.Error())
        } else {
            fmt.Printf("Describe table is successful : %s\n",
 JsonMarshalIgnoreError(*describeTableOutput))
            // If table is created with composite partition key, it will be included
 in the output
        }

        return describeTableOutput, err
    }
```

Berikut ini adalah sebuah contoh output.

```
{
  "Table": {
    "Arn":"arn:aws:timestream:us-east-1:351861611069:database/cdpk-wr-db/table/
host_metrics_dim_pk",
    "CreationTime":"2023-05-31T22:36:10.66Z",
    "DatabaseName":"cdpk-wr-db",
    "LastUpdatedTime":"2023-05-31T22:36:10.66Z",
    "MagneticStoreWriteProperties":{
      "EnableMagneticStoreWrites":true,
      "MagneticStoreRejectedDataLocation":{
        "S3Configuration":{
          "BucketName":"error-configuration-sample-s3-bucket-cq8my",
          "EncryptionOption":"SSE_S3",
          "KmsKeyId":null,"ObjectKeyPrefix":null
        }
      }
    },
    "RetentionProperties":{
      "MagneticStoreRetentionPeriodInDays":73000,
      "MemoryStoreRetentionPeriodInHours":6
    },
    "Schema":{
      "CompositePartitionKey":[{
        "Type":"DIMENSION",
        "EnforcementInRecord":"OPTIONAL",
        "Name":"hostId"
      }]
    },
    "TableName":"host_metrics_dim_pk",
```

```
    "TableStatus":"ACTIVE"
  },
  "ResultMetadata":{}
}
```

Python

```
 def describe_table(self):
       print('Describing table')
       try:
           result = self.client.describe_table(DatabaseName=DATABASE_NAME,
TableName=TABLE_NAME)
           print("Table [%s] has id [%s]" % (TABLE_NAME, result['Table']['Arn']))
           # If table is created with composite partition key, it can be described
with
           # print(result['Table']['Schema'])
       except self.client.exceptions.ResourceNotFoundException:
           print("Table doesn't exist")
       except Exception as err:
           print("Describe table failed:", err)
```

Berikut ini adalah output contoh.

1. Tabel memiliki kunci partisi tipe dimensi

```
[{'CompositePartitionKey': [{'Type': 'DIMENSION', 'Name': 'hostId',
 'EnforcementInRecord': 'OPTIONAL'}]}]
```

2. Tabel memiliki kunci partisi tipe nama ukuran

```
[{'CompositePartitionKey': [{'Type': 'MEASURE'}]}]
```

3. Mendapatkan kunci partisi komposit dari tabel yang dibuat tanpa menentukan kunci partisi komposit

```
[{'CompositePartitionKey': [{'Type': 'MEASURE'}]}]
```

Memperbarui konfigurasi skema partisi

Anda dapat memperbarui konfigurasi tabel untuk skema partisi SDK dengan akses tindakan. `UpdateTable`

Perbarui tabel dengan kunci partisi

Anda dapat menggunakan cuplikan kode berikut untuk memperbarui tabel dengan kunci partisi.

Java

```
   public void updateTableCompositePartitionKeyEnforcement() {
       System.out.println("Updating table");

       UpdateTableRequest updateTableRequest = new UpdateTableRequest();
       updateTableRequest.setDatabaseName(DATABASE_NAME);
       updateTableRequest.setTableName(TABLE_NAME);

       // Can update enforcement level for dimension type partition key with
OPTIONAL or REQUIRED enforcement
       final List<PartitionKey> partitionKeyWithDimensionAndRequiredEnforcement =
Collections.singletonList(new PartitionKey()
           .withName(COMPOSITE_PARTITION_KEY_DIM_NAME)
           .withType(PartitionKeyType.DIMENSION)
           .withEnforcementInRecord(PartitionKeyEnforcementLevel.REQUIRED));
       Schema schema = new Schema();

schema.setCompositePartitionKey(partitionKeyWithDimensionAndRequiredEnforcement);
       updateTableRequest.withSchema(schema);

       writeClient.updateTable(updateTableRequest);
       System.out.println("Table updated");
```

Java v2

```
   public void updateTableCompositePartitionKeyEnforcement() {
       System.out.println("Updating table");
       // Can update enforcement level for dimension type partition key with
OPTIONAL or REQUIRED enforcement
       final List<PartitionKey> partitionKeyWithDimensionAndRequiredEnforcement =
Collections.singletonList(PartitionKey
           .builder()
           .name(COMPOSITE_PARTITION_KEY_DIM_NAME)
```

```
            .type(PartitionKeyType.DIMENSION)
            .enforcementInRecord(PartitionKeyEnforcementLevel.REQUIRED)
            .build());
        final Schema schema = Schema.builder()

.compositePartitionKey(partitionKeyWithDimensionAndRequiredEnforcement).build();
        final UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()

.databaseName(DATABASE_NAME).tableName(TABLE_NAME).schema(schema).build();

        writeClient.updateTable(updateTableRequest);
        System.out.println("Table updated");
```

Go v1

```
// Update table partition key enforcement attribute
   updateTableInput := &timestreamwrite.UpdateTableInput{
        DatabaseName: aws.String(*databaseName),
        TableName:    aws.String(*tableName),
        // Can update enforcement level for dimension type partition key with
OPTIONAL or REQUIRED enforcement
        Schema: &timestreamwrite.Schema{
            CompositePartitionKey: []*timestreamwrite.PartitionKey{
                {
                        Name:
aws.String(CompositePartitionKeyDimName),
                        EnforcementInRecord: aws.String("REQUIRED"),
                        Type:               aws.String("DIMENSION"),
                },
            }},
    }
    updateTableOutput, err := writeSvc.UpdateTable(updateTableInput)
        if err != nil {
            fmt.Println("Error:")
            fmt.Println(err)
        } else {
            fmt.Println("Update table is successful, below is the output:")
            fmt.Println(updateTableOutput)
        }
```

Go v2

```
// Update table partition key enforcement attribute
```

```
        updateTableInput := &timestreamwrite.UpdateTableInput{
            DatabaseName: aws.String(*databaseName),
            TableName:    aws.String(*tableName),
            // Can update enforcement level for dimension type partition key with
OPTIONAL or REQUIRED enforcement
            Schema: &types.Schema{
                CompositePartitionKey: []types.PartitionKey{
                    {
                        Name:
aws.String(CompositePartitionKeyDimName),
                        EnforcementInRecord:
types.PartitionKeyEnforcementLevelRequired,
                        Type:               types.PartitionKeyTypeDimension,
                    },
                }},
        }
        updateTableOutput, err :=
timestreamBuilder.WriteSvc.UpdateTable(context.TODO(), updateTableInput)
        if err != nil {
            fmt.Println("Error:")
            fmt.Println(err)
        } else {
            fmt.Println("Update table is successful, below is the output:")
            fmt.Println(updateTableOutput)
        }
```

Python

```
   def update_table(self):
       print('Updating table')
       try:
           # Can update enforcement level for dimension type partition key with
OPTIONAL or REQUIRED enforcement
           partition_key_with_dimension_and_required_enforcement = [
               {
                   'Type': 'DIMENSION',
                   'Name': COMPOSITE_PARTITION_KEY_DIM_NAME,
                   'EnforcementInRecord': 'REQUIRED'
               }
           ]
           schema = {
               'CompositePartitionKey':
partition_key_with_dimension_and_required_enforcement
```

```
            }
            self.client.update_table(DatabaseName=DATABASE_NAME,
TableName=TABLE_NAME,
                                     Schema=schema)
            print('Table updated.')
        except Exception as err:
            print('Update table failed:', err)
```

Keuntungan dari kunci partisi yang ditentukan pelanggan

Kinerja kueri yang ditingkatkan: Kunci partisi yang ditentukan pelanggan memungkinkan Anda mengoptimalkan eksekusi kueri dan meningkatkan kinerja kueri secara keseluruhan. Dengan mendefinisikan kunci partisi yang selaras dengan pola kueri, Anda dapat meminimalkan pemindaian data dan mengoptimalkan pemangkasan data, sehingga latensi kueri lebih rendah.

Prediktabilitas kinerja jangka panjang yang lebih baik: Kunci partisi yang ditentukan pelanggan memungkinkan pelanggan untuk mendistribusikan data secara merata di seluruh partisi, meningkatkan efisiensi manajemen data. Ini akan memastikan bahwa kinerja kueri Anda tetap stabil karena skala data yang disimpan dari waktu ke waktu.

Keterbatasan kunci partisi yang ditentukan pelanggan

Sebagai Timestream untuk LiveAnalytics pengguna, penting untuk mengingat keterbatasan di sekitar kunci partisi pelanggan. Pertama, ini membutuhkan pemahaman yang baik tentang beban kerja dan pola kueri Anda. Ini berarti bahwa Anda harus memiliki gagasan yang jelas tentang dimensi mana yang paling sering digunakan sebagai kondisi penyaringan utama dalam kueri dan memiliki kardinalitas tinggi untuk membuat penggunaan kunci partisi yang paling efektif.

Kedua, kunci partisi perlu didefinisikan pada saat pembuatan tabel dan tidak dapat ditambahkan ke tabel yang ada. Ini berarti bahwa Anda harus hati-hati mempertimbangkan strategi partisi Anda sebelum membuat tabel untuk memastikan bahwa itu selaras dengan kebutuhan bisnis Anda.

Terakhir, penting untuk dicatat bahwa setelah tabel dibuat, Anda tidak dapat mengubah kunci partisi sesudahnya. Ini berarti bahwa Anda harus benar-benar menguji dan mengevaluasi strategi partisi Anda sebelum berkomitmen untuk itu. Dengan keterbatasan ini, kunci partisi yang ditentukan pelanggan Timestream dapat sangat meningkatkan kinerja kueri dan kepuasan jangka panjang.

Kunci partisi yang ditentukan pelanggan dan dimensi kardinalitas rendah

Jika Anda memutuskan untuk menggunakan kunci partisi dengan kardinalitas yang sangat rendah, seperti wilayah atau negara tertentu, penting untuk dicatat bahwa data untuk entitas lain seperti`customerID`,, dan lainnya`ProductCategory`, dapat berakhir tersebar di terlalu banyak partisi kadang-kadang dengan sedikit atau tanpa data yang ada. Hal ini dapat menyebabkan eksekusi query yang tidak efisien dan penurunan kinerja.

Untuk menghindari hal ini, kami sarankan Anda memilih dimensi yang tidak hanya bagian dari kondisi penyaringan kunci Anda tetapi memiliki kardinalitas yang lebih tinggi. Ini akan membantu memastikan bahwa data didistribusikan secara merata di seluruh partisi dan meningkatkan kinerja kueri.

Membuat kunci partisi untuk tabel yang ada

Jika Anda sudah memiliki tabel di Timestream untuk LiveAnalytics dan ingin menggunakan kunci partisi yang ditentukan pelanggan, Anda harus memigrasikan data Anda ke tabel baru dengan definisi skema partisi yang diinginkan. Ini dapat dilakukan dengan menggunakan ekspor ke S3 dan pemuatan batch bersama-sama, yang melibatkan mengekspor data dari tabel yang ada ke S3, memodifikasi data untuk memasukkan kunci partisi (jika perlu) dan menambahkan header ke CSV file Anda, dan kemudian mengimpor data ke tabel baru dengan skema partisi yang diinginkan ditentukan. Perlu diingat bahwa metode ini bisa memakan waktu dan mahal, terutama untuk meja besar.

Atau, Anda dapat menggunakan kueri terjadwal untuk memigrasikan data Anda ke tabel baru dengan skema partisi yang diinginkan. Metode ini melibatkan pembuatan kueri terjadwal yang membaca dari tabel yang ada dan menulis ke tabel baru. Kueri terjadwal dapat diatur untuk berjalan secara teratur sampai semua data telah dimigrasikan. Ingatlah bahwa Anda akan dikenakan biaya untuk membaca dan menulis data selama proses migrasi.

Timestream untuk validasi LiveAnalytics skema dengan kunci partisi komposit khusus

Validasi skema di Timestream untuk LiveAnalytics membantu memastikan bahwa data yang dicerna ke dalam database sesuai dengan skema yang ditentukan, meminimalkan kesalahan konsumsi dan meningkatkan kualitas data. Secara khusus, validasi skema sangat berguna saat mengadopsi kunci partisi yang ditentukan pelanggan dengan tujuan mengoptimalkan kinerja kueri Anda.

Apa itu Timestream untuk validasi LiveAnalytics skema dengan kunci partisi yang ditentukan pelanggan?

Timestream untuk validasi LiveAnalytics skema adalah fitur yang memvalidasi data yang dicerna ke dalam Timestream untuk LiveAnalytics tabel berdasarkan skema yang telah ditentukan. Skema ini mendefinisikan model data, termasuk kunci partisi, tipe data, dan kendala untuk catatan yang dimasukkan.

Saat menggunakan kunci partisi yang ditentukan pelanggan, validasi skema menjadi lebih penting. Kunci partisi memungkinkan Anda untuk menentukan kunci partisi, yang menentukan bagaimana data Anda disimpan di Timestream untuk LiveAnalytics. Dengan memvalidasi data yang masuk terhadap skema dengan kunci partisi khusus, Anda dapat menerapkan konsistensi data, mendeteksi kesalahan lebih awal, dan meningkatkan kualitas keseluruhan data yang disimpan di Timestream untuk. LiveAnalytics

Cara Menggunakan Timestream untuk validasi LiveAnalytics skema dengan kunci partisi komposit khusus

Untuk menggunakan Timestream untuk validasi LiveAnalytics skema dengan kunci partisi komposit kustom, ikuti langkah-langkah berikut:

Pikirkan tentang seperti apa pola kueri Anda: Untuk memilih dan menentukan skema Timestream Anda untuk LiveAnalytics tabel dengan benar, Anda harus mulai dengan persyaratan kueri Anda.

Tentukan kunci partisi komposit khusus: Saat membuat tabel, tentukan kunci partisi khusus. Kunci ini menentukan atribut yang akan digunakan untuk mempartisi data tabel. Anda dapat memilih antara tombol dimensi dan tombol ukur untuk partisi. Kunci dimensi mempartisi data berdasarkan nama dimensi, sedangkan kunci ukuran mempartisi data berdasarkan nama ukuran.

Tetapkan tingkat penegakan: Untuk memastikan partisi data yang tepat dan manfaat yang menyertainya, Amazon Timestream LiveAnalytics for memungkinkan Anda mengatur tingkat penegakan untuk setiap kunci partisi dalam skema Anda. Tingkat penegakan menentukan apakah dimensi kunci partisi diperlukan atau opsional saat menelan catatan. Anda dapat memilih di antara dua opsi:`REQUIRED`, yang berarti kunci partisi harus ada dalam catatan tertelan`OPTIONAL`, dan, yang berarti kunci partisi tidak harus ada. Disarankan agar Anda menggunakan tingkat `REQUIRED` penegakan saat menggunakan partisi yang ditentukan pelanggan untuk memastikan bahwa data Anda dipartisi dengan benar dan Anda mendapatkan manfaat penuh dari fitur ini. Selain itu, Anda dapat mengubah konfigurasi tingkat penegakan kapan saja setelah pembuatan skema untuk menyesuaikan dengan persyaratan konsumsi data Anda.

Menyerap data: Saat memasukkan data ke Timestream untuk LiveAnalytics tabel, proses validasi skema akan memeriksa catatan terhadap skema yang ditentukan dengan kunci partisi komposit khusus. Jika catatan tidak mematuhi skema, Timestream for LiveAnalytics akan mengembalikan kesalahan validasi.

Menangani kesalahan validasi: Jika terjadi kesalahan validasi, Timestream for LiveAnalytics akan mengembalikan a `ValidationException` atau a`RejectedRecordsException`, tergantung pada jenis kesalahannya. Pastikan untuk menangani pengecualian ini dalam aplikasi Anda dan mengambil tindakan yang tepat, seperti memperbaiki catatan yang salah dan mencoba kembali konsumsi.

Perbarui tingkat penegakan: Jika perlu, Anda dapat memperbarui tingkat penegakan kunci partisi setelah pembuatan tabel menggunakan `UpdateTable` tindakan. Namun, penting untuk dicatat bahwa beberapa aspek konfigurasi kunci partisi, seperti nama, dan jenis, tidak dapat diubah setelah pembuatan tabel. Jika Anda mengubah tingkat penegakan dari `REQUIRED` ke`OPTIONAL`, semua catatan akan diterima terlepas dari keberadaan atribut yang dipilih sebagai kunci partisi yang ditentukan pelanggan. Sebaliknya, jika Anda mengubah tingkat penegakan dari `OPTIONAL` ke`REQUIRED`, Anda mungkin mulai melihat kesalahan penulisan 4xx untuk catatan yang tidak memenuhi kondisi ini. Oleh karena itu, penting untuk memilih tingkat penegakan yang sesuai untuk kasus penggunaan Anda saat membuat tabel Anda, berdasarkan persyaratan partisi data Anda.

Kapan menggunakan Timestream untuk validasi LiveAnalytics skema dengan kunci partisi komposit khusus

Timestream untuk validasi LiveAnalytics skema dengan kunci partisi komposit khusus harus digunakan dalam skenario di mana konsistensi data, kualitas, dan partisi yang dioptimalkan sangat penting. Dengan menerapkan skema selama konsumsi data, Anda dapat mencegah kesalahan dan inkonsistensi yang dapat menyebabkan analisis yang salah atau hilangnya wawasan berharga.

Interaksi dengan pekerjaan pemuatan batch

Saat menyiapkan pekerjaan pemuatan batch untuk mengimpor data ke dalam tabel dengan kunci partisi yang ditentukan pelanggan, ada beberapa skenario yang dapat memengaruhi proses:

1. Jika tingkat penegakan diatur ke`OPTIONAL`, peringatan akan ditampilkan di konsol selama alur pembuatan jika kunci partisi tidak dipetakan selama konfigurasi pekerjaan. Peringatan ini tidak akan muncul saat menggunakan API atauCLI.
2. Jika tingkat penegakan diatur ke`REQUIRED`, penciptaan pekerjaan akan ditolak kecuali kunci partisi dipetakan ke kolom data sumber.

3. Jika tingkat penegakan diubah menjadi `REQUIRED` setelah pekerjaan dibuat, pekerjaan akan terus dijalankan, tetapi catatan apa pun yang tidak memiliki pemetaan yang tepat untuk kunci partisi akan ditolak dengan kesalahan 4xx.

Interaksi dengan kueri terjadwal

Saat menyiapkan pekerjaan kueri terjadwal untuk menghitung dan menyimpan agregat, rollup, dan bentuk lain dari data yang telah diproses sebelumnya ke dalam tabel dengan kunci partisi yang ditentukan pelanggan, ada beberapa skenario yang dapat memengaruhi proses:

1. Jika tingkat penegakan diatur ke`OPTIONAL`, peringatan akan ditampilkan jika kunci partisi tidak dipetakan selama konfigurasi pekerjaan. Peringatan ini tidak akan muncul saat menggunakan API atauCLI.
2. Jika tingkat penegakan diatur ke`REQUIRED`, penciptaan pekerjaan akan ditolak kecuali kunci partisi dipetakan ke kolom data sumber.
3. Jika tingkat penegakan diubah menjadi `REQUIRED` setelah pekerjaan dibuat dan hasil kueri terjadwal tidak berisi dimensi kunci partisi, semua iterasi pekerjaan berikutnya akan gagal.

Menambahkan tanda dan label untuk sumber daya

Anda dapat memberi label Amazon Timestream untuk LiveAnalytics sumber daya menggunakan tag. Tag memungkinkan Anda mengkategorikan sumber daya Anda dengan cara yang berbeda—misalnya, berdasarkan tujuan, pemilik, lingkungan, atau kriteria lainnya. Tanda membantu Anda melakukan hal berikut:

- Identifikasi sumber daya dengan cepat berdasarkan tanda yang Anda tetapkan padanya.
- Lihat AWS tagihan yang dipecah berdasarkan tag.

Penandaan didukung oleh AWS layanan seperti Amazon Elastic Compute Cloud (AmazonEC2), Amazon Simple Storage Service (Amazon S3), Timestream for, dan banyak lagi. LiveAnalytics Penandaan yang efisien dapat memberikan wawasan biaya dengan memungkinkan Anda membuat laporan di seluruh layanan yang membawa tanda tertentu.

Untuk memulai penandaan, lakukan hal berikut:

1. Memahami pembatasan penandaan.

2. Buat tag dengan menggunakan operasi Penandaan.

Terakhir, merupakan praktik yang baik untuk mengikuti strategi penandaan yang optimal. Untuk informasi, lihat Strategi Pemberian Tag AWS.

Pembatasan penandaan

Setiap tanda terdiri dari kunci dan nilai, yang keduanya Anda tentukan. Pembatasan berikut berlaku:

- Setiap Timestream untuk LiveAnalytics tabel hanya dapat memiliki satu tag dengan kunci yang sama. Jika Anda mencoba menambahkan tag yang ada, nilai tag yang ada diperbarui ke nilai baru.
- Nilai bertindak sebagai deskriptor dalam kategori tag. Dalam Timestream untuk LiveAnalytics nilai tidak bisa kosong atau null.
- Kunci dan nilai tanda peka huruf besar-kecil.
- Panjang kunci maksimum adalah 128 karakter Unicode.
- Panjang nilai maksimum adalah 256 karakter Unicode.
- Karakter yang diperbolehkan adalah huruf, spasi kosong, dan angka, ditambah karakter khusus berikut: `+ - = . _ : /`
- Jumlah maksimum tanda per sumber daya adalah 50.
- AWS-Assigned nama tag dan nilai secara otomatis diberi `aws:` awalan, yang tidak dapat Anda tetapkan. AWS-nama-nama tag yang ditetapkan tidak dihitung terhadap batas tag 50. Nama tanda yang ditetapkan pengguna memiliki prefiks `user:` dalam laporan alokasi biaya.
- Anda tidak dapat mengubah penerapan tanda ke versi sebelumnya.

Operasi penandaan

Anda dapat menambahkan, membuat daftar, mengedit, atau menghapus tag untuk database dan tabel menggunakan Amazon Timestream LiveAnalytics untuk konsol, bahasa kueri, atau AWS Command Line Interface ().AWS CLI

Topik

- Menambahkan tag ke database dan tabel baru atau yang sudah ada menggunakan konsol

Menambahkan tag ke database dan tabel baru atau yang sudah ada menggunakan konsol

Anda dapat menggunakan Timestream for LiveAnalytics console untuk menambahkan tag ke database baru, tabel, dan kueri terjadwal saat Anda membuatnya. Anda juga dapat menambahkan, mengedit, atau menghapus tag untuk tabel yang ada.

Untuk menandai database saat membuatnya (konsol)

1. Buka konsol Timestream di https://console.aws.amazon.com/timestream.
2. Di panel navigasi, pilih Database, lalu pilih Buat database.
3. Pada halaman Buat database, berikan nama untuk database. Masukkan kunci dan nilai untuk tag, lalu pilih Tambahkan tag baru.
4. Pilih Buat basis data.

Untuk menandai tabel saat membuatnya (konsol)

1. Buka konsol Timestream di https://console.aws.amazon.com/timestream.
2. Di panel navigasi, pilih Tabel, lalu pilih Buat tabel.
3. Pada halaman Create Timestream untuk LiveAnalytics tabel, berikan nama untuk tabel. Masukkan kunci dan nilai untuk tag, dan pilih Tambahkan tag baru.
4. Pilih Buat tabel.

Untuk menandai kueri terjadwal saat membuatnya (konsol)

1. Buka konsol Timestream di https://console.aws.amazon.com/timestream.
2. Di panel navigasi, pilih Kueri terjadwal, lalu pilih Buat kueri terjadwal.
3. Pada Langkah 3. Konfigurasikan halaman pengaturan kueri, pilih Tambahkan tag baru. Masukkan kunci dan nilai untuk tanda. Pilih Tambahkan tag baru untuk menambahkan tag tambahan.
4. Pilih Berikutnya.

Untuk memberi tanda pada sumber daya yang ada (konsol)

1. Buka konsol Timestream di https://console.aws.amazon.com/timestream.

2. Di panel navigasi, pilih Database, Tabel, atau Kueri terjadwal.
3. Pilih database atau tabel dalam daftar. Kemudian pilih Kelola tag untuk menambah, mengedit, atau menghapus tag Anda.

Untuk informasi selengkapnya tentang struktur tag, lihat Pembatasan penandaan.

Keamanan di Timestream untuk LiveAnalytics

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

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. Model tanggung jawab bersama menjelaskan hal ini sebagai keamanan cloud dan keamanan dalam cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan AWS layanan di AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Efektivitas keamanan kami diuji dan diverifikasi secara rutin oleh auditor pihak ketiga sebagai bagian dari program kepatuhan AWS. Untuk mempelajari tentang program kepatuhan yang berlaku untuk Timestream LiveAnalytics, lihat AWS Layanan dalam Lingkup berdasarkan Program Kepatuhan.
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, termasuk sensitivitas data, persyaratan perusahaan, serta hukum dan peraturan yang berlaku.

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

Topik

- Perlindungan data di Timestream untuk LiveAnalytics
- Identitas dan manajemen akses untuk Amazon Timestream untuk LiveAnalytics
- Pencatatan dan pemantauan di Timestream untuk LiveAnalytics

- Ketahanan di Amazon Timestream Live Analytics
- Keamanan infrastruktur di Amazon Timestream Live Analytics
- Analisis konfigurasi dan kerentanan di Timestream
- Respons insiden di Timestream untuk LiveAnalytics
- VPCtitik akhir ()AWS PrivateLink
- Praktik terbaik keamanan untuk Amazon Timestream untuk LiveAnalytics

Perlindungan data di Timestream untuk LiveAnalytics

Model tanggung jawab AWS bersama model berlaku untuk perlindungan data di Amazon Timestream Live Analytics. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk mempertahankan kendali atas konten yang di-host pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Untuk informasi selengkapnya tentang privasi data, lihat Privasi Data FAQ. Untuk informasi tentang perlindungan data di Eropa, lihat Model Tanggung Jawab AWS Bersama dan posting GDPR blog di Blog AWS Keamanan.

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensil dan mengatur pengguna individu dengan 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 otentikasi multi-faktor (MFA) dengan setiap akun.
- GunakanSSL/TLSuntuk berkomunikasi dengan AWS sumber daya. Kami membutuhkan TLS 1.2 dan merekomendasikan TLS 1.3.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail. Untuk informasi tentang penggunaan CloudTrail jejak untuk menangkap AWS aktivitas, lihat Bekerja dengan CloudTrail jejak di AWS CloudTrail Panduan Pengguna.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya 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 FIPS 140-3 modul kriptografi yang divalidasi saat mengakses AWS melalui antarmuka baris perintah atau, gunakan titik akhir. API FIPS Untuk informasi selengkapnya tentang FIPS titik akhir yang tersedia, lihat Federal Information Processing Standard (FIPS) 140-3.

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 Timestream Live Analytics atau lainnya Layanan AWS menggunakan konsol,, API AWS CLI, atau AWS SDKs. 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. Jika Anda memberikan URL ke server eksternal, kami sangat menyarankan agar Anda tidak menyertakan informasi kredensil dalam URL untuk memvalidasi permintaan Anda ke server tersebut.

Untuk informasi lebih rinci tentang Timestream untuk topik perlindungan LiveAnalytics data seperti Enkripsi saat Istirahat dan Manajemen Kunci, pilih salah satu topik yang tersedia di bawah ini.

Topik

Enkripsi diam

Timestream untuk LiveAnalytics enkripsi saat istirahat memberikan keamanan yang ditingkatkan dengan mengenkripsi semua data Anda saat istirahat menggunakan kunci enkripsi yang disimpan di AWS Key Management Service ().AWS KMS Fungsi ini membantu mengurangi beban operasional dan kompleksitas yang terlibat dalam melindungi data sensitif. Dengan enkripsi saat diam, Anda dapat membuat aplikasi yang sensitif terhadap keamanan yang memenuhi persyaratan kepatuhan dan peraturan enkripsi yang ketat.

- Enkripsi diaktifkan secara default pada Timestream Anda untuk LiveAnalytics database, dan tidak dapat dimatikan. Algoritma enkripsi standar industri AES -256 adalah algoritma enkripsi default yang digunakan.
- AWS KMS diperlukan untuk enkripsi saat istirahat di Timestream untuk LiveAnalytics.
- Anda tidak dapat mengenkripsi subset item saja dalam tabel.
- Anda tidak perlu memodifikasi aplikasi klien basis data untuk menggunakan enkripsi.

Jika Anda tidak memberikan kunci, Timestream untuk LiveAnalytics membuat dan menggunakan AWS KMS kunci bernama `alias/aws/timestream` di akun Anda.

Anda dapat menggunakan kunci terkelola pelanggan Anda sendiri KMS untuk mengenkripsi Timestream Anda untuk LiveAnalytics data. Untuk informasi selengkapnya tentang kunci di Timestream LiveAnalytics, lihatManajemen kunci.

Timestream untuk LiveAnalytics menyimpan data Anda dalam dua tingkatan penyimpanan, penyimpanan memori dan penyimpanan magnetik. Data penyimpanan memori dienkripsi menggunakan Timestream untuk LiveAnalytics kunci layanan. Data penyimpanan magnetik dienkripsi menggunakan kunci Anda AWS KMS.

Layanan Timestream Query memerlukan kredensyal untuk mengakses data Anda. Kredensi ini dienkripsi menggunakan kunci Anda. KMS

> **Note**
>
> Timestream for LiveAnalytics tidak memanggil setiap AWS KMS operasi Dekripsi. Sebaliknya, ia mempertahankan cache kunci lokal selama 5 menit dengan lalu lintas aktif. Setiap perubahan izin disebarkan melalui Timestream untuk LiveAnalytics sistem dengan konsistensi akhirnya dalam waktu paling lama 5 menit.

Enkripsi bergerak

Semua data Timestream Live Analytics Anda dienkripsi saat transit. Secara default, semua komunikasi ke dan dari Timestream untuk LiveAnalytics dilindungi dengan menggunakan enkripsi Transport Layer Security (TLS).

Manajemen kunci

Anda dapat mengelola kunci untuk Amazon Timestream Live Analytics menggunakan Layanan Manajemen AWS Kunci ()AWS KMS. Timestream Live Analytics memerlukan penggunaan KMS untuk mengenkripsi data Anda. Anda memiliki opsi berikut untuk manajemen kunci, tergantung pada seberapa banyak kontrol yang Anda butuhkan atas kunci Anda:

Database dan sumber daya tabel

- Kunci yang dikelola Timestream Live Analytics: Jika Anda tidak memberikan kunci, Timestream Live Analytics akan membuat kunci menggunakan. `alias/aws/timestream` KMS

- Kunci terkelola KMS pelanggan: kunci yang dikelola pelanggan didukung. Pilih opsi ini jika Anda memerlukan kontrol lebih besar atas izin dan siklus hidup kunci Anda, termasuk kemampuan untuk memutarnya secara otomatis setiap tahun.

Sumber daya kueri terjadwal

- Kunci milik Timestream Live Analytics: Jika Anda tidak memberikan kunci, Timestream Live Analytics akan menggunakan kunci miliknya sendiri untuk mengenkripsi sumber daya Kueri, KMS kunci ini ada di akun timestream. Lihat kunci yang AWS dimiliki di panduan KMS pengembang untuk detail selengkapnya.
- Kunci terkelola KMS pelanggan: kunci yang dikelola pelanggan didukung. Pilih opsi ini jika Anda memerlukan kontrol lebih besar atas izin dan siklus hidup kunci Anda, termasuk kemampuan untuk memutarnya secara otomatis setiap tahun.

KMSkunci di penyimpanan kunci eksternal (XKS) tidak didukung.

Identitas dan manajemen akses untuk Amazon Timestream untuk LiveAnalytics

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. IAMadministrator mengontrol siapa yang dapat diautentikasi (masuk) dan diberi wewenang (memiliki izin) untuk menggunakan Timestream untuk sumber daya. LiveAnalytics IAMadalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

- Audiens
- Mengautentikasi dengan identitas
- Mengelola akses menggunakan kebijakan
- Bagaimana Amazon Timestream bekerja dengan LiveAnalytics IAM
- AWS kebijakan terkelola untuk Amazon Timestream Live Analytics
- Amazon Timestream untuk contoh kebijakan berbasis LiveAnalytics identitas
- Memecahkan masalah Amazon Timestream LiveAnalytics untuk identitas dan akses

Audiens

Bagaimana Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan di Timestream untuk LiveAnalytics.

Pengguna layanan - Jika Anda menggunakan Timestream untuk LiveAnalytics layanan untuk melakukan pekerjaan Anda, maka administrator Anda memberi Anda kredensi dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak Timestream untuk LiveAnalytics 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 Timestream untuk LiveAnalytics, lihatMemecahkan masalah Amazon Timestream LiveAnalytics untuk identitas dan akses.

Administrator layanan - Jika Anda bertanggung jawab atas Timestream untuk LiveAnalytics sumber daya di perusahaan Anda, Anda mungkin memiliki akses penuh ke Timestream untuk. LiveAnalytics Tugas Anda adalah menentukan Timestream untuk LiveAnalytics fitur dan sumber daya yang harus diakses pengguna layanan Anda. Anda kemudian harus mengirimkan permintaan ke IAM administrator Anda untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep dasarIAM. Untuk mempelajari lebih lanjut tentang bagaimana perusahaan Anda dapat menggunakan IAM Timestream LiveAnalytics, lihatBagaimana Amazon Timestream bekerja dengan LiveAnalytics IAM.

IAMadministrator - Jika Anda seorang IAM administrator, Anda mungkin ingin mempelajari detail tentang cara menulis kebijakan untuk mengelola akses ke Timestream. LiveAnalytics Untuk melihat contoh Timestream untuk kebijakan LiveAnalytics berbasis identitas yang dapat Anda gunakan, lihat. IAM Amazon Timestream untuk contoh kebijakan berbasis LiveAnalytics identitas

Mengautentikasi dengan identitas

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

Anda dapat masuk AWS sebagai identitas federasi dengan menggunakan kredensil yang disediakan melalui sumber identitas. AWS IAM Identity Center Pengguna (Pusat IAM Identitas), autentikasi masuk tunggal perusahaan Anda, dan kredensi Google atau Facebook Anda adalah contoh identitas federasi. Saat Anda masuk sebagai identitas federasi, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan IAM peran. 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 menggunakan metode yang disarankan untuk menandatangani permintaan sendiri, lihat Versi AWS Tanda Tangan 4 untuk API permintaan di Panduan IAM Pengguna.

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

Pengguna dan grup IAM

IAMPengguna adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, sebaiknya mengandalkan kredensi sementara daripada membuat IAM pengguna yang memiliki kredensi jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan khusus yang memerlukan kredensil jangka panjang dengan IAM pengguna, kami sarankan Anda memutar kunci akses. Untuk informasi selengkapnya, lihat Memutar kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensi jangka panjang di IAMPanduan Pengguna.

IAMGrup adalah identitas yang menentukan kumpulan IAM pengguna. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin bagi beberapa pengguna sekaligus. Grup mempermudah manajemen izin untuk sejumlah besar pengguna sekaligus. Misalnya, Anda dapat memiliki grup bernama IAMAdminsdan memberikan izin grup tersebut untuk mengelola sumber dayaIAM.

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

IAMperan

IAMPeran adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus. Ini mirip dengan IAM pengguna, tetapi tidak terkait dengan orang tertentu. Untuk mengambil IAM peran sementara di dalam AWS Management Console, Anda dapat beralih dari pengguna ke IAM peran (konsol). Anda dapat mengambil peran dengan memanggil AWS CLI atau AWS API operasi atau dengan menggunakan kustomURL. Untuk informasi selengkapnya tentang metode penggunaan peran, lihat Metode untuk mengambil peran dalam Panduan IAM Pengguna.

IAMperan dengan kredensyal sementara berguna dalam situasi berikut:

- Akses pengguna terfederasi – Untuk menetapkan izin ke identitas terfederasi, Anda membuat peran dan menentukan izin untuk peran tersebut. Ketika identitas terfederasi mengautentikasi, identitas tersebut terhubung dengan peran dan diberi izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat Membuat peran untuk Penyedia Identitas pihak ketiga di Panduan IAM Pengguna. Jika Anda menggunakan Pusat IAM Identitas, Anda mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah diautentikasi, Pusat IAM Identitas mengkorelasikan izin yang disetel ke peran. IAM Untuk informasi tentang set izin, lihat Set izin dalam Panduan Pengguna AWS IAM Identity Center .
- Izin IAM pengguna sementara — IAM Pengguna atau peran dapat mengambil IAM peran untuk sementara mengambil izin yang berbeda untuk tugas tertentu.
- Akses lintas akun — Anda dapat menggunakan IAM peran untuk memungkinkan seseorang (prinsipal tepercaya) di akun lain mengakses sumber daya di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapa Layanan AWS, Anda dapat melampirkan kebijakan secara langsung ke sumber daya (alih-alih menggunakan peran sebagai proxy). Untuk mempelajari perbedaan antara peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Akses sumber daya lintas akun di IAM Panduan Pengguna. IAM
- Akses lintas layanan — Beberapa Layanan AWS menggunakan fitur lain Layanan AWS. Misalnya, saat Anda melakukan panggilan dalam suatu layanan, biasanya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Sebuah layanan mungkin melakukannya menggunakan izin prinsipal yang memanggil, menggunakan peran layanan, atau peran terkait layanan.
  - Sesi akses teruskan (FAS) — Saat Anda menggunakan IAM pengguna atau peran untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FASmenggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan

hilir. FASPermintaan 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 saat membuat FAS permintaan, lihat Meneruskan sesi akses.

- Peran layanan — Peran layanan adalah IAMperan yang diasumsikan layanan untuk melakukan tindakan atas nama Anda. IAMAdministrator dapat membuat, memodifikasi, dan menghapus peran layanan dari dalamIAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS dalam IAMPanduan Pengguna.
- Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke peran layanan. Layanan AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. IAMAdministrator dapat melihat, tetapi tidak mengedit izin untuk peran terkait layanan.

- Aplikasi yang berjalan di Amazon EC2 — Anda dapat menggunakan IAM peran untuk mengelola kredenal sementara untuk aplikasi yang berjalan pada EC2 instance dan membuat AWS CLI atau AWS API meminta. Ini lebih baik untuk menyimpan kunci akses dalam EC2 instance. Untuk menetapkan AWS peran ke EC2 instance dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instance yang dilampirkan ke instance. Profil instance berisi peran dan memungkinkan program yang berjalan pada EC2 instance untuk mendapatkan kredensi sementara. Untuk informasi selengkapnya, lihat Menggunakan IAM peran untuk memberikan izin ke aplikasi yang berjalan di EC2 instans Amazon di IAMPanduan Pengguna.

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 JSON dokumen. Untuk informasi selengkapnya tentang struktur dan isi dokumen JSON kebijakan, lihat Ringkasan JSON kebijakan di Panduan IAM Pengguna.

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

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan pada sumber daya yang mereka butuhkan, IAM administrator dapat

membuat IAM kebijakan. Administrator kemudian dapat menambahkan IAM kebijakan ke peran, dan pengguna dapat mengambil peran.

IAMkebijakan menentukan izin untuk tindakan terlepas dari metode yang Anda gunakan untuk melakukan operasi. Misalnya, anggaplah Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan itu bisa mendapatkan informasi peran dari AWS Management Console, AWS CLI, atau AWS API.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan JSON izin yang dapat Anda lampirkan ke identitas, seperti pengguna, grup IAM pengguna, atau peran. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Menentukan IAM izin khusus dengan kebijakan yang dikelola pelanggan di Panduan Pengguna. IAM

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan yang dikelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran dalam. Akun AWS Kebijakan AWS terkelola mencakup kebijakan terkelola dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan terkelola atau kebijakan sebaris, lihat Memilih antara kebijakan terkelola dan kebijakan sebaris di IAMPanduan Pengguna.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen JSON kebijakan yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan IAM peran dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan prinsipal dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

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

Daftar kontrol akses (ACLs)

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

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung. ACLs Untuk mempelajari selengkapnyaACLs, lihat Ikhtisar daftar kontrol akses (ACL) di Panduan Pengembang Layanan Penyimpanan Sederhana Amazon.

Jenis-jenis kebijakan lain

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

- Batas izin — Batas izin adalah fitur lanjutan tempat Anda menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas (pengguna atau peran). IAM IAM Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya tentang batas izin, lihat Batas izin untuk IAM entitas di IAMPanduan Pengguna.
- Kebijakan kontrol layanan (SCPs) — SCPs adalah JSON kebijakan yang menentukan izin maksimum untuk organisasi atau unit organisasi (OU) di AWS Organizations. AWS Organizations adalah layanan untuk mengelompokkan dan mengelola secara terpusat beberapa Akun AWS yang dimiliki bisnis Anda. Jika Anda mengaktifkan semua fitur dalam suatu organisasi, maka Anda dapat menerapkan kebijakan kontrol layanan (SCPs) ke salah satu atau semua akun Anda. SCPMembatasi izin untuk entitas di akun anggota, termasuk masing-masing Pengguna root akun AWS. Untuk informasi selengkapnya tentang Organizations danSCPs, lihat Kebijakan kontrol layanan di Panduan AWS Organizations Pengguna.
- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda berikan sebagai parameter ketika Anda membuat sesi sementara secara programatis untuk peran atau pengguna terfederasi. Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan secara tegas dalam salah satu kebijakan ini membatalkan izin. Untuk informasi selengkapnya, lihat Kebijakan sesi di Panduan IAM Pengguna.

Berbagai jenis kebijakan

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

Bagaimana Amazon Timestream bekerja dengan LiveAnalytics IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke Timestream LiveAnalytics, Anda harus memahami IAM fitur apa yang tersedia untuk digunakan dengan Timestream. LiveAnalytics Untuk mendapatkan tampilan tingkat tinggi tentang cara kerja Timestream untuk LiveAnalytics dan AWS layanan lainnyaIAM, lihat AWS Layanan yang Bekerja dengan IAM di IAMPanduan Pengguna.

Topik

- Timestream untuk kebijakan berbasis LiveAnalytics identitas
- Timestream untuk kebijakan berbasis sumber LiveAnalytics daya
- Otorisasi berdasarkan Timestream untuk tag LiveAnalytics
- Timestream untuk peran LiveAnalytics IAM

Timestream untuk kebijakan berbasis LiveAnalytics identitas

Dengan kebijakan IAM berbasis identitas, Anda dapat menentukan tindakan dan sumber daya yang diizinkan atau ditolak serta kondisi di mana tindakan diizinkan atau ditolak. Timestream untuk LiveAnalytics mendukung tindakan dan sumber daya tertentu, dan kunci kondisi. Untuk mempelajari semua elemen yang Anda gunakan dalam JSON kebijakan, lihat Referensi Elemen IAM JSON Kebijakan di Panduan IAM Pengguna.

Tindakan

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

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

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

Anda dapat menentukan tindakan berikut dalam elemen Tindakan dari pernyataan IAM kebijakan. Gunakan kebijakan untuk memberikan izin untuk melakukan operasi diAWS. Saat Anda menggunakan tindakan dalam kebijakan, Anda biasanya mengizinkan atau menolak akses ke API operasi, CLI perintah, atau SQL perintah dengan nama yang sama.

Dalam beberapa kasus, tindakan tunggal mengontrol akses ke API operasi serta SQL perintah. Atau, beberapa operasi memerlukan beberapa tindakan yang berbeda.

Untuk daftar Timestream yang didukung untuk LiveAnalytics `Action`'s, lihat tabel di bawah ini:

> **Note**
>
> Untuk semua database spesifik`Actions`, Anda dapat menentukan database ARN untuk membatasi tindakan ke database tertentu.

Tindakan	Deskripsi	Tingkat akses	Jenis sumber daya (*diperlukan)
DescribeEndpoints	Mengembalikan titik akhir Timestream yang permintaan berikutnya harus dibuat.	Semua	*
Pilih	Jalankan kueri di Timestream yang memilih data dari satu atau beberapa tabel. Lihat catatan ini untuk penjelasan rinci	Baca	meja*
CancelQuery	Batalkan kueri.	Baca	*
ListTables	Dapatkan daftar tabel.	Daftar	basis data*

Tindakan	Deskripsi	Tingkat akses	Jenis sumber daya (*diperlukan)
ListDatabases	Dapatkan daftar database.	Daftar	*
ListMeasures	Dapatkan daftar tindakan.	Baca	meja*
DescribeTable	Dapatkan deskripsi tabel.	Baca	meja*
DescribeDatabase	Dapatkan deskripsi database.	Baca	basis data*
SelectValues	Jalankan kueri yang tidak memerlukan sumber daya tertentu untuk ditentukan. Lihat catatan ini untuk penjelasan rinci.	Baca	*
WriteRecords	Masukkan data ke Timestream.	Tulis	meja*
CreateTable	Buat tabel.	Tulis	basis data*
CreateDatabase	Buat database.	Tulis	*
DeleteDatabase	Hapus database.	Tulis	*
UpdateDatabase	Perbarui database.	Tulis	*
DeleteTable	Hapus tabel.	Tulis	basis data*
UpdateTable	Perbarui tabel.	Tulis	basis data*

SelectValues vs. pilih:

`SelectValues`adalah `Action` yang digunakan untuk kueri yang tidak memerlukan sumber daya. Contoh kueri yang tidak memerlukan sumber daya adalah sebagai berikut:

```
SELECT 1
```

Perhatikan bahwa kueri ini tidak merujuk ke Timestream tertentu untuk LiveAnalytics sumber daya. Pertimbangkan contoh lain:

```
SELECT now()
```

Kueri ini mengembalikan stempel waktu saat ini menggunakan `now()` fungsi, tetapi tidak memerlukan sumber daya yang akan ditentukan. `SelectValues`sering digunakan untuk pengujian, sehingga Timestream untuk LiveAnalytics dapat menjalankan kueri tanpa sumber daya. Sekarang, pertimbangkan `Select` kueri:

```
SELECT * FROM database.table
```

Jenis kueri ini membutuhkan sumber daya, khususnya Timestream untuk LiveAnalytics `table`, sehingga data yang ditentukan dapat diambil dari tabel.

Sumber daya

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

Elemen `Resource` JSON kebijakan menentukan objek atau objek yang tindakan tersebut berlaku. Pernyataan harus menyertakan elemen `Resource` atau `NotResource`. Sebagai praktik terbaik, 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 menunjukkan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

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

Dalam Timestream untuk LiveAnalytics database dan tabel dapat digunakan dalam `Resource` elemen izin. IAM

Timestream untuk sumber daya LiveAnalytics database memiliki yang berikut: ARN

```
arn:${Partition}:timestream:${Region}:${Account}:database/${DatabaseName}
```

Timestream untuk sumber daya LiveAnalytics tabel memiliki yang berikut: ARN

```
arn:${Partition}:timestream:${Region}:${Account}:database/${DatabaseName}/table/
${TableName}
```

Untuk informasi selengkapnya tentang formatARNs, lihat Amazon Resource Names (ARNs) dan Ruang Nama AWS Layanan.

Misalnya, untuk menentukan `database` ruang kunci dalam pernyataan Anda, gunakan yang berikut ini: ARN

```
"Resource": "arn:aws:timestream:us-east-1:123456789012:database/mydatabase"
```

Untuk menentukan semua database milik akun tertentu, gunakan wildcard (*):

```
"Resource": "arn:aws:timestream:us-east-1:123456789012:database/*"
```

Beberapa Timestream untuk LiveAnalytics tindakan, seperti untuk membuat sumber daya, tidak dapat dilakukan pada sumber daya tertentu. Dalam kasus tersebut, Anda harus menggunakan wildcard (*).

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

Kunci syarat

Timestream for LiveAnalytics tidak menyediakan kunci kondisi khusus layanan apa pun, tetapi mendukung penggunaan beberapa kunci kondisi global. Untuk melihat semua kunci kondisi AWS global, lihat Kunci Konteks Kondisi AWS Global di Panduan IAM Pengguna.

Contoh

Untuk melihat contoh Timestream untuk kebijakan LiveAnalytics berbasis identitas, lihat. Amazon Timestream untuk contoh kebijakan berbasis LiveAnalytics identitas

Timestream untuk kebijakan berbasis sumber LiveAnalytics daya

Timestream for LiveAnalytics tidak mendukung kebijakan berbasis sumber daya. Untuk melihat contoh halaman detail kebijakan berbasis sumber daya, lihat https://docs.aws.amazon.com/lambda/latest/dg/access-control-resource-based.html.

Otorisasi berdasarkan Timestream untuk tag LiveAnalytics

Anda dapat mengelola akses ke Timestream untuk LiveAnalytics sumber daya dengan menggunakan tag. Untuk mengelola akses sumber daya berdasarkan tag, Anda memberikan informasi tag dalam elemen kondisi kebijakan menggunakan kuncitimestream:ResourceTag/*key-name*,aws:RequestTag/*key-name*, atau aws:TagKeys kondisi. Untuk informasi selengkapnya tentang menandai Timestream untuk LiveAnalytics sumber daya, lihat. the section called “Pemberian tag pada sumber daya ”

Untuk melihat contoh kebijakan-kebijakan berbasis identitas untuk membatasi akses ke sumber daya berdasarkan tanda pada sumber daya tersebut, lihat Timestream untuk akses LiveAnalytics sumber daya berdasarkan tag.

Timestream untuk peran LiveAnalytics IAM

IAMPeran adalah entitas dalam AWS akun Anda yang memiliki izin khusus.

Menggunakan kredensil sementara dengan Timestream untuk LiveAnalytics

Anda dapat menggunakan kredensi sementara untuk masuk dengan federasi, mengambil IAM peran, atau untuk mengambil peran lintas akun. Anda memperoleh kredensi keamanan sementara dengan memanggil AWS STS API operasi seperti AssumeRoleatau. GetFederationToken

Peran terkait layanan

Timestream for LiveAnalytics tidak mendukung peran terkait layanan.

Peran layanan

Timestream for LiveAnalytics tidak mendukung peran layanan.

AWS kebijakan terkelola untuk Amazon Timestream Live Analytics

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 API operasi baru tersedia untuk layanan yang ada.

Untuk informasi selengkapnya, lihat kebijakan AWS terkelola di Panduan IAM Pengguna.

Topik

- AWS kebijakan terkelola: AmazonTimestreamReadOnlyAccess
- AWS kebijakan terkelola: AmazonTimestreamConsoleFullAccess
- AWS kebijakan terkelola: AmazonTimestreamFullAccess
- Pembaruan Timestream Live Analytics ke kebijakan AWS terkelola

AWS kebijakan terkelola: AmazonTimestreamReadOnlyAccess

Anda dapat melampirkan `AmazonTimestreamReadOnlyAccess` ke pengguna, grup, dan peran Anda. Kebijakan ini menyediakan akses hanya-baca ke Amazon Timestream.

Detail izin

Kebijakan ini mencakup izin berikut:

- `Amazon Timestream`— Menyediakan akses hanya-baca ke Amazon Timestream. Kebijakan ini juga memberikan izin untuk membatalkan kueri yang sedang berjalan.

Untuk meninjau kebijakan ini dalam JSON format, lihat AmazonTimestreamReadOnlyAccess.

AWS kebijakan terkelola: AmazonTimestreamConsoleFullAccess

Anda dapat melampirkan `AmazonTimestreamConsoleFullAccess` ke pengguna, grup, dan peran Anda.

Kebijakan ini menyediakan akses penuh untuk mengelola Amazon Timestream menggunakan. AWS Management Console Kebijakan ini juga memberikan izin untuk AWS KMS operasi dan operasi tertentu untuk mengelola kueri yang disimpan.

Detail izin

Kebijakan ini mencakup izin berikut:

- `Amazon Timestream`— Memberikan prinsipal akses penuh ke Amazon Timestream.
- `AWS KMS`— Memungkinkan kepala sekolah untuk membuat daftar alias dan mendeskripsikan kunci.
- `Amazon S3`— Memungkinkan kepala sekolah untuk membuat daftar semua ember Amazon S3.
- `Amazon SNS`— Memungkinkan kepala sekolah untuk membuat daftar topik Amazon. SNS
- `IAM`— Memungkinkan kepala sekolah untuk daftar peran. IAM
- `DBQMS` – Mengizinkan pengguna utama mengakses, menghapus, mendeskripsikan, dan memperbarui kueri. Database Query Metadata Service (dbqms) adalah layanan internal saja. Ini memberikan kueri terbaru dan tersimpan Anda untuk editor kueri di AWS Management Console untuk beberapa Layanan AWS, termasuk Amazon Timestream.

Untuk meninjau kebijakan ini dalam JSON format, lihat AmazonTimestreamConsoleFullAccess.

AWS kebijakan terkelola: AmazonTimestreamFullAccess

Anda dapat melampirkan `AmazonTimestreamFullAccess` ke pengguna, grup, dan peran Anda.

Kebijakan ini menyediakan akses penuh ke Amazon Timestream. Kebijakan ini juga memberikan izin untuk operasi tertentu AWS KMS .

Detail izin

Kebijakan ini mencakup izin berikut:

- `Amazon Timestream`— Memberikan prinsipal akses penuh ke Amazon Timestream.
- `AWS KMS`— Memungkinkan kepala sekolah untuk membuat daftar alias dan mendeskripsikan kunci.
- `Amazon S3`— Memungkinkan kepala sekolah untuk membuat daftar semua ember Amazon S3.

Untuk meninjau kebijakan ini dalam JSON format, lihat AmazonTimestreamFullAccess.

Pembaruan Timestream Live Analytics ke kebijakan AWS terkelola

Lihat detail tentang pembaruan kebijakan AWS terkelola untuk Timestream Live Analytics sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan RSS feed di halaman riwayat Dokumen Timestream Live Analytics.

Perubahan	Deskripsi	Tanggal
AmazonTimestreamReadOnlyAccess — Permbaruan ke kebijakan yang sudah ada	Menambahkan `timestream:DescribeAccountSettings` tindakan ke kebijakan `AmazonTimestreamReadOnlyAccess` terkelola yang ada. Tindakan ini digunakan untuk menggambarkan Akun AWS pengaturan. Timestream Live Analytics juga telah memperbarui kebijakan terkelola ini dengan menambahkan `Sid` bidang. Pembaruan kebijakan tidak memengaruhi penggunaan kebijakan `AmazonTimestreamReadOnlyAccess` terkelola.	Juni 03, 2024

Perubahan	Deskripsi	Tanggal
AmazonTimestreamReadOnlyAccess – Pembaruan ke kebijakan yang ada	Menambahkan `timestream:DescribeBatchLoadTask` dan `timestream:ListBatchLoadTasks` tindakan ke kebijakan `AmazonTimestreamReadOnlyAccess` terkelola yang ada. Tindakan ini digunakan saat mencantumkan dan menjelaskan tugas pemuatan batch. Pembaruan kebijakan tidak memengaruhi penggunaan kebijakan `AmazonTimestreamReadOnlyAccess` terkelola.	Februari 24, 2023
AmazonTimestreamReadOnlyAccess – Pembaruan ke kebijakan yang ada	Menambahkan `timestream:DescribeScheduledQuery` dan `timestream:ListScheduledQueries` tindakan ke kebijakan `AmazonTimestreamReadOnlyAccess` terkelola yang ada. Tindakan ini digunakan saat mencantumkan dan menjelaskan kueri terjadwal yang ada. Pembaruan kebijakan tidak memengaruhi penggunaan kebijakan `AmazonTimestreamReadOnlyAccess` terkelola.	29 November 2021

Perubahan	Deskripsi	Tanggal
AmazonTimestreamConsoleFullAccess – Pembaruan ke kebijakan yang ada	Menambahkan `s3:ListAllMyBuckets` tindakan ke kebijakan `AmazonTimestreamConsoleFullAccess` terkelola yang ada. Tindakan ini digunakan saat Anda menentukan bucket Amazon S3 untuk Timestream untuk mencatat kesalahan penulisan penyimpanan magnetik. Pembaruan kebijakan tidak memengaruhi penggunaan kebijakan `AmazonTimestreamConsoleFullAccess` terkelola.	29 November 2021
AmazonTimestreamFullAccess – Pembaruan ke kebijakan yang ada	Menambahkan `s3:ListAllMyBuckets` tindakan ke kebijakan `AmazonTimestreamFullAccess` terkelola yang ada. Tindakan ini digunakan saat Anda menentukan bucket Amazon S3 untuk Timestream untuk mencatat kesalahan penulisan penyimpanan magnetik. Pembaruan kebijakan tidak memengaruhi penggunaan kebijakan `AmazonTimestreamFullAccess` terkelola.	29 November 2021

Perubahan	Deskripsi	Tanggal
AmazonTimestreamConsoleFullAccess – Pembaruan ke kebijakan yang ada	Menghapus tindakan berlebihan dari kebijakan `AmazonTimestreamConsoleFullAccess` terkelola yang ada. Sebelumnya, kebijakan ini termasuk tindakan yang berlebihan. `dbqms:DescribeQueryHistory` Kebijakan yang diperbarui akan menghapus tindakan yang berlebihan. Pembaruan kebijakan tidak memengaruhi penggunaan kebijakan `AmazonTimestreamConsoleFullAccess` terkelola.	23 April 2021
Timestream Live Analytics mulai melacak perubahan	Timestream Live Analytics mulai melacak perubahan untuk kebijakan yang AWS dikelola.	21 April 2021

Amazon Timestream untuk contoh kebijakan berbasis LiveAnalytics identitas

Secara default, IAM pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi Timestream untuk LiveAnalytics sumber daya. Mereka juga tidak dapat melakukan tugas menggunakan AWS Management Console,CQLSH, AWS CLI, atau AWS API. IAMAdministrator harus membuat IAM kebijakan yang memberikan izin kepada pengguna dan peran untuk melakukan API operasi tertentu pada sumber daya tertentu yang mereka butuhkan. Administrator kemudian harus melampirkan kebijakan tersebut ke IAM pengguna atau grup yang memerlukan izin tersebut.

Untuk mempelajari cara membuat kebijakan IAM berbasis identitas menggunakan contoh dokumen kebijakan ini, lihat Membuat JSON Kebijakan di JSON Tab di Panduan Pengguna. IAM

Topik

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus Timestream untuk LiveAnalytics sumber daya di akun Anda. Tindakan ini membuat Akun AWS Anda dikenai biaya. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulailah dengan kebijakan AWS terkelola dan beralih ke izin hak istimewa paling sedikit — Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan AWS terkelola yang memberikan izin untuk banyak kasus penggunaan umum. Mereka tersedia di Anda Akun AWS. Kami menyarankan Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola AWS pelanggan yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat kebijakan AWSAWS terkelola atau kebijakan terkelola untuk fungsi pekerjaan di Panduan IAM Pengguna.
- Menerapkan izin hak istimewa paling sedikit — Saat Anda menetapkan izin dengan IAM kebijakan, berikan hanya izin yang diperlukan untuk melakukan tugas. Anda melakukannya dengan mendefinisikan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, yang juga dikenal sebagai izin dengan hak akses paling rendah. Untuk informasi selengkapnya tentang penggunaan IAM untuk menerapkan izin, lihat Kebijakan dan izin IAM di IAM Panduan Pengguna.
- Gunakan ketentuan dalam IAM kebijakan untuk membatasi akses lebih lanjut — Anda dapat menambahkan kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Misalnya, Anda dapat menulis kondisi kebijakan untuk menentukan bahwa semua permintaan harus dikirim menggunakanSSL. 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 IAM JSON kebijakan: Kondisi dalam Panduan IAM Pengguna.

- Gunakan IAM Access Analyzer untuk memvalidasi IAM kebijakan Anda guna memastikan izin yang aman dan fungsional — IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa IAM kebijakan () JSON dan praktik terbaik. IAM IAMAccess 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 Memvalidasi kebijakan dengan IAM Access Analyzer di IAMPanduan Pengguna.
- Memerlukan otentikasi multi-faktor (MFA) - Jika Anda memiliki skenario yang mengharuskan IAM pengguna atau pengguna root di dalam Anda Akun AWS, aktifkan MFA untuk keamanan tambahan. Untuk meminta MFA kapan API operasi dipanggil, tambahkan MFA kondisi ke kebijakan Anda. Untuk informasi selengkapnya, lihat APIAkses aman dengan MFA di Panduan IAM Pengguna.

Untuk informasi selengkapnya tentang praktik terbaik diIAM, lihat Praktik terbaik keamanan IAM di Panduan IAM Pengguna.

Menggunakan Timestream untuk konsol LiveAnalytics

Timestream for LiveAnalytics tidak memerlukan izin khusus untuk mengakses Amazon LiveAnalytics Timestream untuk konsol. Anda memerlukan setidaknya izin hanya-baca untuk membuat daftar dan melihat detail tentang Timestream untuk LiveAnalytics sumber daya di akun Anda. AWS Jika Anda membuat kebijakan berbasis identitas yang lebih ketat daripada izin minimum yang diperlukan, konsol tidak akan berfungsi sebagaimana dimaksud untuk entitas (IAMpengguna atau peran) dengan kebijakan tersebut.

Mengizinkan pengguna melihat izin mereka sendiri

Contoh ini menunjukkan cara Anda membuat kebijakan yang memungkinkan IAM pengguna melihat kebijakan sebaris dan terkelola yang dilampirkan pada identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan ini di konsol atau secara terprogram menggunakan atau. AWS CLI AWS API

```
{
    "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": "*"
        }
    ]
}
```

Operasi umum di Timestream untuk LiveAnalytics

Di bawah ini adalah contoh IAM kebijakan yang memungkinkan operasi umum di Timestream untuk LiveAnalytics layanan.

Topik

- Mengizinkan semua operasi
- Mengizinkan SELECT operasi
- Mengizinkan SELECT operasi pada berbagai sumber daya
- Mengizinkan operasi metadata
- Mengizinkan INSERT operasi
- Mengizinkan CRUD operasi
- Batalkan kueri dan pilih data tanpa menentukan sumber daya
- Membuat, mendeskripsikan, menghapus, dan mendeskripsikan database

- Batasi database yang terdaftar dengan tag {"Owner": "${username}"}
- Daftar semua tabel dalam database
- Buat, jelaskan, hapus, perbarui, dan pilih di atas meja
- Batasi kueri berdasarkan tabel

Mengizinkan semua operasi

Berikut ini adalah contoh kebijakan yang memungkinkan semua operasi di Timestream untuk LiveAnalytics.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Mengizinkan SELECT operasi

Kebijakan sampel berikut memungkinkan kueri `SELECT` -style pada sumber daya tertentu.

> **Note**
>
> Ganti `<account_ID>` dengan ID akun Amazon Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream:Select",
                "timestream:DescribeTable",
```

```
                "timestream:ListMeasures"
            ],
            "Resource": "arn:aws:timestream:us-east-1:<account_ID>:database/sampleDB/
table/DevOps"
        },
        {
            "Effect": "Allow",
            "Action": [
                "timestream:DescribeEndpoints",
                "timestream:SelectValues",
                "timestream:CancelQuery"
            ],
            "Resource": "*"
        }
    ]
}
```

Mengizinkan SELECT operasi pada berbagai sumber daya

Kebijakan sampel berikut memungkinkan kueri SELECT -style pada beberapa sumber daya.

> **Note**
>
> Ganti <account_ID> dengan ID akun Amazon Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream:Select",
                "timestream:DescribeTable",
                "timestream:ListMeasures"
            ],
            "Resource": [
                "arn:aws:timestream:us-east-1:<account_ID>:database/sampleDB/table/
DevOps",
                "arn:aws:timestream:us-east-1:<account_ID>:database/sampleDB/table/
DevOps1",
                "arn:aws:timestream:us-east-1:<account_ID>:database/sampleDB/table/
DevOps2"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "timestream:DescribeEndpoints",
                "timestream:SelectValues",
                "timestream:CancelQuery"
            ],
            "Resource": "*"
        }
    ]
}
```

Mengizinkan operasi metadata

Kebijakan contoh berikut memungkinkan pengguna untuk melakukan kueri metadata, tetapi tidak mengizinkan pengguna untuk melakukan operasi yang membaca atau menulis data aktual di Timestream for. LiveAnalytics

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream:DescribeEndpoints",
                "timestream:DescribeTable",
                "timestream:ListMeasures",
                "timestream:SelectValues",
                "timestream:ListTables",
                "timestream:ListDatabases",
                "timestream:CancelQuery"
            ],
            "Resource": "*"
        }
    ]
}
```

Mengizinkan INSERT operasi

Contoh kebijakan berikut memungkinkan pengguna untuk melakukan `INSERT` operasi di `database/sampleDB/table/DevOps` dalam akun`<account_id>`.

> **Note**
>
> Ganti <account_ID> dengan ID akun Amazon Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "timestream:WriteRecords"
            ],
            "Resource": [
                "arn:aws:timestream:us-east-1:<account_id>:database/sampleDB/table/
DevOps"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "timestream:DescribeEndpoints"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

Mengizinkan CRUD operasi

Kebijakan contoh berikut memungkinkan pengguna untuk melakukan CRUD operasi di Timestream untuk LiveAnalytics.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream:DescribeEndpoints",
                "timestream:CreateTable",
                "timestream:DescribeTable",
```

```
                "timestream:CreateDatabase",
                "timestream:DescribeDatabase",
                "timestream:ListTables",
                "timestream:ListDatabases",
                "timestream:DeleteTable",
                "timestream:DeleteDatabase",
                "timestream:UpdateTable",
                "timestream:UpdateDatabase"
            ],
            "Resource": "*"
        }
    ]
}
```

Batalkan kueri dan pilih data tanpa menentukan sumber daya

Kebijakan contoh berikut memungkinkan pengguna untuk membatalkan kueri dan melakukan `Select` kueri pada data yang tidak memerlukan spesifikasi sumber daya:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream:SelectValues",
                "timestream:CancelQuery"
            ],
            "Resource": "*"
        }
    ]
}
```

Membuat, mendeskripsikan, menghapus, dan mendeskripsikan database

Kebijakan contoh berikut memungkinkan pengguna untuk membuat, mendeskripsikan, menghapus, dan mendeskripsikan database`sampleDB`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "timestream:CreateDatabase",
                "timestream:DescribeDatabase",
                "timestream:DeleteDatabase",
                "timestream:UpdateDatabase"
            ],
            "Resource": "arn:aws:timestream:us-east-1:<account_ID>:database/sampleDB"
        }
    ]
}
```

Batasi database yang terdaftar dengan tag **`{"Owner": "${username}"}`**

Kebijakan sampel berikut memungkinkan pengguna untuk mencantumkan semua database yang ditandai dengan pasangan nilai kunci: `{"Owner": "${username}"}`

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream:ListDatabases"
            ],
            "Resource": "arn:aws:timestream:us-east-1:<account_ID>:database/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Owner": "${aws:username}"
                }
            }
        }
    ]
}
```

Daftar semua tabel dalam database

Contoh kebijakan berikut untuk daftar semua tabel dalam database`sampleDB`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "timestream:ListTables"
            ],
            "Resource": "arn:aws:timestream:us-east-1:<account_ID>:database/sampleDB/"
        }
    ]
}
```

Buat, jelaskan, hapus, perbarui, dan pilih di atas meja

Kebijakan contoh berikut memungkinkan pengguna untuk membuat tabel, mendeskripsikan tabel, menghapus tabel, memperbarui tabel, dan melakukan `Select` kueri pada tabel `DevOps` dalam database`sampleDB`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream:CreateTable",
                "timestream:DescribeTable",
                "timestream:DeleteTable",
                "timestream:UpdateTable",
                "timestream:Select"
            ],
            "Resource": "arn:aws:timestream:us-east-1:<account_ID>:database/sampleDB/
table/DevOps"
        }
    ]
}
```

Batasi kueri berdasarkan tabel

Kebijakan contoh berikut memungkinkan pengguna untuk menanyakan semua tabel kecuali `DevOps` dalam database`sampleDB`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "timestream:Select"
            ],
            "Resource": "arn:aws:timestream:us-east-1:<account_ID>:database/sampleDB/
table/*"
        },
        {
            "Effect": "Deny",
            "Action": [
                "timestream:Select"
            ],
            "Resource": "arn:aws:timestream:us-east-1:<account_ID>:database/sampleDB/
table/DevOps"
        }
    ]
}
```

Timestream untuk akses LiveAnalytics sumber daya berdasarkan tag

Anda dapat menggunakan kondisi dalam kebijakan berbasis identitas untuk mengontrol akses ke Timestream untuk LiveAnalytics sumber daya berdasarkan tag. Bagian ini menyediakan beberapa contoh.

Contoh berikut menunjukkan cara membuat kebijakan yang memberikan izin kepada pengguna untuk melihat tabel jika tabel `Owner` berisi nilai nama pengguna tersebut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadOnlyAccessTaggedTables",
            "Effect": "Allow",
            "Action": "timestream:Select",
            "Resource": "arn:aws:timestream:us-east-2:111122223333:database/mydatabase/
table/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Owner": "${aws:username}"
                }
            }
        }
    ]
}
```

Anda dapat melampirkan kebijakan ini ke IAM pengguna di akun Anda. Jika pengguna bernama `richard-roe` mencoba untuk melihat Timestream untuk LiveAnalytics tabel, tabel harus diberi tag `Owner=richard-roe` atau. `owner=richard-roe` Jika tidak, aksesnya akan ditolak. Kunci tanda syarat `Owner` sama dengan kedua `Owner` dan `owner` karena nama kunci syarat tidak terpengaruh huruf besar/kecil. Untuk informasi selengkapnya, lihat Elemen IAM JSON Kebijakan: Kondisi dalam Panduan IAM Pengguna.

Kebijakan berikut memberikan izin kepada pengguna untuk membuat tabel dengan tag jika tag yang diteruskan dalam permintaan memiliki kunci `Owner` dan nilai: `username`

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateTagTableUser",
            "Effect": "Allow",
            "Action": [
                "timestream:Create",
                "timestream:TagResource"
            ],
            "Resource": "arn:aws:timestream:us-east-2:111122223333:database/mydatabase/
table/*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:RequestTag/Owner": "${aws:username}"
                }
            }
        }
    ]
}
```

Kebijakan di bawah ini memungkinkan penggunaan `DescribeDatabase` API pada Database apa pun yang memiliki `env` tag yang disetel ke salah satu `dev` atau`test`:

```
{ "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowDescribeEndpoints",
      "Effect": "Allow",
      "Action": [
        "timestream:DescribeEndpoints"
      ],
```

```
      "Resource": "*"
    },
    {
      "Sid": "AllowDevTestAccess",
      "Effect": "Allow",
      "Action": [
        "timestream:DescribeDatabase"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "timestream:tag/env": [
              "dev",
              "test"
          ]
        }
      }
    }
  ]
}
{ "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowTagAccessForDevResources",
      "Effect": "Allow",
      "Action": [
        "timestream:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/env": [
            "test",
            "dev"
          ]
        }
      }
    }
  ]
}
```

Kebijakan ini menggunakan `Condition` kunci untuk mengizinkan tag yang memiliki kunci `env` dan nilai`test`,`qa`, atau ditambahkan `dev` ke sumber daya.

Pertanyaan terjadwal

Daftar, hapus, perbarui, jalankan ScheduledQuery

Kebijakan contoh berikut memungkinkan pengguna untuk membuat daftar, menghapus, memperbarui, dan mengeksekusi kueri terjadwal.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream:DeleteScheduledQuery",
                "timestream:ExecuteScheduledQuery",
                "timestream:UpdateScheduledQuery",
                "timestream:ListScheduledQueries",
                "timestream:DescribeEndpoints"
            ],
            "Resource": "*"
        }
    ]
}
```

CreateScheduledQuery menggunakan KMS kunci yang dikelola pelanggan

Kebijakan contoh berikut memungkinkan pengguna untuk membuat kueri terjadwal yang dienkripsi menggunakan kunci terkelola KMS pelanggan; *<keyid for ScheduledQuery>*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/ScheduledQueryExecutionRole"
            ],
            "Effect": "Allow"
        },
        {
```

```
            "Action": [
                "timestream:CreateScheduledQuery",
                "timestream:DescribeEndpoints"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:us-west-2:123456789012:key/<keyid for
 ScheduledQuery>",
            "Effect": "Allow"
        }
    ]
}
```

DescribeScheduledQuery menggunakan KMS kunci yang dikelola pelanggan

Kebijakan contoh berikut memungkinkan pengguna untuk mendeskripsikan kueri terjadwal yang dibuat menggunakan KMS kunci yang dikelola pelanggan; *<keyid for ScheduledQuery>*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "timestream:DescribeScheduledQuery",
                "timestream:DescribeEndpoints"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:us-west-2:123456789012:key/<keyid for
 ScheduledQuery>",
            "Effect": "Allow"
        }
```

```
    ]
}
```

Izin peran eksekusi (menggunakan KMS kunci terkelola pelanggan untuk kueri terjadwal dan SSE - KMS untuk laporan kesalahan)

Lampirkan kebijakan contoh berikut ke IAM peran yang ditentukan dalam `ScheduledQueryExecutionRoleArn` parameter, `CreateScheduledQuery` API yang menggunakan KMS kunci terkelola pelanggan untuk enkripsi kueri terjadwal dan `SSE-KMS` enkripsi untuk laporan kesalahan.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "kms:GenerateDataKey",
            ],
            "Resource": "arn:aws:kms:us-west-2:123456789012:key/<keyid for
 ScheduledQuery>",
            "Effect": "Allow"
        },
        {
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:kms:us-west-2:123456789012:key/<keyid for database-1>",
                "arn:aws:kms:us-west-2:123456789012:key/<keyid for database-n>",
                "arn:aws:kms:us-west-2:123456789012:key/<keyid for ScheduledQuery>"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:us-west-2:123456789012:scheduled-query-notification-topic-
*"
            ],
            "Effect": "Allow"
        },
```

```
        {
            "Action": [
                "timestream:Select",
                "timestream:SelectValues",
                "timestream:WriteRecords"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:PutObject",
                "s3:GetBucketAcl"
            ],
            "Resource": [
                "arn:aws:s3:::scheduled-query-error-bucket",
                "arn:aws:s3:::scheduled-query-error-bucket/*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Hubungan kepercayaan peran eksekusi

Berikut ini adalah hubungan kepercayaan untuk IAM peran yang ditentukan dalam `ScheduledQueryExecutionRoleArn` parameter `CreateScheduledQuery`API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "timestream.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Izinkan akses ke semua kueri terjadwal yang dibuat dalam akun

Lampirkan kebijakan contoh berikut ke IAM peran yang ditentukan dalam `ScheduledQueryExecutionRoleArn` parameter, dari `CreateScheduledQuery`API, untuk mengizinkan akses ke semua kueri terjadwal yang dibuat dalam akun *`Account_ID`*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "timestream.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "Account_ID"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:timestream:us-
west-2:Account_ID:scheduled-query/*"
                }
            }
        }
    ]
}
```

Izinkan akses ke semua kueri terjadwal dengan nama tertentu

Lampirkan kebijakan sampel berikut ke IAM peran yang ditentukan dalam `ScheduledQueryExecutionRoleArn` parameter, dari `CreateScheduledQuery`API, untuk mengizinkan akses ke semua kueri terjadwal dengan nama yang dimulai dengan *`Scheduled_Query_Name`*, dalam akun *`Account_ID`*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "timestream.amazonaws.com"
```

```
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "Account_ID"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:timestream:us-
west-2:Account_ID:scheduled-query/Scheduled_Query_Name*"
                }
            }
        }
    ]
}
```

Memecahkan masalah Amazon Timestream LiveAnalytics untuk identitas dan akses

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan Timestream untuk LiveAnalytics danIAM.

Topik

- Saya tidak berwenang untuk melakukan tindakan di Timestream untuk LiveAnalytics
- Saya tidak berwenang untuk melakukan iam: PassRole
- Saya ingin mengizinkan orang di luar AWS akun saya untuk mengakses Timestream saya untuk sumber daya LiveAnalytics

Saya tidak berwenang untuk melakukan tindakan di Timestream untuk LiveAnalytics

Jika AWS Management Console memberitahu Anda bahwa Anda tidak berwenang untuk melakukan suatu tindakan, maka Anda harus menghubungi administrator Anda untuk bantuan. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Contoh kesalahan berikut terjadi ketika `mateojackson` IAM pengguna mencoba menggunakan konsol untuk melihat detail tentang *`table`* tetapi tidak memiliki `timestream:`*`Select`* izin untuk tabel.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 timestream:Select on resource: mytable
```

Dalam hal ini, Mateo meminta administratornya untuk memperbarui kebijakannya untuk mengizinkan dia mengakses sumber daya *mytable* menggunakan tindakan `timestream:Select`.

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 ke Timestream. LiveAnalytics

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 IAM pengguna bernama `marymajor` mencoba menggunakan konsol untuk melakukan tindakan di Timestream for LiveAnalytics. 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 AWS akun saya untuk mengakses Timestream saya untuk sumber daya LiveAnalytics

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACLs), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa referensi berikut:

- Untuk mempelajari apakah Timestream untuk LiveAnalytics mendukung fitur-fitur ini, lihatBagaimana Amazon Timestream bekerja dengan LiveAnalytics IAM.

- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh sumber daya Akun AWS yang Anda miliki, lihat Menyediakan akses ke IAM pengguna lain Akun AWS yang Anda miliki di Panduan IAM Pengguna.
- 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 IAM Pengguna.
- Untuk mempelajari cara menyediakan akses melalui federasi identitas, lihat Menyediakan akses ke pengguna yang diautentikasi secara eksternal (federasi identitas) di Panduan Pengguna. IAM
- Untuk mempelajari perbedaan antara menggunakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Akses sumber daya lintas akun di IAM Panduan Pengguna. IAM

Pencatatan dan pemantauan di Timestream untuk LiveAnalytics

Pemantauan adalah bagian penting dalam menjaga keandalan, ketersediaan, dan kinerja Timestream untuk LiveAnalytics dan AWS solusi Anda. Anda harus mengumpulkan data pemantauan dari semua bagian AWS solusi Anda sehingga Anda dapat lebih mudah men-debug kegagalan multi-titik jika terjadi. Namun, sebelum Anda mulai memantau Timestream LiveAnalytics, Anda harus membuat rencana pemantauan yang mencakup jawaban atas pertanyaan-pertanyaan berikut:

- Apa tujuan pemantauan Anda?
- Sumber daya apa yang akan Anda pantau?
- Seberapa sering Anda akan memantau sumber daya ini?
- Alat pemantauan apa yang akan Anda gunakan?
- Siapa yang akan melakukan tugas pemantauan?
- Siapa yang harus diberi tahu saat terjadi kesalahan?

Langkah selanjutnya adalah menetapkan garis dasar untuk Timestream normal untuk LiveAnalytics kinerja di lingkungan Anda, dengan mengukur kinerja pada berbagai waktu dan dalam kondisi beban yang berbeda. Saat Anda memantau Timestream LiveAnalytics, simpan data pemantauan historis sehingga Anda dapat membandingkannya dengan data kinerja saat ini, mengidentifikasi pola kinerja normal dan anomali kinerja, dan merancang metode untuk mengatasi masalah.

Untuk menetapkan baseline, Anda harus, setidaknya, memantau item-item berikut:

- Kesalahan sistem, sehingga Anda dapat menentukan apakah ada permintaan yang mengakibatkan kesalahan.

Topik

Alat pemantauan

AWS menyediakan berbagai alat yang dapat Anda gunakan untuk memantau Timestream. LiveAnalytics Anda dapat mengonfigurasi beberapa alat ini untuk melakukan pemantauan untuk Anda, sementara beberapa alat memerlukan intervensi manual. Kami menyarankan agar Anda mengautomasi tugas pemantauan sebanyak mungkin.

Topik

Alat pemantauan otomatis

Anda dapat menggunakan alat pemantauan otomatis berikut untuk menonton Timestream LiveAnalytics dan melaporkan ketika ada sesuatu yang salah:

- CloudWatch Alarm Amazon — Tonton satu metrik selama periode waktu yang Anda tentukan, dan lakukan satu atau beberapa tindakan berdasarkan nilai metrik relatif terhadap ambang batas tertentu selama beberapa periode waktu. Tindakannya adalah pemberitahuan yang dikirim ke topik Amazon Simple Notification Service (AmazonSNS) atau kebijakan Amazon EC2 Auto Scaling. CloudWatch alarm tidak memanggil tindakan hanya karena mereka berada dalam keadaan tertentu; negara harus telah berubah dan dipertahankan untuk sejumlah periode tertentu. Untuk informasi selengkapnya, lihat Pemantauan CloudWatch dengan Amazon.

Alat pemantauan manual

Bagian penting lainnya dari pemantauan Timestream LiveAnalytics melibatkan pemantauan secara manual item yang tidak CloudWatch tercakup oleh alarm. Timestream untuk LiveAnalytics

CloudWatch, Trusted Advisor,, dan AWS Management Console dasbor lainnya memberikan at-a-glance tampilan keadaan lingkungan Anda AWS .

- CloudWatch Halaman beranda menunjukkan yang berikut:
  - Alarm dan status saat ini
  - Grafik alarm dan sumber daya
  - Status kesehatan layanan

  Selain itu, Anda dapat menggunakan CloudWatch untuk melakukan hal berikut:
  - Membuat dasbor yang disesuaikan untuk memantau layanan yang penting bagi Anda
  - Data metrik grafik untuk memecahkan masalah dan mengungkap tren
  - Cari dan telusuri semua metrik AWS sumber daya Anda
  - Membuat dan mengedit alarm untuk menerima notifikasi terkait masalah

Logging Timestream untuk LiveAnalytics API panggilan dengan AWS CloudTrail

Timestream for LiveAnalytics terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di Timestream untuk. LiveAnalytics CloudTrail menangkap API panggilan Data Definition Language (DDL) untuk Timestream untuk LiveAnalytics sebagai peristiwa. Panggilan yang ditangkap termasuk panggilan dari Timestream untuk LiveAnalytics konsol dan panggilan kode ke Timestream untuk LiveAnalytics API operasi. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail peristiwa secara terus menerus ke bucket Amazon Simple Storage Service (Amazon S3), termasuk peristiwa untuk Timestream for. LiveAnalytics 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 ke Timestream untuk LiveAnalytics, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari selengkapnya CloudTrail, lihat Panduan AWS CloudTrail Pengguna.

Timestream untuk LiveAnalytics informasi di CloudTrail

CloudTrail diaktifkan di AWS akun Anda saat Anda membuat akun. Ketika aktivitas terjadi di Timestream for LiveAnalytics, aktivitas tersebut direkam dalam suatu CloudTrail peristiwa bersama dengan peristiwa AWS layanan lainnya dalam riwayat Peristiwa. Anda dapat melihat, mencari, dan

mengunduh peristiwa terbaru di akun AWS . Untuk informasi selengkapnya, lihat Melihat Acara dengan Riwayat CloudTrail Acara.

> **Warning**
>
> Saat ini, Timestream untuk LiveAnalytics menghasilkan CloudTrail peristiwa untuk semua manajemen dan `Query` API operasi, tetapi tidak menghasilkan peristiwa untuk `WriteRecords` dan `DescribeEndpoints`APIs.

Untuk catatan peristiwa yang sedang berlangsung di AWS akun Anda, termasuk acara untuk Timestream LiveAnalytics, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon 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 Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log.

Untuk informasi selengkapnya, lihat topik berikut di Panduan Pengguna AWS CloudTrail :

- Gambaran Umum untuk Membuat Jejak
- CloudTrail Layanan dan Integrasi yang Didukung
- Mengkonfigurasi SNS Pemberitahuan Amazon untuk CloudTrail
- Menerima File CloudTrail Log dari Beberapa Wilayah
- Menerima File CloudTrail Log dari Beberapa Akun
- Pencatatan peristiwa data

Setiap entri peristiwa atau log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan berikut ini:

- Apakah permintaan dibuat dengan root atau AWS Identity and Access Management (IAM) kredensyal pengguna
- Baik permintaan tersebut dibuat dengan kredensial keamanan sementara untuk peran atau pengguna gabungan
- Apakah permintaan itu dibuat oleh AWS layanan lain

Untuk informasi selengkapnya, lihat CloudTrail userIdentityElemen.

Untuk `Query` API acara:

- Buat jejak yang menerima semua acara atau memilih acara dengan Timestream untuk jenis LiveAnalytics sumber daya `AWS::Timestream::Database` atau`AWS::Timestream::Table`.
- `Query`APIpermintaan yang tidak mengakses database atau tabel apa pun atau yang menghasilkan pengecualian validasi karena string kueri cacat dicatat CloudTrail dengan jenis sumber daya `AWS::Timestream::Database` dan ARN nilai:

  ```
  arn:aws:timestream:(region):(accountId):database/NO_RESOURCE_ACCESSED
  ```

  Peristiwa ini dikirimkan hanya ke jalur yang menerima acara dengan tipe `AWS::Timestream::Database` sumber daya.

Ketahanan di Amazon Timestream Live Analytics

Infrastruktur AWS global dibangun di sekitar AWS Wilayah dan Zona Ketersediaan. AWS Wilayah menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang 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 AWS Wilayah dan Availability Zone, lihat Infrastruktur AWS Global.

Untuk informasi tentang fungsionalitas perlindungan data untuk Timestream yang tersedia melalui AWS Backup, lihatBekerja dengan AWS Backup.

Keamanan infrastruktur di Amazon Timestream Live Analytics

Sebagai layanan terkelola, Amazon Timestream Live Analytics dilindungi oleh prosedur keamanan jaringan AWS global yang dijelaskan dalam whitepaper Amazon Web Services: Tinjauan Proses Keamanan.

Anda menggunakan API panggilan yang AWS dipublikasikan untuk mengakses Timestream Live Analytics melalui jaringan. Klien harus mendukung Transport Layer Security (TLS) 1.0 atau yang

lebih baru. Kami merekomendasikan TLS 1.2 atau yang lebih baru. Klien juga harus mendukung cipher suite dengan perfect forward secrecy (PFS) seperti Ephemeral Diffie-Hellman () atau Elliptic Curve Ephemeral Diffie-Hellman (). DHE 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 IAM prinsipal. Atau Anda dapat menggunakan AWS Security Token Service (AWS STS) untuk menghasilkan kredensial keamanan sementara untuk menandatangani permintaan.

Timestream Live Analytics dirancang sehingga lalu lintas Anda diisolasi ke AWS Wilayah tertentu tempat instans Timestream Live Analytics Anda berada.

Analisis konfigurasi dan kerentanan di Timestream

Konfigurasi dan kontrol TI adalah tanggung jawab bersama antara AWS dan Anda, pelanggan kami. Untuk informasi selengkapnya, lihat model tanggung jawab AWS bersama. Selain model tanggung jawab bersama, Timestream untuk LiveAnalytics pengguna harus menyadari hal-hal berikut:

- Pelanggan bertanggung jawab untuk menambalkan patch aplikasi klien mereka dengan dependensi sisi klien yang relevan.
- Pelanggan harus mempertimbangkan pengujian penetrasi jika sesuai (lihat pengujian https://aws.amazon.com/security/penetrasi/.)

Respons insiden di Timestream untuk LiveAnalytics

Amazon Timestream untuk insiden LiveAnalytics layanan dilaporkan di Dasbor Personal Health. Anda dapat mempelajari lebih lanjut tentang dasbor dan AWS Health di sini.

Timestream untuk LiveAnalytics mendukung pelaporan menggunakan AWS CloudTrail. Untuk informasi selengkapnya, lihat Logging Timestream untuk LiveAnalytics API panggilan dengan AWS CloudTrail.

VPCtitik akhir ()AWS PrivateLink

Anda dapat membuat koneksi pribadi antara Timestream Anda VPC dan Amazon LiveAnalytics dengan membuat titik akhir antarmuka VPC. Endpoint antarmuka didukung oleh AWS PrivateLink, teknologi yang memungkinkan Anda mengakses Timestream secara pribadi LiveAnalytics APIs tanpa gateway internet, NAT perangkat, koneksi, atau VPN koneksi Direct AWS Connect. Instans di Anda

VPC tidak memerlukan alamat IP publik untuk berkomunikasi dengan Timestream. LiveAnalytics APIs Lalu lintas antara Anda VPC dan Timestream untuk LiveAnalytics tidak meninggalkan jaringan Amazon.

Setiap titik akhir antarmuka diwakili oleh satu atau beberapa Antarmuka Jaringan Elastis di subnet Anda. Untuk informasi selengkapnya tentang VPC titik akhir Antarmuka, lihat VPCTitik akhir antarmuka (AWS PrivateLink) di VPCPanduan Pengguna Amazon.

Untuk memulai Timestream for LiveAnalytics dan VPC endpoint, kami telah memberikan informasi tentang pertimbangan khusus untuk Timestream LiveAnalytics dengan titik akhir, membuat VPC titik akhir antarmuka untuk Timestream untuk, membuat kebijakan VPC VPC titik akhir untuk Timestream untuk LiveAnalytics LiveAnalytics, dan menggunakan klien Timestream (baik untuk Tulis atau Kueri) dengan titik akhir.. SDK VPC

Topik

- Bagaimana VPC titik akhir bekerja dengan Timestream
- Membuat VPC titik akhir antarmuka untuk Timestream untuk LiveAnalytics
- Membuat kebijakan VPC titik akhir untuk Timestream untuk LiveAnalytics

Bagaimana VPC titik akhir bekerja dengan Timestream

Saat Anda membuat VPC titik akhir untuk mengakses Timestream Write atau Timestream QuerySDK, semua permintaan dirutekan ke titik akhir dalam jaringan Amazon dan tidak mengakses internet publik. Lebih khusus lagi, permintaan Anda dirutekan ke titik akhir tulis dan kueri sel tempat akun Anda telah dipetakan untuk wilayah tertentu. Untuk mempelajari lebih lanjut tentang arsitektur seluler Timestream dan titik akhir khusus sel, Anda dapat merujuk ke. Arsitektur seluler Misalnya, misalkan akun Anda telah dipetakan ke `cell1` in`us-west-2`, dan Anda telah menyiapkan titik akhir VPC antarmuka untuk write (`ingest-cell1.timestream.us-west-2.amazonaws.com`) dan queries (). `query-cell1.timestream.us-west-2.amazonaws.com` Dalam hal ini, setiap permintaan tulis yang dikirim menggunakan titik akhir ini akan tetap sepenuhnya berada dalam jaringan Amazon dan tidak akan mengakses internet publik.

Pertimbangan untuk titik akhir Timestream VPC

Pertimbangkan hal berikut saat membuat VPC titik akhir untuk Timestream:

- Sebelum menyiapkan VPC titik akhir antarmuka untuk Timestream LiveAnalytics, pastikan Anda meninjau properti dan batasan titik akhir Antarmuka di Panduan Pengguna Amazon VPC.

- Timestream untuk LiveAnalytics mendukung membuat panggilan ke semua API tindakannya dari AndaVPC.
- VPCkebijakan endpoint didukung untuk Timestream untuk. LiveAnalytics Secara default, akses penuh ke Timestream untuk LiveAnalytics diizinkan melalui titik akhir. Untuk informasi selengkapnya, lihat Mengontrol akses ke layanan dengan VPC titik akhir di Panduan VPC Pengguna Amazon.
- Karena arsitektur Timestream, akses ke tindakan Tulis dan Kueri memerlukan pembuatan dua titik akhir VPC antarmuka, satu untuk masing-masing. SDK Selain itu, Anda harus menentukan titik akhir sel (Anda hanya dapat membuat titik akhir untuk sel Timestream yang Anda petakan). Informasi terperinci dapat ditemukan di membuat VPC titik akhir antarmuka untuk Timestream untuk LiveAnalytics bagian panduan ini.

Sekarang setelah Anda memahami cara LiveAnalytics kerja Timestream dengan VPC titik akhir, buat VPC titik akhir antarmuka untuk Timestream untuk. LiveAnalytics

Membuat VPC titik akhir antarmuka untuk Timestream untuk LiveAnalytics

Anda dapat membuat VPCtitik akhir antarmuka untuk Timestream untuk LiveAnalytics layanan menggunakan VPC konsol Amazon atau AWS Command Line Interface ()AWS CLI. Untuk membuat VPC titik akhir Timestream, selesaikan langkah-langkah khusus TimeStream yang dijelaskan di bawah ini.

> **Note**
>
> Sebelum menyelesaikan langkah-langkah di bawah ini, pastikan Anda memahami pertimbangan spesifik untuk titik akhir Timestream VPC.

Membangun nama layanan VPC endpoint menggunakan sel Timestream Anda

Karena arsitektur unik Timestream, titik akhir VPC antarmuka yang terpisah harus dibuat untuk masing-masing SDK (Tulis dan Kueri). Selain itu, Anda harus menentukan titik akhir sel Timestream (Anda hanya akan dapat membuat titik akhir untuk sel Timestream yang Anda petakan). Untuk menggunakan VPC Endpoint Antarmuka untuk langsung terhubung ke Timestream dari dalam AndaVPC, selesaikan langkah-langkah di bawah ini:

1. Pertama, temukan titik akhir sel Timestream yang tersedia. Untuk menemukan titik akhir sel yang tersedia, gunakan `DescribeEndpoints`tindakan (tersedia melalui Tulis dan KueriAPIs) untuk

mencantumkan titik akhir sel yang tersedia di akun Timestream Anda. Lihat contoh untuk detail lebih lanjut.

2. Setelah Anda memilih titik akhir sel untuk digunakan, buat string titik akhir VPC antarmuka untuk Timestream Write atau Query: API

   - Untuk MenulisAPI:

     ```
     com.amazonaws.<region>.timestream.ingest-<cell>
     ```

   - Untuk QueryAPI:

     ```
     com.amazonaws.<region>.timestream.query-<cell>
     ```

   di mana *<region>* adalah kode AWS wilayah yang valid dan *<cell>* adalah salah satu alamat titik akhir sel (seperti `cell1` atau`cell2`) yang dikembalikan dalam objek Endpoints oleh tindakan. DescribeEndpoints Lihat contoh untuk detail lebih lanjut.

3. Sekarang Anda telah membangun nama layanan VPC endpoint, buat endpoint antarmuka. Ketika diminta untuk memberikan nama layanan VPC endpoint, gunakan nama layanan VPC endpoint yang Anda buat di Langkah 2.

Contoh: Membangun nama layanan VPC endpoint Anda

Dalam contoh berikut, `DescribeEndpoints` tindakan dijalankan dalam AWS CLI menggunakan Tulis API di `us-west-2` wilayah:

```
aws timestream-write describe-endpoints --region us-west-2
```

Perintah ini akan mengembalikan output berikut:

```
{
    "Endpoints": [
        {
            "Address": "ingest-cell1.timestream.us-west-2.amazonaws.com",
            "CachePeriodInMinutes": 1440
        }
    ]
}
```

Dalam hal ini, *cell1* adalah *<cell>* , dan *us-west-2* adalah *<region>*. Jadi, nama layanan VPC endpoint yang dihasilkan akan terlihat seperti:

```
com.amazonaws.us-west-2.timestream.ingest-cell1
```

Sekarang setelah Anda membuat VPC titik akhir antarmuka untuk Timestream LiveAnalytics, buat kebijakan VPC titik akhir untuk Timestream untuk. LiveAnalytics

Membuat kebijakan VPC titik akhir untuk Timestream untuk LiveAnalytics

Anda dapat melampirkan kebijakan titik akhir ke VPC titik akhir yang mengontrol akses ke Timestream untuk. LiveAnalytics Kebijakan titik akhir menentukan informasi berikut:

- Prinsipal yang dapat melakukan tindakan.
- Tindakan yang dapat dilakukan.
- Sumber daya yang menjadi target tindakan.

Untuk informasi selengkapnya, lihat Mengontrol akses ke layanan dengan VPC titik akhir di Panduan VPC Pengguna Amazon.

Contoh: kebijakan VPC endpoint untuk Timestream untuk tindakan LiveAnalytics

Berikut ini adalah contoh kebijakan endpoint untuk Timestream untuk. LiveAnalytics Saat dilampirkan ke titik akhir, kebijakan ini memberikan akses ke Timestream yang terdaftar untuk LiveAnalytics tindakan (dalam hal ini, `ListDatabases`) untuk semua prinsipal di semua sumber daya.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "timestream:ListDatabases"
         ],
         "Resource":"*"
      }
   ]
}
```

Praktik terbaik keamanan untuk Amazon Timestream untuk LiveAnalytics

Amazon Timestream for LiveAnalytics menyediakan sejumlah fitur keamanan untuk dipertimbangkan saat Anda mengembangkan dan menerapkan kebijakan keamanan Anda sendiri. Praktik terbaik berikut adalah pedoman umum dan tidak mewakili solusi keamanan yang lengkap. Karena praktik terbaik ini mungkin tidak sesuai atau cukup untuk lingkungan Anda, anggap sebagai pertimbangan yang membantu dan bukan sebagai resep.

Topik

Timestream untuk praktik LiveAnalytics terbaik keamanan preventif

Praktik terbaik berikut dapat membantu Anda mengantisipasi dan mencegah insiden keamanan di Timestream for. LiveAnalytics

Enkripsi diam

Timestream untuk LiveAnalytics mengenkripsi saat istirahat semua data pengguna yang disimpan dalam tabel menggunakan kunci enkripsi yang disimpan di AWS Key Management Service ().AWS KMS Hal ini memberi lapisan perlindungan data tambahan dengan mengamankan data Anda dari akses yang tidak sah ke penyimpanan dasar.

Timestream untuk LiveAnalytics menggunakan kunci default layanan tunggal (AWS dimilikiCMK) untuk mengenkripsi semua tabel Anda. Jika kunci ini tidak ada, itu dibuat untuk Anda. Kunci default layanan tidak dapat dinonaktifkan. Untuk informasi selengkapnya, lihat Timestream untuk LiveAnalytics Enkripsi saat Istirahat.

Gunakan IAM peran untuk mengautentikasi akses ke Timestream LiveAnalytics

Untuk pengguna, aplikasi, dan AWS layanan lain untuk mengakses Timestream LiveAnalytics, mereka harus menyertakan AWS kredensi yang valid dalam permintaan mereka. AWS API Anda tidak boleh menyimpan AWS kredensil secara langsung di aplikasi atau EC2 instance. Ini adalah kredensial jangka panjang yang tidak dirotasi secara otomatis, dan karenanya dapat menimbulkan dampak bisnis yang signifikan jika dibobol. IAMPeran memungkinkan Anda memperoleh kunci akses sementara yang dapat digunakan untuk mengakses AWS layanan dan sumber daya.

Untuk informasi selengkapnya, lihat IAMPeran.

Gunakan IAM kebijakan untuk Timestream untuk otorisasi LiveAnalytics dasar

Saat memberikan izin, Anda memutuskan siapa yang mendapatkannya, Timestream mana yang mendapatkan izin, dan tindakan spesifik yang ingin Anda izinkan pada sumber daya tersebut. LiveAnalytics APIs Menerapkan akses hak akses paling rendah adalah hal mendasar dalam mengurangi risiko keamanan dan dampak yang dapat disebabkan oleh kesalahan atau niat jahat.

Lampirkan kebijakan izin ke IAM identitas (yaitu, pengguna, grup, dan peran) dan dengan demikian memberikan izin untuk melakukan operasi di Timestream untuk sumber daya. LiveAnalytics

Anda dapat melakukan hal ini dengan cara berikut:

- AWS kebijakan terkelola (standar)
- Kebijakan yang dikelola pelanggan
- Otorisasi berbasis tag

Pertimbangkan enkripsi di sisi klien

Jika Anda menyimpan data sensitif atau rahasia di Timestream for LiveAnalytics, Anda mungkin ingin mengenkripsi data tersebut sedekat mungkin dengan asalnya sehingga data Anda terlindungi sepanjang siklus hidupnya. Mengenkripsi data sensitif Anda saat transit dan diam membantu memastikan bahwa data teks biasa Anda tidak tersedia untuk pihak ketiga mana pun.

Bekerja dengan layanan yang lain

Amazon Timestream untuk LiveAnalytics terintegrasi dengan berbagai AWS layanan dan alat pihak ketiga yang populer. Saat ini, Timestream untuk LiveAnalytics mendukung integrasi dengan yang berikut:

Topik

- Amazon DynamoDB
- AWS Lambda
- AWS IoT Core
- Layanan Terkelola Amazon untuk Apache Flink
- Amazon Kinesis
- Amazon MQ
- Amazon MSK

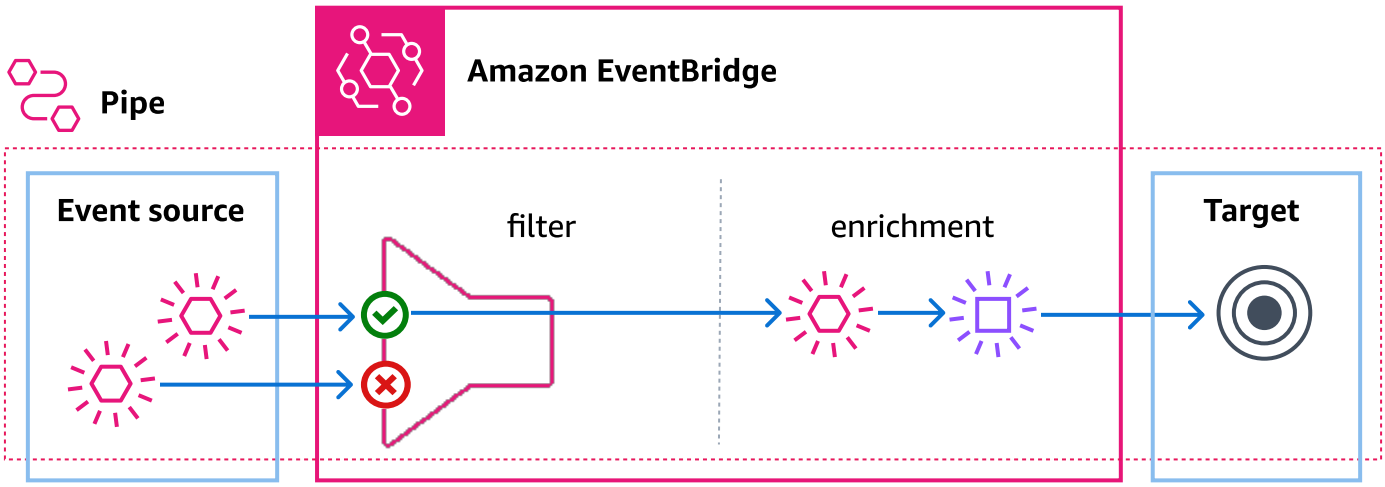
- Amazon QuickSight
- Amazon SageMaker
- Amazon SQS
- Menggunakan DBeaver untuk bekerja dengan Amazon Timestream
- Grafana
- Menggunakan SquaredUp untuk bekerja dengan Amazon Timestream
- Telegraf sumber terbuka
- JDBC
- ODBC
- VPCtitik akhir ()AWS PrivateLink

Amazon DynamoDB

Menggunakan EventBridge Pipes untuk mengirim data DynamoDB ke Timestream

Anda dapat menggunakan EventBridge Pipes untuk mengirim data dari aliran DynamoDB ke Amazon Timestream untuk tabel. LiveAnalytics

Pipa dimaksudkan untuk point-to-point integrasi antara sumber dan target yang didukung, dengan dukungan untuk transformasi dan pengayaan lanjutan. Pipa mengurangi kebutuhan akan pengetahuan khusus dan kode integrasi saat mengembangkan arsitektur berbasis peristiwa. Untuk menyiapkan pipa, Anda memilih sumber, menambahkan pemfilteran opsional, menentukan pengayaan opsional, dan memilih target untuk data peristiwa.



Untuk informasi lebih lanjut tentang EventBridge Pipa, lihat EventBridge Pipa di Panduan EventBridge Pengguna. Untuk informasi tentang mengonfigurasi pipa untuk mengirimkan peristiwa ke Amazon Timestream LiveAnalytics untuk tabel, EventBridge lihat Spesifikasi target Pipes.

AWS Lambda

Anda dapat membuat fungsi Lambda yang berinteraksi dengan Timestream untuk. LiveAnalytics Misalnya, Anda dapat membuat fungsi Lambda yang berjalan secara berkala untuk mengeksekusi kueri di Timestream dan mengirim SNS pemberitahuan berdasarkan hasil kueri yang memenuhi satu atau beberapa kriteria. Untuk mempelajari lebih lanjut tentang Lambda, lihat dokumentasi Lambda AWS .

Topik

- Bangun fungsi AWS Lambda menggunakan Amazon LiveAnalytics Timestream dengan Python
- Bangun fungsi AWS Lambda menggunakan Amazon Timestream untuk with LiveAnalytics JavaScript
- Bangun fungsi AWS Lambda menggunakan Amazon Timestream dengan Go LiveAnalytics
- Bangun fungsi AWS Lambda menggunakan Amazon LiveAnalytics Timestream dengan C #

Bangun fungsi AWS Lambda menggunakan Amazon LiveAnalytics Timestream dengan Python

Untuk membangun fungsi AWS Lambda menggunakan Amazon Timestream LiveAnalytics dengan Python, ikuti langkah-langkah di bawah ini.

1. Buat IAM peran untuk Lambda untuk berasumsi yang akan memberikan izin yang diperlukan untuk mengakses Layanan Timestream, sebagaimana diuraikan dalam. Menyediakan Timestream untuk akses LiveAnalytics
2. Edit hubungan kepercayaan IAM peran untuk menambahkan layanan Lambda. Anda dapat menggunakan perintah di bawah ini untuk memperbarui peran yang ada sehingga AWS Lambda dapat menganggapnya:

    a. Buat dokumen kebijakan kepercayaan:

```
cat > Lambda-Role-Trust-Policy.json << EOF
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "lambda.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

b. Perbarui peran dari langkah sebelumnya dengan dokumen kepercayaan

```
aws iam update-assume-role-policy --role-name <name_of_the_role_from_step_1> --
policy-document file://Lambda-Role-Trust-Policy.json
```

Referensi terkait ada di TimestreamWritedan TimestreamQuery.

Bangun fungsi AWS Lambda menggunakan Amazon Timestream untuk with LiveAnalytics JavaScript

Untuk membangun fungsi AWS Lambda menggunakan Amazon Timestream LiveAnalytics for JavaScript with, ikuti petunjuk yang diuraikan di sini.

Referensi terkait ada di Timestream Write Client - AWS SDK untuk JavaScript v3 dan Timestream Query Client - AWS SDK untuk v3. JavaScript

Bangun fungsi AWS Lambda menggunakan Amazon Timestream dengan Go LiveAnalytics

Untuk membangun fungsi AWS Lambda menggunakan Amazon Timestream LiveAnalytics for with Go, ikuti petunjuk yang diuraikan di sini.

Referensi terkait ada di timestreamwrite dan timestreamquery.

Bangun fungsi AWS Lambda menggunakan Amazon LiveAnalytics Timestream dengan C

Untuk membangun fungsi AWS Lambda menggunakan Amazon Timestream LiveAnalytics dengan C#, ikuti petunjuk yang diuraikan di sini.

Referensi terkait ada di Amazon. TimestreamWritedan Amazon. TimestreamQuery.

AWS IoT Core

Anda dapat mengumpulkan data dari perangkat IoT menggunakan IoT Core dan merutekan data ke Amazon Timestream melalui tindakan aturan AWS IoT Core. AWS Tindakan aturan IoT menentukan apa yang harus dilakukan ketika aturan dipicu. Anda dapat menentukan tindakan untuk mengirim data ke tabel Amazon Timestream, database Amazon DynamoDB, dan menjalankan fungsi Lambda. AWS

Tindakan Timestream dalam Aturan IoT digunakan untuk menyisipkan data dari pesan masuk langsung ke Timestream. Tindakan mem-parsing hasil pernyataan IoT SQL Core dan menyimpan data di Timestream. Nama-nama bidang dari kumpulan SQL hasil yang dikembalikan digunakan sebagai ukuran: :name dan nilai bidang adalah ukuran: :value.

Misalnya, perhatikan SQL pernyataan dan muatan pesan sampel:

```
SELECT temperature, humidity from 'iot/topic'
```

```
{
  "dataFormat": 5,
  "rssi": -88,
  "temperature": 24.04,
  "humidity": 43.605,
  "pressure": 101082,
  "accelerationX": 40,
  "accelerationY": -20,
  "accelerationZ": 1016,
  "battery": 3007,
  "txPower": 4,
  "movementCounter": 219,
  "device_id": 46216,
  "device_firmware_sku": 46216
}
```

Jika tindakan aturan IoT Core untuk Timestream dibuat dengan SQL pernyataan di atas, dua catatan akan ditambahkan ke Timestream dengan nama ukuran suhu dan kelembaban dan nilai pengukuran masing-masing 24,04 dan 43,605.

Anda dapat mengubah nama ukuran rekaman yang ditambahkan ke Timestream dengan menggunakan operator AS dalam SELECT pernyataan. SQLPernyataan di bawah ini akan membuat catatan dengan nama pesan temp bukan suhu.

Tipe data ukuran disimpulkan dari tipe data dari nilai payload pesan. JSONtipe data seperti integer, double, boolean, dan string dipetakan ke tipe data Timestream dariBIGINT,,DOUBLE, BOOLEAN dan masing-masing. VARCHAR Data juga dapat dipaksa ke tipe data tertentu menggunakan fungsi cast (). Anda dapat menentukan stempel waktu pengukuran. Jika stempel waktu dibiarkan kosong, waktu entri diproses digunakan.

Anda dapat merujuk ke dokumentasi tindakan aturan Timestream untuk detail tambahan

Untuk membuat tindakan aturan IoT Core untuk menyimpan data di Timestream, ikuti langkah-langkah di bawah ini:

Topik

- Prasyarat
- Menggunakan konsol
- Menggunakan CLI
- Aplikasi sampel
- Video tutorial

Prasyarat

1. Buat database di Amazon Timestream menggunakan petunjuk yang dijelaskan di. Buat database
2. Buat tabel di Amazon Timestream menggunakan petunjuk yang dijelaskan di. Membuat tabel

Menggunakan konsol

1. Gunakan AWS Management Console for AWS IoT Core untuk membuat aturan dengan mengklik Manage > Messsage routing > Rules diikuti oleh Create rule.
2. Tetapkan nama aturan ke nama pilihan Anda dan SQL ke teks yang ditunjukkan di bawah ini

```
SELECT temperature as temp, humidity from 'iot/topic'
```

3. Pilih Timestream dari daftar Tindakan
4. Tentukan nama database, tabel, dan dimensi Timestream bersama dengan peran untuk menulis data ke Timestream. Jika peran tidak ada, Anda dapat membuatnya dengan mengklik Buat Peran
5. Untuk menguji aturan, ikuti instruksi yang ditunjukkan di sini.

Menggunakan CLI

Jika Anda belum menginstal AWS Command Line Interface (AWS CLI), lakukan dari sini.

1. Simpan payload aturan berikut dalam JSON file bernama timestream_rule.json. Ganti *arn:aws:iam::123456789012:role/TimestreamRole* dengan peran Anda yang memberikan akses AWS IoT untuk menyimpan data di Amazon Timestream

```
{
    "actions": [
            {
                "timestream": {
                    "roleArn": "arn:aws:iam::123456789012:role/TimestreamRole",
                    "tableName": "devices_metrics",
                    "dimensions": [
                        {
                            "name": "device_id",
                            "value": "${clientId()}"
                        },
                        {
                            "name": "device_firmware_sku",
                            "value": "My Static Metadata"
                        }
                    ],
                    "databaseName": "record_devices"
                }
            }
    ],
    "sql": "select * from 'iot/topic'",
    "awsIotSqlVersion": "2016-03-23",
    "ruleDisabled": false
```

```
}
```

2. Buat aturan topik menggunakan perintah berikut

```
aws iot create-topic-rule --rule-name timestream_test --topic-rule-payload file://
<path/to/timestream_rule.json> --region us-east-1
```

3. Mengambil rincian aturan topik menggunakan perintah berikut

```
aws iot get-topic-rule --rule-name timestream_test
```

4. Simpan payload pesan berikut dalam file bernama timestream_msg.json

```
{
  "dataFormat": 5,
  "rssi": -88,
  "temperature": 24.04,
  "humidity": 43.605,
  "pressure": 101082,
  "accelerationX": 40,
  "accelerationY": -20,
  "accelerationZ": 1016,
  "battery": 3007,
  "txPower": 4,
  "movementCounter": 219,
  "device_id": 46216,
  "device_firmware_sku": 46216
}
```

5. Uji aturan menggunakan perintah berikut

```
aws iot-data publish --topic 'iot/topic' --payload file://<path/to/
timestream_msg.json>
```

Aplikasi sampel

Untuk membantu Anda memulai menggunakan Timestream dengan AWS IoT Core, kami telah membuat aplikasi sampel yang berfungsi penuh yang membuat artefak yang diperlukan AWS di IoT Core dan Timestream untuk membuat aturan topik dan contoh aplikasi untuk menerbitkan data ke topik tersebut.

1. Kloning GitHub repositori untuk aplikasi sampel untuk integrasi AWS IoT Core mengikuti instruksi dari GitHub
2. Ikuti petunjuk dalam READMEuntuk menggunakan AWS CloudFormation templat untuk membuat artefak yang diperlukan di Amazon Timestream dan AWS IoT Core dan untuk mempublikasikan pesan sampel ke topik tersebut.

Video tutorial

Video ini menjelaskan cara kerja IoT Core dengan Timestream.

Layanan Terkelola Amazon untuk Apache Flink

Anda dapat menggunakan Apache Flink untuk mentransfer data deret waktu Anda dari Amazon Managed Service untuk Apache Flink, MSK Amazon, Apache Kafka, dan teknologi streaming lainnya langsung ke Amazon Timestream untuk. LiveAnalytics Kami telah membuat konektor data sampel Apache Flink untuk Timestream. Kami juga telah membuat contoh aplikasi untuk mengirim data ke Amazon Kinesis sehingga data dapat mengalir dari Kinesis ke Managed Service untuk Apache Flink, dan akhirnya ke Amazon Timestream. Semua artefak ini tersedia untuk Anda. GitHub Tutorial video ini menjelaskan pengaturan.

> **Note**
>
> Java 11 adalah versi yang direkomendasikan untuk menggunakan Managed Service untuk Apache Flink Application. Jika Anda memiliki beberapa versi Java, pastikan Anda mengekspor Java 11 ke variabel HOME lingkungan JAVA _ Anda.

Topik

- Aplikasi sampel
- Video tutorial

Aplikasi sampel

Untuk memulai, ikuti prosedur di bawah ini:

1. Buat database di Timestream dengan nama `kdaflink` mengikuti petunjuk yang dijelaskan di Buat database

2. Buat tabel di Timestream dengan nama `kinesisdata1` mengikuti petunjuk yang dijelaskan dalam Membuat tabel
3. Membuat Amazon Kinesis Data Stream dengan nama `TimestreamTestStream` mengikuti petunjuk yang dijelaskan dalam Membuat Stream
4. Kloning GitHub repositori untuk konektor data Apache Flink untuk Timestream mengikuti instruksi dari GitHub
5. Untuk mengkompilasi, menjalankan dan menggunakan aplikasi sampel, ikuti instruksi di konektor data sampel Apache Flink README
6. Kompilasi Layanan Terkelola untuk aplikasi Apache Flink mengikuti instruksi untuk Mengompilasi Kode Aplikasi
7. Unggah Layanan Terkelola untuk aplikasi Apache Flink biner mengikuti petunjuk untuk Mengunggah Kode Streaming Apache Flink

    a. Setelah mengklik Buat Aplikasi, klik tautan IAM Peran untuk aplikasi

    b. Lampirkan IAM kebijakan untuk AmazonKinesisReadOnlyAccessdan AmazonTimestreamFullAccess.

    > **Note**
    >
    > IAMKebijakan di atas tidak terbatas pada sumber daya tertentu dan tidak cocok untuk penggunaan produksi. Untuk sistem produksi, pertimbangkan untuk menggunakan kebijakan yang membatasi akses ke sumber daya tertentu.

8. Kloning GitHub repositori untuk data penulisan aplikasi sampel ke Kinesis mengikuti instruksi dari GitHub
9. Ikuti petunjuk dalam READMEuntuk menjalankan aplikasi sampel untuk menulis data ke Kinesis
10. Jalankan satu atau beberapa kueri di Timestream untuk memastikan bahwa data sedang dikirim dari Kinesis ke Layanan Terkelola untuk Apache Flink ke Timestream mengikuti petunjuk Membuat tabel

Video tutorial

Video ini menjelaskan cara menggunakan Timestream dengan Managed Service untuk Apache Flink.

Amazon Kinesis

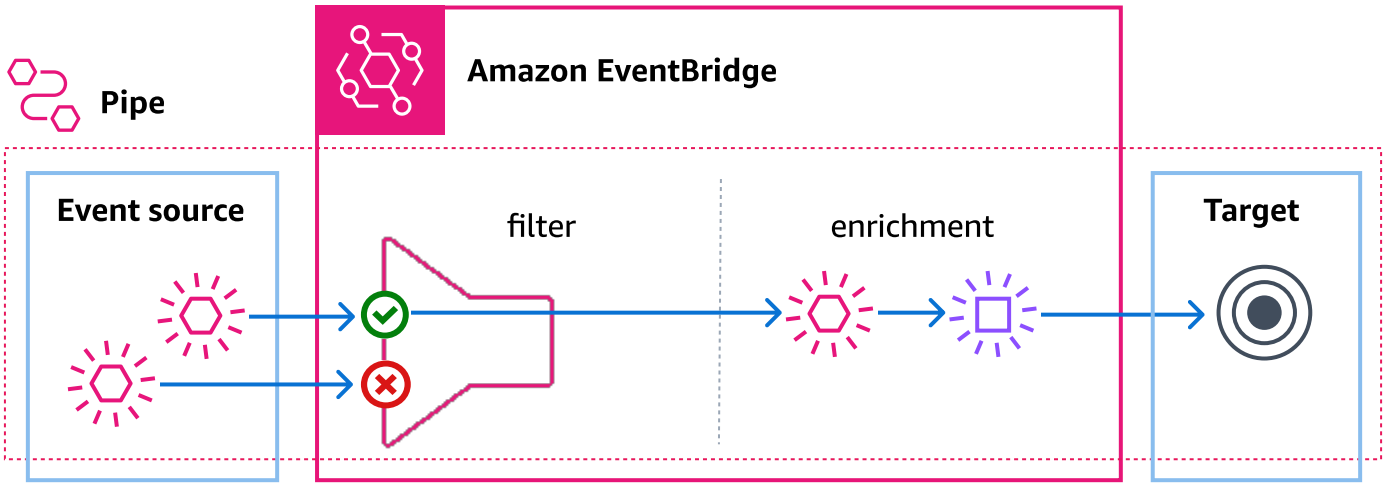
Menggunakan Layanan Terkelola Amazon untuk Apache Flink

Anda dapat mengirim data dari Kinesis Data Streams ke LiveAnalytics Timestream untuk menggunakan konektor data sampel untuk Managed Timestream Service for Apache Flink. Lihat Apache Flink Layanan Terkelola Amazon untuk Apache Flink untuk informasi lebih lanjut.

Menggunakan EventBridge Pipa untuk mengirim data Kinesis ke Timestream

Anda dapat menggunakan EventBridge Pipes untuk mengirim data dari aliran Kinesis ke Amazon LiveAnalytics Timestream untuk tabel.

Pipa dimaksudkan untuk point-to-point integrasi antara sumber dan target yang didukung, dengan dukungan untuk transformasi dan pengayaan lanjutan. Pipa mengurangi kebutuhan akan pengetahuan khusus dan kode integrasi saat mengembangkan arsitektur berbasis peristiwa. Untuk menyiapkan pipa, Anda memilih sumber, menambahkan pemfilteran opsional, menentukan pengayaan opsional, dan memilih target untuk data peristiwa.



Integrasi ini memungkinkan Anda untuk memanfaatkan kekuatan kemampuan analisis data deret waktu, sekaligus menyederhanakan saluran konsumsi data Anda. Timestream

Menggunakan EventBridge Pipa dengan Timestream menawarkan manfaat sebagai berikut:

- Penyerapan Data Real-time: Streaming data dari Kinesis langsung ke Timestream untuk LiveAnalytics, memungkinkan analisis dan pemantauan waktu nyata.

- Integrasi Seamless: Memanfaatkan EventBridge Pipa untuk mengelola aliran data tanpa perlu integrasi kustom yang kompleks.
- Penyaringan dan Transformasi yang Ditingkatkan: Saring atau ubah catatan Kinesis sebelum disimpan Timestream untuk memenuhi persyaratan pemrosesan data spesifik Anda.
- Skalabilitas: Menangani aliran data throughput tinggi dan memastikan pemrosesan data yang efisien dengan paralelisme bawaan dan kemampuan batching.

Konfigurasi

Untuk menyiapkan EventBridge Pipe untuk mengalirkan data dari Kinesis ke Timestream, ikuti langkah-langkah berikut:

1. Buat aliran Kinesis

   Pastikan Anda memiliki aliran data Kinesis aktif dari mana Anda ingin menelan data.

2. Buat Timestream database dan tabel

   Siapkan Timestream database dan tabel tempat data akan disimpan.

3. Konfigurasikan EventBridge Pipa:
   - Sumber: Pilih aliran Kinesis Anda sebagai sumbernya.
   - Target: Pilih Timestream sebagai target.
   - Pengaturan Batching: Tentukan jendela batching dan ukuran batch untuk mengoptimalkan pemrosesan data dan mengurangi latensi.

> **Important**
>
> Saat menyiapkan pipa, kami sarankan untuk menguji kebenaran semua konfigurasi dengan menelan beberapa catatan. Harap dicatat bahwa pembuatan pipa yang berhasil tidak menjamin bahwa pipa sudah benar dan data akan mengalir tanpa kesalahan. Mungkin ada kesalahan runtime, seperti tabel yang salah, parameter jalur dinamis yang salah, atau Timestream catatan tidak valid setelah menerapkan pemetaan, yang akan ditemukan ketika data aktual mengalir melalui pipa.

Konfigurasi berikut menentukan tingkat di mana data dicerna:

- BatchSize: Ukuran maksimum batch yang akan dikirim ke Timestream untuk LiveAnalytics. Rentang: 0 - 100. Rekomendasi adalah untuk menjaga nilai ini sebagai 100 untuk mendapatkan throughput maksimum.
- MaximumBatchingWindowInSeconds: Waktu maksimum untuk menunggu untuk mengisi batchSize sebelum batch dikirim ke Timestream untuk LiveAnalytics target. Bergantung pada tingkat kejadian yang masuk, konfigurasi ini akan menentukan penundaan konsumsi, rekomendasinya adalah menjaga nilai ini < 10 detik untuk terus mengirim data dalam waktu dekat. Timestream
- ParallelizationFactor: Jumlah batch yang akan diproses secara bersamaan dari setiap pecahan. Rekomendasi adalah menggunakan nilai maksimum 10 untuk mendapatkan throughput maksimum dan mendekati konsumsi real-time.

  Jika aliran Anda dibaca oleh beberapa target, gunakan fan-out yang disempurnakan untuk menyediakan konsumen khusus ke pipa Anda untuk mencapai throughput yang tinggi. Untuk informasi selengkapnya, lihat Mengembangkan konsumen fan-out yang disempurnakan dengan Kinesis Data Streams API di Kinesis Data Streams Panduan Pengguna.

> **Note**
>
> Throughput maksimum yang dapat dicapai dibatasi oleh eksekusi pipa bersamaan per akun.

Konfigurasi berikut memastikan pencegahan kehilangan data:

- DeadLetterConfig: Rekomendasi adalah untuk selalu mengkonfigurasi DeadLetterConfig untuk menghindari kehilangan data untuk kasus ketika peristiwa tidak dapat dicerna ke Timestream LiveAnalytics karena kesalahan pengguna.

Optimalkan kinerja pipa Anda dengan pengaturan konfigurasi berikut, yang membantu mencegah catatan menyebabkan perlambatan atau penyumbatan.

- MaximumRecordAgeInSeconds: Catatan yang lebih tua dari ini tidak akan diproses dan akan langsung dipindahkan keDLQ. Sebaiknya atur nilai ini tidak lebih tinggi dari periode retensi penyimpanan memori yang dikonfigurasi dari Timestream tabel target.
- MaximumRetryAttempts: Jumlah percobaan ulang untuk catatan sebelum catatan dikirim ke DeadLetterQueue. Rekomendasi adalah untuk mengkonfigurasi ini pada 10. Ini harus dapat

membantu mengatasi masalah sementara dan untuk masalah yang terus-menerus, catatan akan dipindahkan ke DeadLetterQueue dan membuka blokir sisa aliran.

- OnPartialBatchItemFailure: Untuk sumber yang mendukung pemrosesan batch sebagian, kami sarankan Anda untuk mengaktifkan ini dan mengonfigurasinya sebagai AUTOMATIC _ BISECT untuk mencoba lagi tambahan catatan yang gagal sebelum menjatuhkan/mengirim ke. DLQ

Contoh konfigurasI

Berikut adalah contoh cara mengkonfigurasi EventBridge Pipe untuk mengalirkan data dari aliran Kinesis ke tabel: Timestream

Example IAM pembaruan kebijakan untuk Timestream

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream:WriteRecords"
            ],
            "Resource": [
                "arn:aws:timestream:us-east-1:123456789012:database/my-database/table/
my-table"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "timestream:DescribeEndpoints"
            ],
            "Resource": "*"
        }
    ]
}
```

Example Konfigurasi aliran Kinesis

```
{
  "Source": "arn:aws:kinesis:us-east-1:123456789012:stream/my-kinesis-stream",
  "SourceParameters": {
```

```
    "KinesisStreamParameters": {
        "BatchSize": 100,
        "DeadLetterConfig": {
            "Arn": "arn:aws:sqs:us-east-1:123456789012:my-sqs-queue"
        },
       "MaximumBatchingWindowInSeconds": 5,
        "MaximumRecordAgeInSeconds": 1800,
        "MaximumRetryAttempts": 10,
        "StartingPosition": "LATEST",
       "OnPartialBatchItemFailure": "AUTOMATIC_BISECT"
    }
  }
}
```

Example Timestream konfigurasi target

```
{
    "Target": "arn:aws:timestream:us-east-1:123456789012:database/my-database/table/my-
table",
    "TargetParameters": {
        "TimestreamParameters": {
            "DimensionMappings": [
                {
                    "DimensionName": "sensor_id",
                    "DimensionValue": "$.data.device_id",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "DimensionName": "sensor_type",
                    "DimensionValue": "$.data.sensor_type",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "DimensionName": "sensor_location",
                    "DimensionValue": "$.data.sensor_loc",
                    "DimensionValueType": "VARCHAR"
                }
            ],
            "MultiMeasureMappings": [
                {
                    "MultiMeasureName": "readings",
                    "MultiMeasureAttributeMappings": [
                        {
```

```
                            "MultiMeasureAttributeName": "temperature",
                            "MeasureValue": "$.data.temperature",
                            "MeasureValueType": "DOUBLE"
                        },
                        {
                            "MultiMeasureAttributeName": "humidity",
                            "MeasureValue": "$.data.humidity",
                            "MeasureValueType": "DOUBLE"
                        },
                        {
                            "MultiMeasureAttributeName": "pressure",
                            "MeasureValue": "$.data.pressure",
                            "MeasureValueType": "DOUBLE"
                        }
                    ]
                }
            ],
            "SingleMeasureMappings": [],
            "TimeFieldType": "TIMESTAMP_FORMAT",
            "TimestampFormat": "yyyy-MM-dd HH:mm:ss.SSS",
            "TimeValue": "$.data.time",
            "VersionValue": "$.approximateArrivalTimestamp"
        }
    }
}
```

Transformasi acara

EventBridge Pipa memungkinkan Anda untuk mengubah data sebelum mencapai Timestream. Anda dapat menentukan aturan transformasi untuk memodifikasi Kinesis catatan yang masuk, seperti mengubah nama bidang.

Misalkan Kinesis aliran Anda berisi data suhu dan kelembaban. Anda dapat menggunakan EventBridge transformasi untuk mengganti nama bidang ini sebelum memasukkannya ke dalam. Timestream

Praktik terbaik

Batching dan Buffering

- Konfigurasikan jendela dan ukuran batching untuk menyeimbangkan antara latensi tulis dan efisiensi pemrosesan.

- Gunakan jendela batching untuk mengumpulkan data yang cukup sebelum diproses, mengurangi overhead dari batch kecil yang sering.

Pemrosesan Paralel

Manfaatkan ParallelizationFactorpengaturan untuk meningkatkan konkurensi, terutama untuk aliran throughput tinggi. Ini memastikan bahwa beberapa batch dari setiap pecahan dapat diproses secara bersamaan.

Transformasi Data

Manfaatkan kemampuan transformasi EventBridge Pipa untuk menyaring dan meningkatkan catatan sebelum menyimpannya Timestream. Ini dapat membantu menyelaraskan data dengan persyaratan analitis Anda.

Keamanan

- Pastikan bahwa IAM peran yang digunakan untuk EventBridge Pipes memiliki izin yang diperlukan untuk membaca Kinesis dan menulis. Timestream
- Gunakan enkripsi dan langkah-langkah kontrol akses untuk mengamankan data dalam perjalanan dan saat istirahat.

Kegagalan debugging

- Menonaktifkan Pipa Secara Otomatis

  Pipa akan dinonaktifkan secara otomatis dalam waktu sekitar 2 jam jika target tidak ada atau memiliki masalah izin
- Pembatasan

  Pipa memiliki kemampuan untuk mundur dan mencoba kembali secara otomatis sampai throttle berkurang.
- Mengaktifkan Log

  Kami menyarankan Anda mengaktifkan Log di ERROR level dan menyertakan data eksekusi untuk mendapatkan lebih banyak wawasan tentang kegagalan. Setelah kegagalan, log ini akan berisi request/response sent/received dari Timestream. Ini membantu Anda memahami kesalahan yang terkait dan jika perlu memproses ulang catatan setelah memperbaikinya.

Pemantauan

Kami menyarankan Anda untuk mengatur alarm berikut ini untuk mendeteksi masalah apa pun dengan aliran data:

- Usia Maksimum Catatan dalam Sumber
  - `GetRecords.IteratorAgeMilliseconds`
- Metrik kegagalan dalam Pipa
  - `ExecutionFailed`
  - `TargetStageFailed`
- Timestream Menulis API kesalahan
  - `UserErrors`

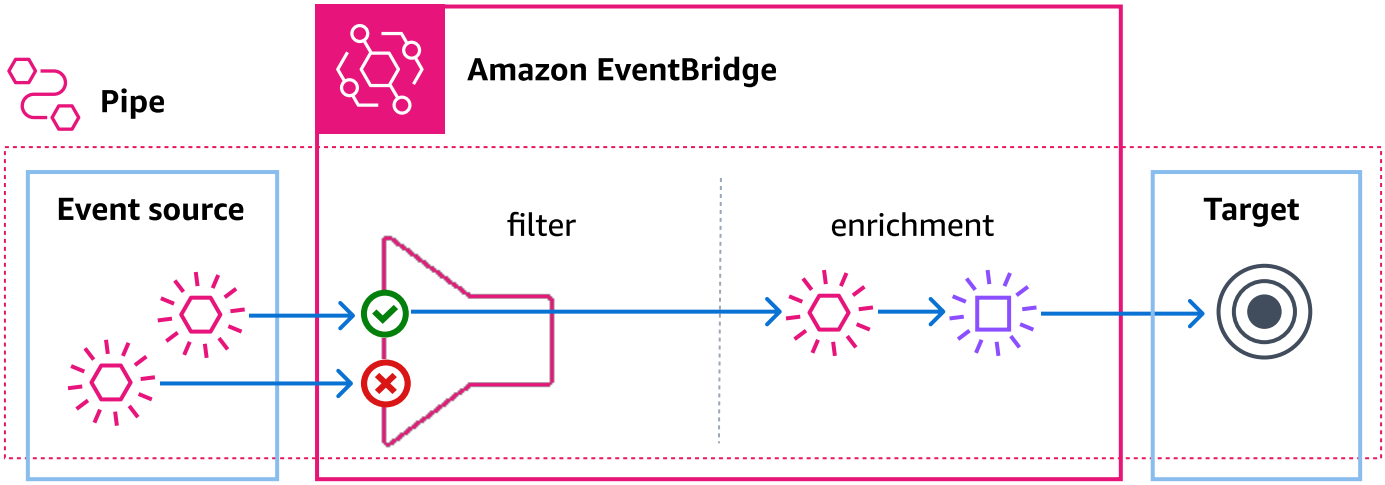
Untuk metrik pemantauan tambahan, lihat Pemantauan EventBridge di Panduan EventBridge Pengguna.

Amazon MQ

Menggunakan EventBridge Pipes untuk mengirim data Amazon MQ ke Timestream

Anda dapat menggunakan EventBridge Pipes untuk mengirim data dari broker Amazon MQ ke Amazon LiveAnalytics Timestream untuk tabel.

Pipa dimaksudkan untuk point-to-point integrasi antara sumber dan target yang didukung, dengan dukungan untuk transformasi dan pengayaan lanjutan. Pipa mengurangi kebutuhan akan pengetahuan khusus dan kode integrasi saat mengembangkan arsitektur berbasis peristiwa. Untuk menyiapkan pipa, Anda memilih sumber, menambahkan pemfilteran opsional, menentukan pengayaan opsional, dan memilih target untuk data peristiwa.



Untuk informasi lebih lanjut tentang EventBridge Pipa, lihat EventBridge Pipa di Panduan EventBridge Pengguna. Untuk informasi tentang mengonfigurasi pipa untuk mengirimkan peristiwa ke Amazon Timestream LiveAnalytics untuk tabel, EventBridge lihat Spesifikasi target Pipes.

Amazon MSK

Menggunakan Managed Service untuk Apache Flink untuk mengirim Amazon MSK data ke Timestream LiveAnalytics

Anda dapat mengirim data dari Amazon MSK ke Timestream dengan membuat konektor data yang mirip dengan konektor Timestream data sampel untuk Layanan Terkelola untuk Apache Flink. Lihat Layanan Terkelola Amazon untuk Apache Flink untuk informasi lebih lanjut.

Menggunakan Kafka Connect untuk mengirim MSK data Amazon ke Timestream LiveAnalytics

Anda dapat menggunakan Kafka Connect untuk menyerap data deret waktu Anda dari Amazon MSK langsung ke Timestream untuk. LiveAnalytics

Kami telah membuat sampel Konektor Sink Kafka untuk Timestream. Kami juga telah membuat contoh rencana jMeter pengujian Apache untuk menerbitkan data ke topik Kafka, sehingga data dapat mengalir dari topik melalui Konektor Sink Timestream Kafka, ke Timestream untuk tabel. LiveAnalytics Semua artefak ini tersedia di GitHub.

> **Note**
>
> Java 11 adalah versi yang direkomendasikan untuk menggunakan Konektor Sink Timestream Kafka. Jika Anda memiliki beberapa versi Java, pastikan Anda mengekspor Java 11 ke variabel HOME lingkungan JAVA _ Anda.

Membuat contoh aplikasi

Untuk memulai, ikuti prosedur di bawah ini.

1. Di Timestream for LiveAnalytics, buat database dengan nama`kafkastream`.

   Lihat prosedur ??? untuk instruksi terperinci.
2. Di Timestream for LiveAnalytics, buat tabel dengan nama`purchase_history`.

   Lihat prosedur ??? untuk instruksi terperinci.
3. Ikuti petunjuk yang dibagikan untuk membuat yang berikut:, dan.

   - Sebuah Amazon MSK cluster
   - Sebuah Amazon EC2 instance yang dikonfigurasi sebagai mesin klien produsen Kafka
   - Topik Kafka

   Lihat prasyarat proyek kafka_ingestor untuk instruksi terperinci.
4. Kloning repositori Konektor Timestream Sink Kafka.

   Lihat Mengkloning repositori GitHub untuk petunjuk terperinci.
5. Kompilasi kode plugin.

   Lihat Konektor - Bangun dari sumber GitHub untuk instruksi terperinci.
6. Unggah file berikut ke bucket S3: mengikuti petunjuk yang dijelaskan di.

   - File jar (kafka-connector-timestream-> VERSION <- jar-with-dependencies .jar) dari direktori `/target`
   - Contoh file skema json,. `purchase_history.json`

   Lihat Mengunggah objek di Panduan Amazon S3 Pengguna untuk petunjuk terperinci.

7. Buat dua VPC titik akhir. Titik akhir ini akan digunakan oleh MSK Konektor untuk mengakses sumber daya menggunakan AWS PrivateLink.

   - Satu untuk mengakses Amazon S3 ember
   - Satu untuk mengakses Timestream untuk LiveAnalytics tabel.

   Lihat VPCEndpoints untuk petunjuk rinci.

8. Buat plugin khusus dengan file jar yang diunggah.

   Lihat Plugin di Panduan Amazon MSK Pengembang untuk petunjuk terperinci.

9. Buat konfigurasi pekerja kustom dengan JSON konten yang dijelaskan dalam parameter Konfigurasi Pekerja. mengikuti petunjuk yang dijelaskan dalam

   Lihat Membuat konfigurasi pekerja kustom di Panduan Amazon MSK Pengembang untuk petunjuk terperinci.

10. Buat IAM peran eksekusi layanan.

    Lihat Peran IAM Layanan untuk petunjuk terperinci.

11. Buat Amazon MSK konektor dengan plugin kustom, konfigurasi pekerja kustom, dan IAM peran eksekusi layanan yang dibuat pada langkah sebelumnya dan dengan Konfigurasi Konektor Sampel.

    Lihat Membuat konektor di Panduan Amazon MSK Pengembang untuk petunjuk terperinci.

    Pastikan untuk memperbarui nilai parameter konfigurasi di bawah ini dengan nilai masing-masing. Lihat parameter Konfigurasi Konektor untuk detailnya.

    - `aws.region`
    - `timestream.schema.s3.bucket.name`
    - `timestream.ingestion.endpoint`

    Pembuatan konektor membutuhkan waktu 5-10 menit untuk menyelesaikannya. Pipa siap ketika statusnya berubah menjadi`Running`.

12. Publikasikan aliran pesan berkelanjutan untuk menulis data ke topik Kafka yang dibuat.

    Lihat Cara menggunakannya untuk instruksi terperinci.

13. Jalankan satu atau beberapa kueri untuk memastikan bahwa data sedang dikirim dari MSK Connect Amazon MSK ke Timestream untuk LiveAnalytics tabel.

    Lihat prosedur ??? untuk instruksi terperinci.

Sumber daya tambahan

Blog, konsumsi data tanpa server real-time dari cluster Kafka Anda ke Timestream untuk menggunakan LiveAnalytics Kafka Connect menjelaskan pengaturan end-to-end pipeline menggunakan Timestream untuk Kafka Sink Connector, mulai dari mesin klien produsen LiveAnalytics Kafka yang menggunakan rencana jMeter pengujian Apache untuk menerbitkan ribuan pesan sampel ke topik Kafka untuk memverifikasi catatan yang dicerna dalam Timestream untuk tabel. LiveAnalytics

Amazon QuickSight

Anda dapat menggunakan Amazon QuickSight untuk menganalisis dan mempublikasikan dasbor data yang berisi data Amazon Timestream Anda. Bagian ini menjelaskan bagaimana Anda dapat membuat koneksi sumber QuickSight data baru, mengubah izin, membuat kumpulan data baru, dan melakukan analisis. Tutorial video ini menjelaskan cara bekerja dengan Timestream dan Amazon QuickSight.

> **Note**
>
> Semua dataset di Amazon QuickSight adalah read-only. Anda tidak dapat membuat perubahan apa pun pada data aktual Anda di Timestream dengan menggunakan Amazon QuickSight untuk menghapus sumber data, kumpulan data, atau bidang.

Topik

- Mengakses Amazon Timestream dari QuickSight
- Buat koneksi sumber QuickSight data baru untuk Timestream
- Edit izin untuk koneksi sumber QuickSight data untuk Timestream
- Buat QuickSight dataset baru untuk Timestream
- Buat analisis baru untuk Timestream
- Video tutorial

Mengakses Amazon Timestream dari QuickSight

Sebelum Anda dapat melanjutkan, Amazon QuickSight harus diberi wewenang untuk terhubung ke Amazon Timestream. Jika koneksi tidak diaktifkan, Anda akan menerima kesalahan saat mencoba menghubungkan. QuickSight Administrator dapat mengotorisasi koneksi ke AWS sumber daya. Untuk mengotorisasi koneksi dari QuickSight Timestream, ikuti prosedur di Menggunakan AWS Layanan Lain: Scoping Down Access, pilih Amazon Timestream di langkah 5.

Buat koneksi sumber QuickSight data baru untuk Timestream

> **Note**
>
> Koneksi antara Amazon QuickSight dan Amazon Timestream dienkripsi dalam perjalanan menggunakan SSL (1.2). TLS Anda tidak dapat membuat koneksi yang tidak terenkripsi.

1. Pastikan Anda telah mengonfigurasi izin yang sesuai untuk Amazon QuickSight untuk mengakses Amazon Timestream, seperti yang dijelaskan dalam. Mengakses Amazon Timestream dari QuickSight
2. Mulailah dengan membuat dataset baru. Pilih Datasets dari panel navigasi, lalu pilih New Dataset.
3. Pilih kartu sumber data Timestream.
4. Untuk nama sumber Data, masukkan nama untuk koneksi sumber data Timestream Anda, misalnya`US Timestream Data`.

   > **Note**
   >
   > Karena Anda dapat membuat banyak kumpulan data dari koneksi ke Timestream, yang terbaik adalah menjaga namanya tetap sederhana.

5. Pilih Validasi koneksi untuk memeriksa apakah Anda berhasil terhubung ke Timestream.

   > **Note**
   >
   > Validasi koneksi hanya memvalidasi bahwa Anda dapat terhubung. Namun, itu tidak memvalidasi tabel atau kueri tertentu.

6. Pilih Buat sumber data untuk melanjutkan.

7. Untuk Database, pilih Pilih... untuk melihat daftar opsi yang tersedia. Pilih salah satu yang ingin Anda gunakan.
8. Pilih Pilih untuk melanjutkan.
9. Pilih salah satu cara berikut:
   - Untuk mengimpor data Anda ke QuickSight mesin dalam memori (disebutSPICE), pilih Impor ke SPICE untuk analitik yang lebih cepat.
   - QuickSight Untuk memungkinkan menjalankan kueri terhadap data Anda setiap kali Anda menyegarkan kumpulan data atau menggunakan analisis atau dasbor, pilih Kueri data Anda secara langsung.
10. Pilih Edit/Pratinjau lalu Simpan untuk menyimpan kumpulan data Anda dan menutupnya.

Edit izin untuk koneksi sumber QuickSight data untuk Timestream

Prosedur berikut menjelaskan cara melihat, menambah, dan mencabut izin untuk QuickSight pengguna lain sehingga mereka dapat mengakses sumber data Timestream yang sama. Orang-orang harus menjadi pengguna aktif QuickSight sebelum Anda dapat menambahkannya.

> **Note**
>
> Di QuickSight, sumber data memiliki dua tingkat izin: pengguna dan pemilik.
>
> - Pilih pengguna untuk mengizinkan akses baca.
> - Pilih pemilik untuk mengizinkan pengguna mengedit, membagikan, atau menghapus sumber QuickSight data ini.

1. Pastikan Anda telah mengonfigurasi izin yang sesuai untuk Amazon QuickSight untuk mengakses Amazon Timestream, seperti yang dijelaskan dalam. Mengakses Amazon Timestream dari QuickSight
2. Pilih Datasets di sebelah kiri, lalu gulir ke bawah untuk menemukan kartu sumber data untuk koneksi Timestream Anda. Sebagai contoh, `US Timestream Data`.
3. Pilih kartu sumber `Timestream` data.
4. Pilih `Share data source`. Daftar izin saat ini ditampilkan.
5. (Opsional) Untuk mengedit izin, Anda dapat memilih `user` atau`owner`.

6. (Opsional) Untuk mencabut izin, pilih. `Revoke access` Orang yang Anda cabut tidak dapat membuat kumpulan data baru dari sumber data ini. Namun, dataset mereka yang ada masih akan memiliki akses ke sumber data ini.
7. Untuk menambahkan izin, pilih`Invite users`, lalu ikuti langkah-langkah berikut untuk menambahkan pengguna:
    a. Tambahkan orang untuk memungkinkan mereka menggunakan sumber data yang sama.
    b. Untuk masing-masing, pilih `Permission` yang ingin Anda terapkan.
8. Setelah selesai, pilih`Close`.

Buat QuickSight dataset baru untuk Timestream

1. Pastikan Anda telah mengonfigurasi izin yang sesuai untuk Amazon QuickSight untuk mengakses Amazon Timestream, seperti yang dijelaskan dalam. Mengakses Amazon Timestream dari QuickSight
2. Pilih Datasets di sebelah kiri, lalu gulir ke bawah untuk menemukan kartu sumber data untuk koneksi Timestream Anda. Jika Anda memiliki banyak sumber data, Anda dapat menggunakan bilah pencarian di bagian atas halaman untuk menemukannya dengan kecocokan sebagian pada nama.
3. Pilih kartu sumber data Timestream. Kemudian pilih Buat kumpulan data.
4. Untuk Database, pilih Pilih untuk melihat daftar opsi yang tersedia. Pilih database yang ingin Anda gunakan.
5. Untuk Tabel, pilih tabel yang ingin Anda gunakan.
6. Pilih Edit/Pratinjau.
7. (Opsional) Untuk menambahkan lebih banyak data, pilih Tambahkan data di kanan atas.
    a. Pilih Beralih sumber data, dan pilih sumber data yang berbeda.
    b. Ikuti petunjuk UI untuk menyelesaikan penambahan data.
    c. Setelah menambahkan data baru ke kumpulan data yang sama, pilih Konfigurasikan gabungan ini (dua titik merah). Siapkan gabungan untuk setiap tabel tambahan.
    d. Jika Anda ingin menambahkan bidang terhitung, pilih Tambahkan bidang terhitung.
    e. Untuk menggunakan Sagemaker, pilih Augment with. SageMaker Opsi ini hanya tersedia dalam edisi QuickSight Enterprise.
    f. Hapus centang pada bidang apa pun yang ingin Anda hilangkan.

g. Perbarui tipe data apa pun yang ingin Anda ubah.

8. Setelah selesai, pilih Simpan untuk menyimpan dan menutup kumpulan data.

Buat analisis baru untuk Timestream

1. Pastikan Anda telah mengonfigurasi izin yang sesuai untuk Amazon QuickSight untuk mengakses Amazon Timestream, seperti yang dijelaskan dalam. Mengakses Amazon Timestream dari QuickSight
2. Pilih Analisis di sebelah kiri.
3. Pilih salah satu cara berikut:
   - Untuk membuat analisis baru, pilih Analisis baru di sebelah kanan.
   - Untuk menambahkan kumpulan data Timestream ke analisis yang ada, buka analisis yang ingin Anda edit. Pilih ikon pensil di dekat kiri atas, lalu Tambahkan kumpulan data.
4. Mulai visualisasi data pertama dengan memilih bidang di sebelah kiri.
5. Untuk informasi selengkapnya, lihat Bekerja dengan Analisis - Amazon QuickSight

Video tutorial

Video ini menjelaskan bagaimana Amazon QuickSight bekerja dengan Timestream.

Amazon SageMaker

Anda dapat menggunakan SageMaker Notebook Amazon untuk mengintegrasikan model pembelajaran mesin Anda dengan Amazon Timestream. Untuk membantu Anda memulai, kami telah membuat contoh SageMaker Notebook yang memproses data dari Timestream. Data dimasukkan ke Timestream dari aplikasi Python multi-threaded yang terus mengirim data. Kode sumber untuk SageMaker Notebook sampel dan contoh aplikasi Python tersedia di. GitHub

1. Buat database dan tabel mengikuti petunjuk yang dijelaskan dalam Buat database dan Membuat tabel
2. Kloning GitHub repositori untuk aplikasi sampel Python multi-threaded mengikuti instruksi dari GitHub
3. Kloning GitHub repositori untuk contoh Timestream SageMaker Notebook mengikuti instruksi dari. GitHub

4. Jalankan aplikasi untuk terus menelan data ke Timestream mengikuti instruksi di README
5. Ikuti petunjuk untuk membuat bucket Amazon S3 untuk Amazon SageMaker seperti yang dijelaskan di sini.
6. Buat SageMaker instance Amazon dengan boto3 terbaru yang diinstal: Selain petunjuk yang dijelaskan di sini, ikuti langkah-langkah di bawah ini:
   a. Pada halaman Create notebook instance, klik pada Additional Configuration
   b. Klik pada konfigurasi Siklus Hidup - opsional dan pilih Buat konfigurasi siklus hidup baru
   c. Pada kotak Create lifecycle configuration wizard, lakukan hal berikut:
      i. Isi nama yang diinginkan ke konfigurasi, mis. `on-start`
      ii. Di skrip Start Notebook, salin-tempel konten skrip dari Github
      iii. Ganti `PACKAGE=scipy` dengan `PACKAGE=boto3` skrip yang disisipkan.
7. Klik Buat konfigurasi
8. Buka IAM layanan di AWS Management Console dan temukan peran SageMaker eksekusi yang baru dibuat untuk instance notebook.
9. Lampirkan IAM kebijakan `AmazonTimestreamFullAccess` untuk peran eksekusi.

> **Note**
>
> `AmazonTimestreamFullAccess`IAMKebijakan ini tidak terbatas pada sumber daya tertentu dan tidak cocok untuk penggunaan produksi. Untuk sistem produksi, pertimbangkan untuk menggunakan kebijakan yang membatasi akses ke sumber daya tertentu.

10. Ketika status instance notebook InService, pilih Open Jupyter untuk meluncurkan SageMaker Notebook untuk instance
11. Unggah file `timestreamquery.py` dan `Timestream_SageMaker_Demo.ipynb` masuk ke Notebook dengan memilih tombol Unggah
12. Pilih `Timestream_SageMaker_Demo.ipynb`

> **Note**
>
> Jika Anda melihat pop up dengan Kernel tidak ditemukan, pilih conda_python3 dan klik Set Kernel.

13. Ubah `DB_NAME`,`TABLE_NAME`,`bucket`,, dan `ENDPOINT` untuk mencocokkan nama database, nama tabel, nama bucket S3, dan wilayah untuk model pelatihan.
14. Pilih ikon putar untuk menjalankan sel individual
15. Saat Anda masuk ke sel`Leverage Timestream to find hosts with average CPU utilization across the fleet`, pastikan bahwa output mengembalikan setidaknya 2 nama host.

    > **Note**
    >
    > Jika ada kurang dari 2 nama host dalam output, Anda mungkin perlu menjalankan kembali contoh aplikasi Python yang menyerap data ke Timestream dengan jumlah thread dan skala host yang lebih besar.

16. Saat Anda sampai di sel`Train a Random Cut Forest (RCF) model using the CPU utilization history`, ubah `train_instance_type` berdasarkan persyaratan sumber daya untuk pekerjaan pelatihan Anda
17. Saat Anda sampai di sel`Deploy the model for inference`, ubah `instance_type` berdasarkan persyaratan sumber daya untuk pekerjaan inferensi Anda

    > **Note**
    >
    > Mungkin perlu beberapa menit untuk melatih model. Ketika pelatihan selesai, Anda akan melihat pesan Selesai - Pekerjaan pelatihan selesai dalam output sel.

18. Jalankan sel `Stop and delete the endpoint` untuk membersihkan sumber daya. Anda juga dapat menghentikan dan menghapus instance dari SageMaker konsol

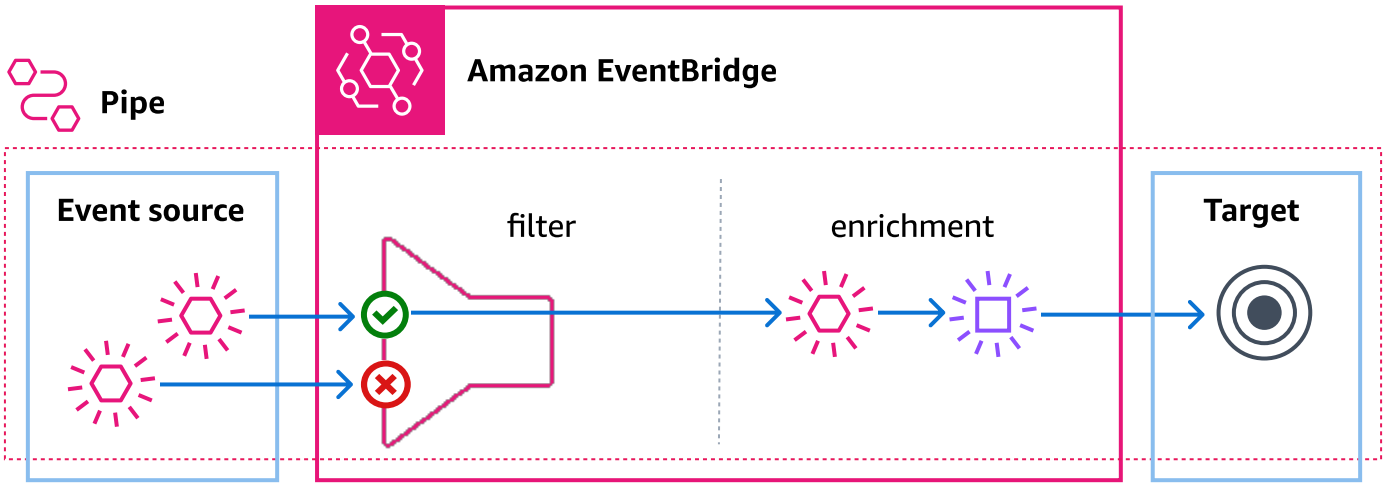
Amazon SQS

Menggunakan EventBridge Pipes untuk mengirim SQS data Amazon Timestream

Anda dapat menggunakan EventBridge Pipes untuk mengirim data dari SQS antrian Amazon ke Amazon Timestream LiveAnalytics untuk tabel.

Pipa dimaksudkan untuk point-to-point integrasi antara sumber dan target yang didukung, dengan dukungan untuk transformasi dan pengayaan lanjutan. Pipa mengurangi kebutuhan akan pengetahuan khusus dan kode integrasi saat mengembangkan arsitektur berbasis peristiwa.

Untuk menyiapkan pipa, Anda memilih sumber, menambahkan pemfilteran opsional, menentukan pengayaan opsional, dan memilih target untuk data peristiwa.



Untuk informasi lebih lanjut tentang EventBridge Pipa, lihat EventBridge Pipa di Panduan EventBridge Pengguna. Untuk informasi tentang mengonfigurasi pipa untuk mengirimkan peristiwa ke Amazon Timestream LiveAnalytics untuk tabel, EventBridge lihat Spesifikasi target Pipes.

Menggunakan DBeaver untuk bekerja dengan Amazon Timestream

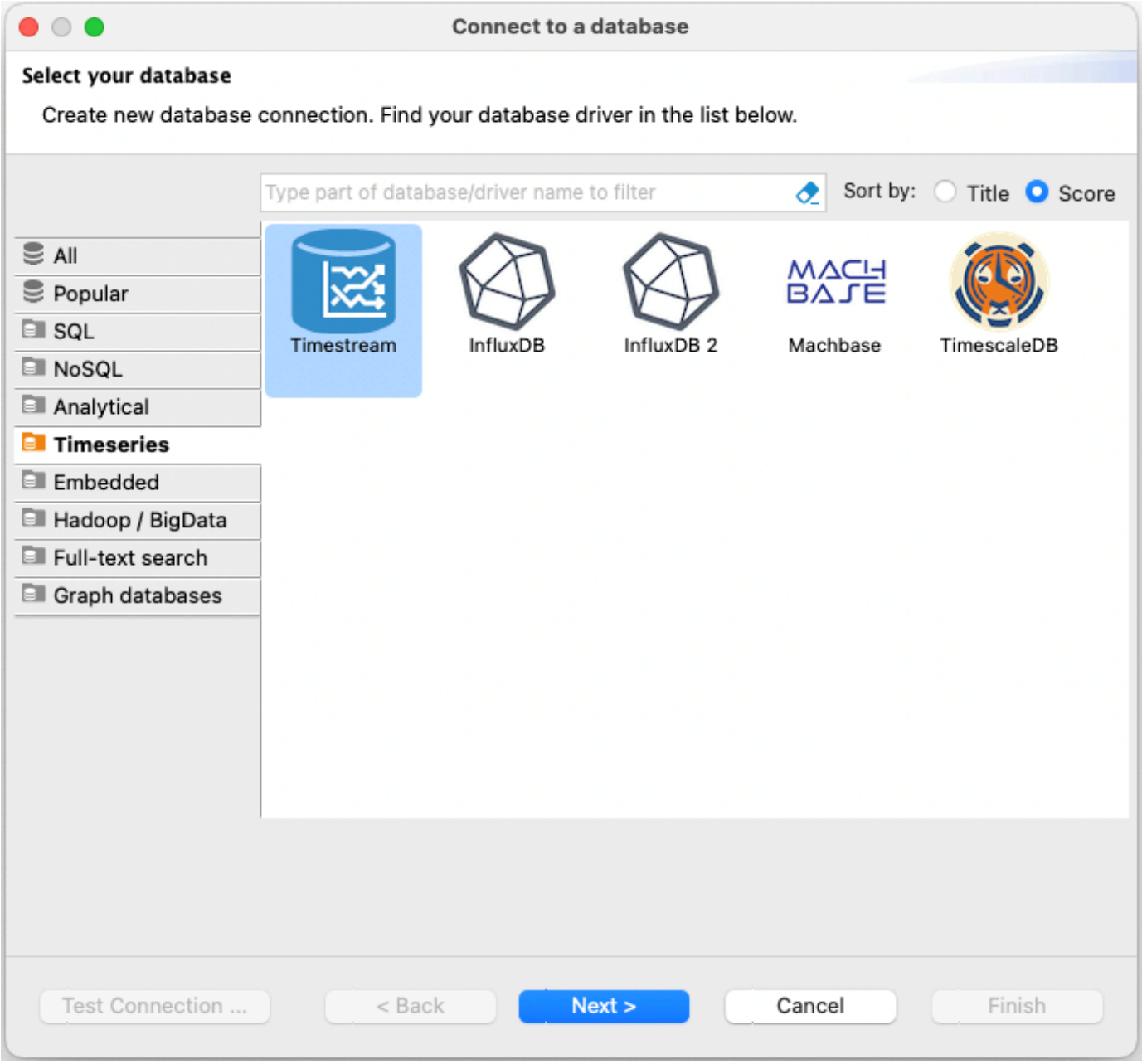
DBeaveradalah SQL klien universal gratis yang dapat digunakan untuk mengelola database apa pun yang memiliki JDBC driver. Ini banyak digunakan di kalangan pengembang dan administrator database karena kemampuan tampilan, pengeditan, dan manajemen datanya yang kuat.

Menggunakan DBeaver opsi konektivitas cloud, Anda dapat terhubung DBeaver ke Amazon Timestream secara native. DBeavermenyediakan antarmuka yang komprehensif dan intuitif untuk bekerja dengan data deret waktu langsung dari dalam DBeaver aplikasi. Menggunakan kredensil Anda, ini juga memberi Anda akses penuh ke kueri apa pun yang dapat Anda jalankan dari antarmuka kueri lain. Bahkan memungkinkan Anda membuat grafik untuk pemahaman dan visualisasi hasil kueri yang lebih baik.

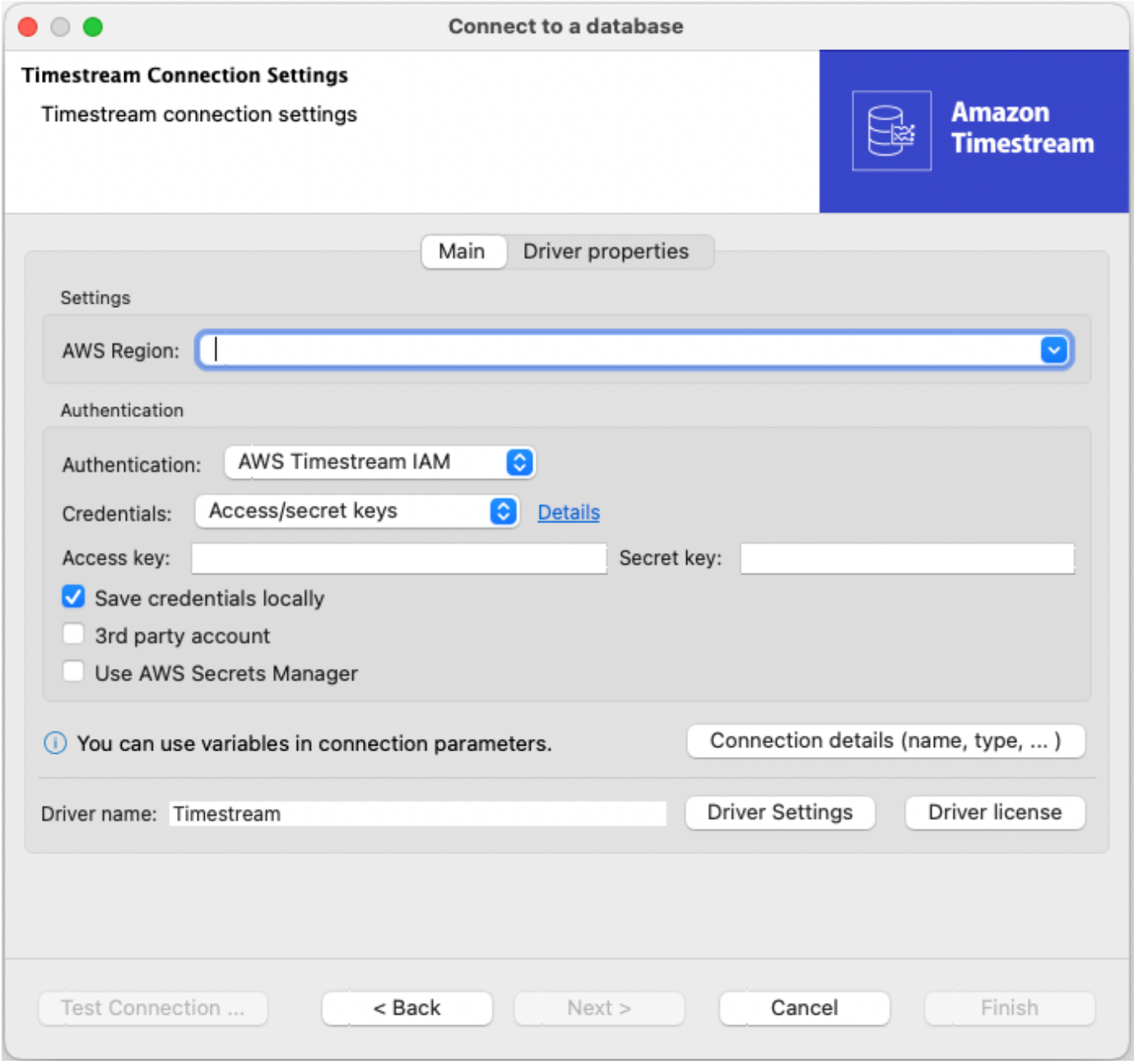
Menyiapkan DBeaver untuk bekerja dengan Timestream

Ambil langkah-langkah berikut untuk mengatur DBeaver agar bekerja dengan Timestream:

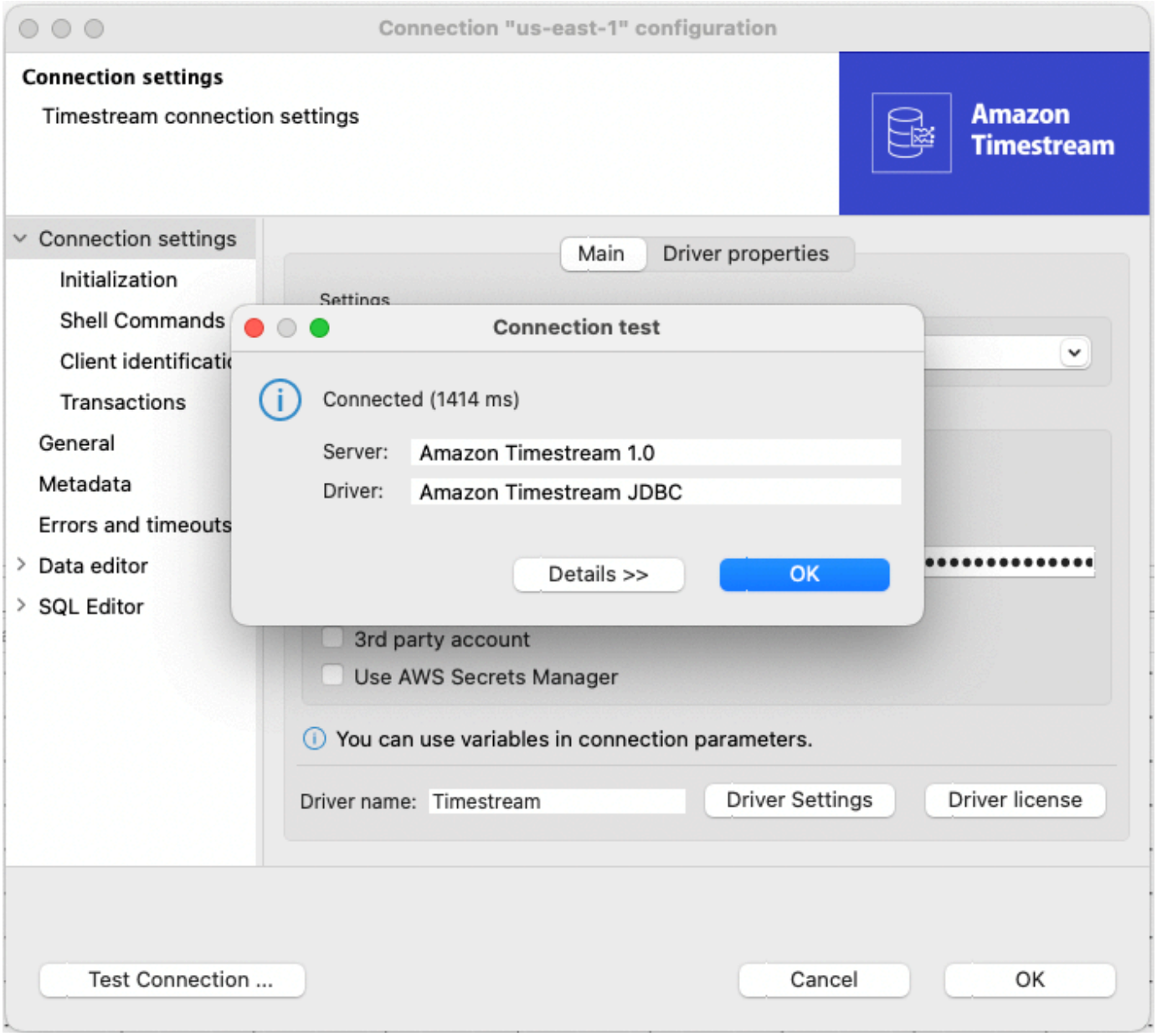
1. Unduh dan instal DBeaver di mesin lokal Anda.
2. LuncurkanDBeaver, navigasikan ke area pemilihan database, pilih Timeseries di panel kiri, lalu pilih ikon Timestream di panel kanan:



3. Di jendela Pengaturan Koneksi Timestream, masukkan semua informasi yang diperlukan untuk terhubung ke database Amazon Timestream Anda. Pastikan bahwa kunci pengguna yang Anda masukkan memiliki izin yang diperlukan untuk mengakses database Timestream Anda. Juga, pastikan untuk menyimpan informasi dan kunci yang Anda masukkan ke dalam DBeaver aman dan pribadi, seperti halnya informasi sensitif lainnya.



4. Uji koneksi untuk memastikan semuanya sudah diatur dengan benar:



5. Jika tes koneksi berhasil, Anda sekarang dapat berinteraksi dengan database Amazon Timestream Anda seperti yang Anda lakukan dengan database lain di. DBeaver Misalnya, Anda dapat menavigasi ke SQL editor atau ke tampilan Diagram ER untuk menjalankan kueri:

6. DBeaverjuga menyediakan alat visualisasi data yang kuat. Untuk menggunakannya, jalankan kueri Anda, lalu pilih ikon grafik untuk memvisualisasikan kumpulan hasil. Alat grafik dapat membantu Anda lebih memahami tren data dari waktu ke waktu.

Memasangkan Amazon Timestream DBeaver dengan menciptakan lingkungan yang efektif untuk mengelola data deret waktu. Anda dapat mengintegrasikannya dengan mulus ke dalam alur kerja yang ada untuk meningkatkan produktivitas dan efisiensi.

Grafana

Anda dapat memvisualisasikan data deret waktu Anda dan membuat peringatan menggunakan Grafana. Untuk membantu Anda memulai visualisasi data, kami telah membuat dasbor sampel di Grafana yang memvisualisasikan data yang dikirim ke Timestream dari aplikasi Python dan tutorial video yang menjelaskan pengaturan.

Topik

- Aplikasi sampel
- Video tutorial

Aplikasi sampel

1. Buat database dan tabel di Timestream mengikuti petunjuk yang dijelaskan Buat database untuk informasi lebih lanjut.

   > **Note**
   >
   > Nama database default dan nama tabel untuk dasbor Grafana diatur ke GrafanaDB dan masing-masing. grafanaTable Gunakan nama-nama ini untuk meminimalkan pengaturan.

2. Instal Python 3.7 atau lebih tinggi
3. Instal dan konfigurasikan Python Timestream SDK
4. Kloning GitHub repositori untuk aplikasi Python multi-thread yang terus menerus menelan data ke Timestream mengikuti instruksi dari GitHub
5. Jalankan aplikasi untuk terus menelan data ke Timestream mengikuti instruksi di README
6. Selesai Memulai dengan Grafana Terkelola Amazon atau selesaikan Instal Grafana.
7. Jika menginstal Grafana alih-alih menggunakan Grafana yang Dikelola Amazon, selesaikan Instal plugin Timestream untuk Grafana.

8. Buka dasbor Grafana menggunakan browser pilihan Anda. Jika Anda telah menginstal Grafana secara lokal, Anda dapat mengikuti instruksi yang dijelaskan dalam dokumentasi Grafana untuk masuk
9. Setelah meluncurkan Grafana, buka Datasources, klik Add Datasource, cari Timestream, dan pilih sumber data Timestream
10. Konfigurasikan Penyedia Auth dan wilayah dan klik Simpan dan Uji
11. Mengatur makro default
    a. Tetapkan $__database ke nama database Timestream Anda (misalnya GrafanaDB)
    b. Setel $__table ke nama tabel Timestream Anda (mis.) grafanaTable
    c. Setel $__measure ke ukuran yang paling umum digunakan dari tabl
12. Klik Simpan dan Uji
13. Klik pada tab Dasbor
14. Klik Impor untuk mengimpor dasbor
15. Klik dua kali Dasbor Aplikasi Contoh
16. Klik pada pengaturan dasbor
17. Pilih Variabel
18. Ubah dbName dan tableName untuk mencocokkan nama-nama database Timestream dan tabel
19. Klik Simpan
20. Segarkan dasbor
21. Untuk membuat lansiran, ikuti petunjuk yang dijelaskan dalam dokumentasi Grafana untuk Membuat aturan peringatan terkelola Grafana
22. Untuk memecahkan masalah peringatan, ikuti petunjuk yang dijelaskan dalam dokumentasi Grafana untuk Pemecahan Masalah
23. Untuk informasi tambahan, lihat dokumentasi Grafana

Video tutorial

Video ini menjelaskan cara Grafana bekerja dengan Timestream.

Menggunakan SquaredUp untuk bekerja dengan Amazon Timestream

SquaredUpadalah platform observabilitas yang terintegrasi dengan Amazon Timestream. Anda dapat menggunakan SquaredUp desainer dasbor intuitif untuk memvisualisasikan, menganalisis, dan

memantau data deret waktu Anda. Dasbor dapat dibagikan secara publik atau pribadi, dan saluran notifikasi dapat dibuat untuk mengingatkan Anda ketika kondisi kesehatan monitor berubah.

Menggunakan SquaredUp dengan Amazon Timestream

1. Mendaftar SquaredUpdan memulai secara gratis.
2. Tambahkan sumber AWS data.
3. Buat ubin dasbor yang menggunakan aliran data Timestream Query.
4. Secara opsional, aktifkan pemantauan ubin, buat saluran notifikasi, atau bagikan dasbor secara publik atau pribadi.
5. Secara opsional buat ubin lain untuk melihat data Timestream Anda bersama data dari alat pemantauan dan observabilitas Anda yang lain.

Telegraf sumber terbuka

Anda dapat menggunakan Timestream untuk plugin LiveAnalytics output untuk Telegraf untuk menulis metrik ke Timestream untuk LiveAnalytics langsung dari Telegraf open source.

Bagian ini memberikan penjelasan tentang cara menginstal Telegraf dengan Timestream untuk plugin LiveAnalytics output, cara menjalankan Telegraf dengan Timestream untuk plugin LiveAnalytics output, dan bagaimana Telegraf open source bekerja dengan Timestream untuk. LiveAnalytics

Topik

- Menginstal Telegraf dengan Timestream untuk Plugin Output LiveAnalytics
- Menjalankan Telegraf dengan Timestream untuk plugin output LiveAnalytics
- Memetakan metrik Telegraf/InfluxDB ke Timestream untuk model LiveAnalytics

Menginstal Telegraf dengan Timestream untuk Plugin Output LiveAnalytics

Pada versi 1.16, Timestream untuk plugin LiveAnalytics output tersedia dalam rilis Telegraf resmi. Untuk menginstal plugin output pada sebagian besar sistem operasi utama, ikuti langkah-langkah yang diuraikan dalam Dokumentasi InfluxData Telegraf. Untuk menginstal di Amazon Linux 2 OS, ikuti petunjuk di bawah ini.

Menginstal Telegraf dengan Timestream untuk plugin LiveAnalytics output di Amazon Linux 2

Untuk menginstal Telegraf dengan Plugin Output Timestream di Amazon Linux 2, lakukan langkah-langkah berikut.

1. Instal Telegraf menggunakan manajer `yum` paket.

```
cat <<EOF | sudo tee /etc/yum.repos.d/influxdb.repo
[influxdb]
name = InfluxDB Repository - RHEL \$releasever
baseurl = https://repos.influxdata.com/rhel/\$releasever/\$basearch/stable
enabled = 1
gpgcheck = 1
gpgkey = https://repos.influxdata.com/influxdb.key
EOF
```

2. Jalankan perintah berikut.

```
sudo sed -i "s/\$releasever/$(rpm -E %{rhel})/g" /etc/yum.repos.d/influxdb.repo
```

3. Instal dan mulai Telegraf.

```
sudo yum install telegraf
sudo service telegraf start
```

Menjalankan Telegraf dengan Timestream untuk plugin output LiveAnalytics

Anda dapat mengikuti petunjuk di bawah ini untuk menjalankan Telegraf dengan Timestream untuk plugin. LiveAnalytics

1. Hasilkan konfigurasi contoh menggunakan Telegraf.

```
telegraf --section-filter agent:inputs:outputs --input-filter cpu:mem --output-
filter timestream config > example.config
```

2. Buat database di Timestream menggunakan konsol manajemen, CLI, atau SDKs.
3. Dalam `example.config` file, tambahkan nama database Anda dengan mengedit kunci berikut di bawah `[[outputs.timestream]]` bagian.

```
database_name = "yourDatabaseNameHere"
```

4. Secara default, Telegraf akan membuat tabel. Jika Anda ingin membuat tabel secara manual, atur `create_table_if_not_exists` ke `false` dan ikuti petunjuk untuk membuat tabel menggunakan konsol manajemen, CLI, atau SDKs.
5. Dalam file example.config, konfigurasikan kredensional di bawah bagian. `[[outputs.timestream]]` Kredensi harus memungkinkan operasi berikut.

```
timestream:DescribeEndpoints
timestream:WriteRecords
```

> **Note**
>
> Jika Anda meninggalkan `create_table_if_not_exists` set ke`true`, sertakan:
>
> ```
> timestream:CreateTable
> ```

> **Note**
>
> Jika Anda mengatur `describe_database_on_start` ke`true`, sertakan yang berikut ini.
>
> ```
> timestream:DescribeDatabase
> ```

6. Anda dapat mengedit konfigurasi lainnya sesuai dengan preferensi Anda.
7. Setelah selesai mengedit file konfigurasi, jalankan Telegraf dengan yang berikut ini.

```
./telegraf --config example.config
```

8. Metrik akan muncul dalam beberapa detik, tergantung pada konfigurasi agen Anda. Anda juga harus melihat tabel baru, cpu dan mem, di konsol Timestream.

Memetakan metrik Telegraf/InfluxDB ke Timestream untuk model LiveAnalytics

Saat menulis data dari Telegraf ke Timestream untuk LiveAnalytics, data dipetakan sebagai berikut.

- Stempel waktu ditulis sebagai bidang waktu.

- Tag ditulis sebagai dimensi.
- Bidang ditulis sebagai ukuran.
- Pengukuran sebagian besar ditulis sebagai nama tabel (lebih lanjut tentang ini di bawah).

Plugin Timestream untuk LiveAnalytics output untuk Telegraf menawarkan beberapa opsi untuk mengatur dan menyimpan data di Timestream untuk. LiveAnalytics Ini dapat dijelaskan dengan contoh yang dimulai dengan data dalam format protokol baris.

```
weather,location=us-midwest,season=summer temperature=82,humidity=71
1465839830100400200 airquality,location=us-west no2=5,pm25=16
1465839830100400200
```

Berikut ini menjelaskan data.

- Nama pengukuran adalah `weather` dan`airquality`.
- Tag adalah `location` dan`season`.
- Bidang adalah`temperature`,`humidity`,`no2`, dan`pm25`.

Topik

- Menyimpan data dalam beberapa tabel
- Menyimpan data dalam satu tabel

Menyimpan data dalam beberapa tabel

Anda dapat memilih untuk membuat tabel terpisah per pengukuran dan menyimpan setiap bidang dalam baris terpisah per tabel.

Konfigurasinya adalah`mapping_mode = "multi-table"`.

- Timestream untuk LiveAnalytics adaptor akan membuat dua tabel, yaitu, `weather` dan`airquality`.
- Setiap baris tabel hanya akan berisi satu bidang.

Timestream yang dihasilkan untuk LiveAnalytics tabel, `weather` dan`airquality`, akan terlihat seperti ini.

`weather`

Waktu	lokasi	musim	ukuran_nama	ukuran_nilai: :bigint
2016-06-13 17:43:50	kami-midwest	musim panas	suhu	82
2016-06-13 17:43:50	kami-midwest	musim panas	kelembaban	71

`airquality`

Waktu	lokasi	ukuran_nama	ukuran_nilai: :bigint
2016-06-13 17:43:50	kami-midwest	no2	5
2016-06-13 17:43:50	kami-midwest	pm25	16

Menyimpan data dalam satu tabel

Anda dapat memilih untuk menyimpan semua pengukuran dalam satu tabel dan menyimpan setiap bidang dalam baris tabel terpisah.

Konfigurasinya adalah`mapping_mode = "single-table"`. Ada dua konfigurasi tambahan saat menggunakan`single-table`, `single_table_name` dan`single_table_dimension_name_for_telegraf_measurement_name`.

- Plugin Timestream untuk LiveAnalytics output akan membuat tabel tunggal dengan nama *`<single_table_name>`* yang meliputi *`<single_table_dimension_name_for_telegraf_measurement_name>`* kolom.
- Tabel mungkin berisi beberapa bidang dalam satu baris tabel.

Timestream yang dihasilkan untuk LiveAnalytics tabel akan terlihat seperti ini.

weather

Waktu	lokasi	musim	*<single_table_dimension_name_for_telegraf_measurement_name>*	ukuran_nama	ukuran_nilai: :bigint
2016-06-13 17:43:50	kami-midwest	musim panas	cuaca	suhu	82
2016-06-13 17:43:50	kami-midwest	musim panas	cuaca	kelembaban	71
2016-06-13 17:43:50	kami-midwest	musim panas	kualitas udara	no2	5
2016-06-13 17:43:50	kami-midwest	musim panas	cuaca	pm25	16

JDBC

Anda dapat menggunakan koneksi Java Database Connectivity (JDBC) untuk menghubungkan Timestream LiveAnalytics ke alat intelijen bisnis Anda dan aplikasi lain, seperti SQLWorkbench. Timestream untuk LiveAnalytics JDBC driver saat ini mendukung SSO dengan Okta dan Microsoft Azure AD.

Topik

- Mengkonfigurasi JDBC driver untuk Timestream untuk LiveAnalytics
- Properti koneksi
- JDBCURLcontoh
- Menyiapkan Timestream untuk otentikasi masuk LiveAnalytics JDBC tunggal dengan Okta
- Menyiapkan Timestream untuk LiveAnalytics JDBC otentikasi masuk tunggal dengan Microsoft Azure AD

Mengkonfigurasi JDBC driver untuk Timestream untuk LiveAnalytics

Ikuti langkah-langkah di bawah ini untuk mengkonfigurasi JDBC driver.

Topik

Timestream untuk driver LiveAnalytics JDBC JARs

Anda dapat memperoleh Timestream untuk LiveAnalytics JDBC driver melalui unduhan langsung atau dengan menambahkan driver sebagai ketergantungan Maven.

- Sebagai unduhan langsung:. Untuk langsung mengunduh Timestream untuk LiveAnalytics JDBC driver, selesaikan langkah-langkah berikut:

  1. Arahkan https://github.com/awslabs/amazon-timestream-driver-jdbcke/release
  2. Anda dapat menggunakan `amazon-timestream-jdbc-1.0.1-shaded.jar` langsung dengan alat dan aplikasi intelijen bisnis Anda
  3. Unduh `amazon-timestream-jdbc-1.0.1-javadoc.jar` ke direktori pilihan Anda.
  4. Di direktori tempat Anda mengunduh`amazon-timestream-jdbc-1.0.1-javadoc.jar`, jalankan perintah berikut untuk mengekstrak file JavadocHTML:

     ```
     jar -xvf amazon-timestream-jdbc-1.0.1-javadoc.jar
     ```

- Sebagai dependensi Maven: Untuk menambahkan Timestream untuk LiveAnalytics JDBC driver sebagai dependensi Maven, selesaikan langkah-langkah berikut:

  1. Arahkan ke dan buka `pom.xml` file aplikasi Anda di editor pilihan Anda.
  2. Tambahkan JDBC driver sebagai dependensi ke dalam `pom.xml` file aplikasi Anda:

     ```
     <!-- https://mvnrepository.com/artifact/software.amazon.timestream/amazon-
     timestream-jdbc -->
     <dependency>
         <groupId>software.amazon.timestream</groupId>
         <artifactId>amazon-timestream-jdbc</artifactId>
         <version>1.0.1</version>
     ```

```
</dependency>
```

Timestream untuk kelas dan URL format LiveAnalytics JDBC driver

Kelas driver untuk Timestream untuk LiveAnalytics JDBC driver adalah:

```
software.amazon.timestream.jdbc.TimestreamDriver
```

JDBCDriver Timestream memerlukan JDBC URL format berikut:

```
jdbc:timestream:
```

Untuk menentukan properti database melalui JDBCURL, gunakan URL format berikut:

```
jdbc:timestream://
```

Aplikasi sampel

Untuk membantu Anda memulai menggunakan Timestream for LiveAnalytics withJDBC, kami telah membuat contoh aplikasi yang berfungsi penuh di GitHub.

1. Buat database dengan data sampel mengikuti instruksi yang dijelaskan di sini.
2. Kloning GitHub repositori untuk aplikasi sampel untuk JDBC mengikuti instruksi dari. GitHub
3. Ikuti petunjuk di dalam READMEuntuk memulai dengan aplikasi sampel.

Properti koneksi

Timestream untuk LiveAnalytics JDBC driver mendukung opsi berikut:

Topik

- Opsi otentikasi dasar
- Opsi info klien standar
- Opsi konfigurasi driver
- SDKpilihan
- Opsi konfigurasi titik akhir

- Opsi penyedia kredensi
- SAMLopsi otentikasi berbasis untuk Okta
- SAMLopsi otentikasi berbasis untuk Azure AD

> **Note**
>
> Jika tidak ada properti yang disediakan, Timestream untuk LiveAnalytics JDBC driver akan menggunakan rantai kredensil default untuk memuat kredensil.

> **Note**
>
> Semua kunci properti peka huruf besar/kecil.

Opsi otentikasi dasar

Tabel berikut menjelaskan opsi Otentikasi Dasar yang tersedia.

Opsi	Deskripsi	Default
AccessKeyId	Id kunci akses AWS pengguna.	NONE
SecretAccessKey	Kunci akses rahasia AWS pengguna.	NONE
SessionToken	Token sesi sementara diperlukan untuk mengakses database dengan otentikasi multi-faktor (MFA) diaktifkan.	NONE

Opsi info klien standar

Tabel berikut menjelaskan Opsi Info Klien Standar.

Opsi	Deskripsi	Default
ApplicationName	Nama aplikasi yang saat ini menggunakan koneksi. `ApplicationName` digunakan untuk tujuan debugging dan tidak akan dikomunikasikan ke Timestream untuk layanan. LiveAnalytics	Nama aplikasi terdeteksi oleh pengemudi.

Opsi konfigurasi driver

Tabel berikut menjelaskan Opsi Konfigurasi Driver.

Opsi	Deskripsi	Default
EnableMetaDataPreparedStatement	Mengaktifkan Timestream bagi LiveAnalytics JDBC driver untuk mengembalikan metadata `PreparedStatements`, tetapi ini akan dikenakan biaya tambahan dengan Timestream saat mengambil metadata. LiveAnalytics	FALSE
Wilayah	Wilayah database.	us-east-1

SDKpilihan

Tabel berikut menjelaskan SDK Opsi.

Opsi	Deskripsi	Default
RequestTimeout	Waktu dalam milidetik AWS SDK akan menunggu permintaan kueri sebelum waktu habis. Nilai non-posit if menonaktifkan batas waktu permintaan.	0
SocketTimeout	Waktu dalam milidetik AWS SDK akan menunggu data ditransfer melalui koneksi terbuka sebelum waktu habis. Nilai harus non-negatif. Nilai 0 menonaktifkan batas waktu soket.	50000
MaxRetryCountClient	Jumlah maksimum upaya coba lagi untuk kesalahan yang dapat dicoba ulang dengan kode kesalahan 5XX di file. SDK Nilainya harus non-negatif.	NONE
MaxConnections	Jumlah maksimum HTTP koneksi yang diizinkan secara bersamaan dibuka ke Timestream untuk LiveAnaly tics layanan. Nilainya harus positif.	50

Opsi konfigurasi titik akhir

Tabel berikut menjelaskan Opsi Konfigurasi Titik Akhir.

Opsi	Deskripsi	Default
Titik Akhir	Titik akhir untuk Timestream untuk LiveAnalytics layanan.	NONE

Opsi penyedia kredensi

Tabel berikut menjelaskan opsi Penyedia Kredenal yang tersedia.

Opsi	Deskripsi	Default
AwsCredentialsProviderClass	Salah satu `PropertiesFileCredentialsProvider` atau `InstanceProfileCredentialsProvider` untuk digunakan untuk otentikasi.	NONE
CustomCredentialsFilePath	Path ke file properti yang berisi `accessKey` kredensi AWS keamanan dan file. `secretKey` Ini hanya diperlukan jika `AwsCredentialsProviderClass` ditentukan sebagai`PropertiesFileCredentialsProvider` .	NONE

SAMLopsi otentikasi berbasis untuk Okta

Tabel berikut menjelaskan opsi otentikasi SAML berbasis yang tersedia untuk Okta.

Opsi	Deskripsi	Default
IdpName	Nama Penyedia Identitas (Idp) yang akan digunakan untuk otentikasi SAML berbasis. Salah satu `Okta` atau`AzureAD`.	NONE
IdpHost	Nama host dari Idp yang ditentukan.	NONE
IdpUserName	Nama pengguna untuk akun Idp yang ditentukan.	NONE
IdpPassword	Kata sandi untuk akun Idp yang ditentukan.	NONE
OktaApplicationID	ID unik yang disediakan OKTA yang terkait dengan Timestream untuk aplikasi. LiveAnalytics `AppId`dapat ditemukan di `entityID` bidang yang disediakan dalam metadata aplikasi. Perhatika n contoh berikut: `entityID = http://www.okta.co m//IdpAppID`	NONE
Peran ARN	Amazon Resource Name (ARN) dari peran yang diasumsikan oleh penelepon.	NONE
Idp ARN	Nama Sumber Daya Amazon (ARN) dari SAML penyedia IAM yang menjelaskan Idp.	NONE

SAMLopsi otentikasi berbasis untuk Azure AD

Tabel berikut menjelaskan opsi otentikasi SAML berbasis yang tersedia untuk Azure AD.

Opsi	Deskripsi	Default
IdpName	Nama Penyedia Identitas (Idp) yang akan digunakan untuk otentikasi SAML berbasis. Salah satu `Okta` atau`AzureAD`.	NONE
IdpHost	Nama host dari Idp yang ditentukan.	NONE
IdpUserName	Nama pengguna untuk akun Idp yang ditentukan.	NONE
IdpPassword	Kata sandi untuk akun Idp yang ditentukan.	NONE
AADApplicationID	Id unik dari aplikasi terdaftar di Azure AD.	NONE
AADClientSecret	Rahasia klien yang terkait dengan aplikasi terdaftar di Azure AD digunakan untuk mengotorisasi pengambilan token.	NONE
AADTenant	ID Penyewa Azure AD.	NONE
Idp ARN	Nama Sumber Daya Amazon (ARN) dari SAML penyedia IAM yang menjelaskan Idp.	NONE

JDBCURLcontoh

Bagian ini menjelaskan cara membuat JDBC koneksiURL, dan memberikan contoh. Untuk menentukan properti koneksi opsional, gunakan URL format berikut:

```
jdbc:timestream://PropertyName1=value1;PropertyName2=value2...
```

> **Note**
>
> Semua properti koneksi adalah opsional. Semua kunci properti peka huruf besar/kecil.

Di bawah ini adalah beberapa contoh JDBC koneksiURLs.

Contoh dengan opsi otentikasi dasar dan wilayah:

```
jdbc:timestream://
AccessKeyId=<myAccessKeyId>;SecretAccessKey=<mySecretAccessKey>;SessionToken=<mySessionToken
east-1
```

Contoh dengan info klien, wilayah, dan SDK opsi:

```
jdbc:timestream://ApplicationName=MyApp;Region=us-
east-1;MaxRetryCountClient=10;MaxConnections=5000;RequestTimeout=20000
```

Connect menggunakan rantai penyedia kredensi default dengan AWS kredensi yang disetel dalam variabel lingkungan:

```
jdbc:timestream
```

Connect menggunakan rantai penyedia kredensi default dengan AWS kredensi yang disetel dalam sambungan: URL

```
jdbc:timestream://
AccessKeyId=<myAccessKeyId>;SecretAccessKey=<mySecretAccessKey>;SessionToken=<mySessionToken
```

Connect menggunakan PropertiesFileCredentialsProvider sebagai metode otentikasi:

```
jdbc:timestream://
AwsCredentialsProviderClass=PropertiesFileCredentialsProvider;CustomCredentialsFilePath=<pat
 to properties file>
```

Connect menggunakan InstanceProfileCredentialsProvider sebagai metode otentikasi:

```
jdbc:timestream://AwsCredentialsProviderClass=InstanceProfileCredentialsProvider
```

Connect menggunakan kredensi Okta sebagai metode otentikasi:

```
jdbc:timestream://
IdpName=Okta;IdpHost=<host>;IdpUserName=<name>;IdpPassword=<password>;OktaApplicationID=<id>
```

Connect menggunakan kredensi Azure AD sebagai metode otentikasi:

```
jdbc:timestream://
IdpName=AzureAD;IdpUserName=<name>;IdpPassword=<password>;AADApplicationID=<id>;AADClientSec
```

Connect dengan endpoint tertentu:

```
jdbc:timestream://Endpoint=abc.us-east-1.amazonaws.com;Region=us-east-1
```

Menyiapkan Timestream untuk otentikasi masuk LiveAnalytics JDBC tunggal dengan Okta

Timestream untuk LiveAnalytics mendukung Timestream untuk otentikasi masuk LiveAnalytics JDBC tunggal dengan Okta. Untuk menggunakan Timestream untuk otentikasi masuk LiveAnalytics JDBC tunggal dengan Okta, lengkapi setiap bagian yang tercantum di bawah ini.

Topik

- Prasyarat
- AWS federasi akun di Okta
- Menyiapkan Okta untuk SAML

Prasyarat

Pastikan Anda telah memenuhi prasyarat berikut sebelum menggunakan Timestream untuk otentikasi masuk LiveAnalytics JDBC tunggal dengan Okta:

- Izin admin AWS untuk membuat penyedia identitas dan peran.
- Akun Okta (Buka https://www.okta.com/login/untuk membuat akun).
- Akses ke Amazon Timestream untuk. LiveAnalytics

Sekarang setelah Anda menyelesaikan Prasyarat, Anda dapat melanjutkan ke. AWS federasi akun di Okta

AWS federasi akun di Okta

Timestream untuk LiveAnalytics JDBC driver mendukung Federasi AWS Akun di Okta. Untuk mengatur Federasi AWS Akun di Okta, selesaikan langkah-langkah berikut:

1. Masuk ke dasbor Admin Okta menggunakan yang berikut ini: URL

   ```
   https://<company-domain-name>-admin.okta.com/admin/apps/active
   ```

   > **Note**
   >
   > Ganti < company-domain-name > dengan nama domain Anda.

2. Setelah berhasil masuk, pilih Tambah Aplikasi dan cari Federasi AWS Akun.
3. Pilih Tambah
4. Ubah Login URL ke yang sesuaiURL.
5. Pilih Berikutnya
6. Pilih SAML2.0 Sebagai metode Sign-On
7. Pilih metadata Penyedia Identitas untuk membuka file XML metadata. Simpan file secara lokal.
8. Biarkan semua opsi konfigurasi lainnya kosong.
9. Pilih Selesai

Sekarang setelah Anda menyelesaikan Federasi AWS Akun di Okta, Anda dapat melanjutkan keMenyiapkan Okta untuk SAML.

Menyiapkan Okta untuk SAML

1. Pilih tab Masuk. Pilih Tampilan.
2. Pilih tombol Setup Instructions di bagian Pengaturan.

Menemukan dokumen metadata Okta

1. Untuk menemukan dokumen, buka:

```
https://<domain>-admin.okta.com/admin/apps/active
```

> **Note**
>
> <domain>adalah nama domain unik Anda untuk akun Okta Anda.

2. Pilih aplikasi Federasi AWS Akun
3. Pilih tab Masuk

Menyiapkan Timestream untuk LiveAnalytics JDBC otentikasi masuk tunggal dengan Microsoft Azure AD

Timestream untuk LiveAnalytics mendukung Timestream untuk otentikasi masuk LiveAnalytics JDBC tunggal dengan Microsoft Azure AD. Untuk menggunakan Timestream untuk LiveAnalytics JDBC autentikasi masuk tunggal dengan Microsoft Azure AD, lengkapi setiap bagian yang tercantum di bawah ini.

Topik

Prasyarat

Pastikan Anda telah memenuhi prasyarat berikut sebelum menggunakan Timestream untuk autentikasi LiveAnalytics JDBC masuk tunggal dengan Microsoft Azure AD:

- Izin admin AWS untuk membuat penyedia identitas dan peran.

- Akun Azure Active Directory (Buka https://azure.microsoft.com/en-ca/services/active-directory/ untuk membuat akun)
- Akses ke Amazon Timestream untuk. LiveAnalytics

Menyiapkan Azure AD

1. Masuk ke Azure Portal
2. Pilih Azure Active Directory dalam daftar layanan Azure. Ini akan mengarahkan ke halaman Direktori Default.
3. Pilih Aplikasi Perusahaan di bawah bagian Kelola di bilah sisi
4. Pilih + Aplikasi baru.
5. Temukan dan pilih Amazon Web Services.
6. Pilih Single Sign-On di bawah bagian Kelola di sidebar
7. Pilih SAMLsebagai metode masuk tunggal
8. Di bagian SAML Konfigurasi Dasar, masukkan yang berikut URL untuk Identifier dan BalasanURL:

```
https://signin.aws.amazon.com/saml
```

9. Pilih Simpan.
10. Unduh Metadata Federasi XML di bagian Sertifikat SAML Penandatanganan. Ini akan digunakan saat membuat Penyedia IAM Identitas nanti
11. Kembali ke halaman Direktori Default dan pilih Pendaftaran aplikasi di bawah Kelola.
12. Pilih Timestream untuk LiveAnalytics dari bagian Semua Aplikasi. Halaman akan diarahkan ke halaman Ikhtisar aplikasi

> **Note**
>
> Perhatikan ID Aplikasi (klien) dan ID Direktori (penyewa). Nilai-nilai ini diperlukan saat membuat koneksi.

13. Pilih Sertifikat dan Rahasia
14. Di bawah rahasia Klien, buat rahasia klien baru dengan+ Rahasia klien baru.

> **Note**
>
> Perhatikan rahasia klien yang dihasilkan, karena ini diperlukan saat membuat koneksi ke Timestream untuk LiveAnalytics.

15. Pada bilah sisi di bawah Kelola, pilih APIizin
16. Di Izin yang Dikonfigurasi, gunakan Tambahkan izin untuk memberikan izin Azure AD untuk masuk ke Timestream untuk. LiveAnalytics Pilih Microsoft Graph pada halaman API Permintaan izin.
17. Pilih izin yang didelegasikan dan pilih izin User.Read
18. Pilih Tambahkan izin
19. Pilih Berikan izin admin untuk Direktori Default

Menyiapkan Penyedia IAM Identitas dan peran di AWS

Selesaikan setiap bagian di bawah ini IAM untuk menyiapkan Timestream untuk LiveAnalytics JDBC otentikasi masuk tunggal dengan Microsoft Azure AD:

Topik

- Buat Penyedia SAML Identitas
- Buat IAM peran
- Buat IAM kebijakan
- Penyediaan

Buat Penyedia SAML Identitas

Untuk membuat Penyedia SAML Identitas Timestream untuk autentikasi masuk LiveAnalytics JDBC tunggal dengan Microsoft Azure AD, selesaikan langkah-langkah berikut:

1. Masuk ke Konsol AWS Manajemen
2. Pilih Layanan dan pilih IAMdi bawah Keamanan, Identitas, & Kepatuhan
3. Pilih Penyedia identitas di bawah Manajemen akses
4. Pilih Buat Penyedia dan pilih SAMLsebagai jenis penyedia. Masukkan Nama Penyedia. Contoh ini akan menggunakan AzureADProvider.
5. Unggah file Federation XML Metadata yang diunduh sebelumnya

6. Pilih Berikutnya, lalu pilih Buat.
7. Setelah selesai, halaman akan diarahkan kembali ke halaman Penyedia Identitas

Buat IAM peran

Untuk membuat IAM peran Timestream untuk autentikasi masuk LiveAnalytics JDBC tunggal dengan Microsoft Azure AD, selesaikan langkah-langkah berikut:

1. Di bilah sisi pilih Peran di bawah Manajemen akses
2. Pilih Buat peran
3. Pilih federasi SAML 2.0 sebagai entitas tepercaya
4. Pilih penyedia Azure AD
5. Pilih Izinkan akses Konsol Terprogram dan AWS Manajemen
6. Pilih Berikutnya: Izin
7. Lampirkan kebijakan izin atau lanjutkan ke Berikutnya:Tag
8. Tambahkan tag opsional atau lanjutkan ke Berikutnya:Tinjau
9. Masukkan Nama peran. Contoh ini akan menggunakan A zureSAMLRole
10. Berikan deskripsi peran
11. Pilih Buat Peran untuk menyelesaikan

Buat IAM kebijakan

Untuk membuat IAM kebijakan Timestream untuk autentikasi masuk LiveAnalytics JDBC tunggal dengan Microsoft Azure AD, selesaikan langkah-langkah berikut:

1. Di bilah sisi, pilih Kebijakan di bawah Manajemen akses
2. Pilih Buat kebijakan dan pilih JSONtab
3. Tambahkan kebijakan berikut

```
{
"Version": "2012-10-17",
"Statement": [
    {
          "Effect": "Allow",
          "Action": [
                "iam:ListRoles",
```

```
                "iam:ListAccountAliases"
            ],
            "Resource": "*"
        }
]
}
```

4. Pilih Buat kebijakan.
5. Masukkan nama kebijakan. Contoh ini akan digunakan TimestreamAccessPolicy.
6. Pilih Buat Kebijakan
7. Di bilah sisi, pilih Peran di bawah Manajemen akses.
8. Pilih peran Azure AD yang dibuat sebelumnya dan pilih Lampirkan kebijakan di bawah Izin.
9. Pilih kebijakan akses yang dibuat sebelumnya.

Penyediaan

Untuk menyediakan penyedia identitas Timestream untuk autentikasi masuk LiveAnalytics JDBC tunggal dengan Microsoft Azure AD, selesaikan langkah-langkah berikut:

1. Kembali ke Azure Portal
2. Pilih Azure Active Directory dalam daftar layanan Azure. Ini akan mengarahkan ke halaman Direktori Default
3. Pilih Aplikasi Perusahaan di bawah bagian Kelola di bilah sisi
4. Pilih Provisioning
5. Pilih mode Otomatis untuk Metode Penyediaan
6. Di bawah Kredensial Admin, masukkan AwsAccessKeyID Anda untuk clientsecret, dan untuk Token Rahasia SecretAccessKey
7. Atur Status Penyediaan ke Aktif
8. Pilih simpan. Ini memungkinkan Azure AD memuat Peran yang diperlukan IAM
9. Setelah status siklus saat ini selesai, pilih Pengguna dan grup di sidebar
10. Pilih + Tambahkan pengguna
11. Pilih pengguna Azure AD untuk menyediakan akses ke Timestream LiveAnalytics
12. Pilih peran IAM Azure AD dan Penyedia Identitas Azure terkait yang dibuat AWS
13. Pilih Tetapkan

ODBC

ODBCDriver open-source untuk Amazon Timestream LiveAnalytics untuk menyediakan SQL antarmuka -relasional ke Timestream LiveAnalytics untuk pengembang dan memungkinkan konektivitas dari alat intelijen bisnis (BI) seperti Power BI Desktop dan Microsoft Excel. Timestream untuk LiveAnalytics ODBC driver saat ini tersedia di Windows, macOS dan Linux, dan juga SSO mendukung dengan Okta dan Microsoft Azure Active Directory (AD).

Untuk informasi selengkapnya, lihat Amazon Timestream untuk dokumentasi LiveAnalytics ODBC driver. GitHub

Topik

- Menyiapkan Timestream untuk driver LiveAnalytics ODBC
- Sintaks string koneksi dan opsi untuk driver ODBC
- Contoh string koneksi untuk Timestream untuk driver LiveAnalytics ODBC
- Memecahkan masalah koneksi dengan driver ODBC

Menyiapkan Timestream untuk driver LiveAnalytics ODBC

Mengatur akses ke Timestream untuk LiveAnalytics di akun Anda AWS

Jika Anda belum menyiapkan AWS akun untuk bekerja dengan Timestream LiveAnalytics, ikuti insructions di. Mengakses Timestream untuk LiveAnalytics

Instal ODBC driver di sistem Anda

Unduh penginstal ODBC driver Timestream yang sesuai untuk sistem Anda dari ODBC GitHubrepositori, dan ikuti petunjuk penginstalan yang berlaku untuk sistem Anda:.

- Panduan instalasi Windows
- Panduan penginstalan macOS
- Panduan instalasi Linux

Siapkan nama sumber data (DSN) untuk ODBC driver

Ikuti petunjuk dalam panduan DSN konfigurasi untuk sistem Anda:

- DSNKonfigurasi Windows

- konfigurasi macOS DSN
- DSNKonfigurasi Linux

Siapkan aplikasi intelijen bisnis (BI) Anda untuk bekerja dengan ODBC pengemudi

Berikut adalah petunjuk untuk mengatur beberapa aplikasi BI umum untuk bekerja dengan ODBC driver:

- Menyiapkan Microsoft Power BI.
- Menyiapkan Microsoft Excel
- Menyiapkan Tableau

Untuk aplikasi lain

Sintaks string koneksi dan opsi untuk driver ODBC

Sintaks untuk menentukan opsi koneksi-string untuk driver adalah ODBC sebagai berikut:

```
DRIVER={Amazon Timestream ODBC Driver};(option)=(value);
```

Opsi yang tersedia adalah sebagai berikut:

Opsi koneksi driver

- **`Driver`**(wajib) — Pengemudi yang digunakan denganODBC.

  Defaultnya adalah Amazon Timestream.
- **`DSN`**— Nama sumber data (DSN) yang digunakan untuk mengkonfigurasi koneksi.

  Default-nya adalah `NONE`.
- **`Auth`**— Mode otentikasi. Ini harus salah satu dari yang berikut:
  - `AWS_PROFILE`— Gunakan rantai kredensi default.
  - `IAM`— Gunakan AWS IAM kredensional.
  - `AAD`— Gunakan penyedia identitas Azure Active Directory (AD).
  - `OKTA`— Gunakan penyedia identitas Okta.

  Default-nya adalah `AWS_PROFILE`.

Opsi konfigurasi titik akhir

- **`EndpointOverride`**— Pengesampingan titik akhir untuk Timestream untuk layanan. LiveAnalytics Ini adalah opsi lanjutan yang mengesampingkan wilayah. Sebagai contoh:

  ```
  query-cell2.timestream.us-east-1.amazonaws.com
  ```

- **`Region`**— Wilayah penandatanganan untuk Timestream untuk titik akhir LiveAnalytics layanan.

  Default-nya adalah `us-east-1`.

Opsi penyedia kredensial

- **`ProfileName`**— Nama profil dalam file AWS konfigurasi.

  Default-nya adalah `NONE`.

AWS IAMopsi otentikasi

- **`UID`**atau **`AccessKeyId`**— Id kunci akses AWS pengguna. Jika keduanya `UID` dan `AccessKeyId` disediakan dalam string koneksi, `UID` nilainya akan digunakan kecuali kosong.

  Default-nya adalah `NONE`.
- **`PWD`**atau **`SecretKey`**— Kunci akses rahasia AWS pengguna. Jika keduanya `PWD` dan `SecretKey` disediakan dalam string koneksi, `PWD` nilai dengan akan digunakan kecuali kosong.

  Default-nya adalah `NONE`.
- **`SessionToken`**— Token sesi sementara yang diperlukan untuk mengakses database dengan otentikasi multi-faktor (MFA) diaktifkan. Jangan sertakan trailing `=` dalam input.

  Default-nya adalah `NONE`.

SAMLopsi otentikasi berbasis untuk Okta

- **`IdPHost`**— Nama host dari iDP yang ditentukan.

  Default-nya adalah `NONE`.
- **`UID`**atau **`IdPUserName`**— Nama pengguna untuk akun iDP yang ditentukan. Jika keduanya `UID` dan `IdPUserName` disediakan dalam string koneksi, `UID` nilainya akan digunakan kecuali kosong.

Default-nya adalah NONE.

- **PWD**atau **IdPPassword**— Kata sandi untuk akun iDP yang ditentukan. Jika keduanya PWD dan IdPPassword disediakan dalam string koneksi, PWD nilainya akan digunakan kecuali kosong.

  Default-nya adalah NONE.

- **OktaApplicationID**— ID unik yang disediakan OKTA yang terkait dengan Timestream untuk aplikasi. LiveAnalytics Tempat untuk menemukan ID aplikasi (AppId) ada di entityID bidang yang disediakan dalam metadata aplikasi. Contohnya adalah:

  ```
  entityID="http://www.okta.com//(IdPAppID)
  ```

  Default-nya adalah NONE.

- **RoleARN**— Nama Sumber Daya Amazon (ARN) dari peran yang diasumsikan oleh penelepon.

  Default-nya adalah NONE.

- **IdPARN**— Nama Sumber Daya Amazon (ARN) dari SAML penyedia IAM yang menjelaskan IDP.

  Default-nya adalah NONE.

SAMLopsi otentikasi berbasis untuk Azure Active Directory

- **UID**atau **IdPUserName**— Nama pengguna untuk akun iDP yang ditentukan..

  Default-nya adalah NONE.

- **PWD**atau **IdPPassword**— Kata sandi untuk akun iDP yang ditentukan.

  Default-nya adalah NONE.

- **AADApplicationID**— Id unik dari aplikasi terdaftar di Azure AD.

  Default-nya adalah NONE.

- **AADClientSecret**— Rahasia klien yang terkait dengan aplikasi terdaftar di Azure AD digunakan untuk mengotorisasi pengambilan token.

  Default-nya adalah NONE.

- **AADTenant**— ID Penyewa Azure AD.

  Default-nya adalah NONE.

- **`RoleARN`**— Nama Sumber Daya Amazon (ARN) dari peran yang diasumsikan oleh penelepon.

  Default-nya adalah `NONE`.

- **`IdPARN`**— Nama Sumber Daya Amazon (ARN) dari SAML penyedia IAM yang menjelaskan IDP.

  Default-nya adalah `NONE`.

AWS SDK(lanjutan) Pilihan

- **`RequestTimeout`**— Waktu dalam milidetik AWS SDK menunggu permintaan kueri sebelum waktu habis. Setiap nilai non-positif menonaktifkan batas waktu permintaan.

  Default-nya adalah `3000`.

- **`ConnectionTimeout`**— Waktu dalam milidetik AWS SDK menunggu data ditransfer melalui koneksi terbuka sebelum waktu habis. Nilai 0 menonaktifkan batas waktu koneksi. Nilai ini tidak boleh negatif.

  Default-nya adalah `1000`.

- **`MaxRetryCountClient`**— Jumlah maksimum upaya coba lagi untuk kesalahan yang dapat dicoba ulang dengan kode kesalahan 5xx di. SDK Nilainya tidak boleh negatif.

  Default-nya adalah `0`.

- **`MaxConnections`**— Jumlah maksimum HTTP koneksi terbuka bersamaan yang diizinkan ke layanan Timestream. Nilainya harus positif.

  Default-nya adalah `25`.

ODBCOpsi pencatatan driver

- **`LogLevel`**— Tingkat log untuk logging driver. Harus menjadi salah satu dari:
  - 0 (OFF).
  - 1 (ERROR).
  - 2 (WARNING).
  - 3 (INFO).
  - 4 (DEBUG).

  Defaultnya adalah 1 (ERROR).

Peringatan: informasi pribadi dapat dicatat oleh pengemudi saat menggunakan mode DEBUG logging.

- **`LogOutput`**— Folder tempat menyimpan file log.

  Defaultnya adalah:

  - Windows:`%USERPROFILE%`, atau jika tidak tersedia,`%HOMEDRIVE%%HOMEPATH%`.
  - macOS dan Linux:`$HOME`, atau jika tidak tersedia, bidang `pw_dir` dari fungsi `getpwuid(getuid())` mengembalikan nilai.

SDKopsi pencatatan

Level AWS SDK log terpisah dari Timestream untuk tingkat log LiveAnalytics ODBC driver. Pengaturan yang satu tidak mempengaruhi yang lain.

Level SDK Log diatur menggunakan variabel lingkunganTS_AWS_LOG_LEVEL. Nilai yang valid adalah:

- `OFF`
- `ERROR`
- `WARN`
- `INFO`
- `DEBUG`
- `TRACE`
- `FATAL`

Jika tidak `TS_AWS_LOG_LEVEL` diatur, tingkat SDK log diatur ke default, yaituWARN.

Menghubungkan melalui proxy

ODBCDriver mendukung koneksi ke Amazon Timestream LiveAnalytics melalui proxy. Untuk menggunakan fitur ini, konfigurasikan variabel lingkungan berikut berdasarkan setelan proxy Anda:

- **`TS_PROXY_HOST`**— host proxy.
- **`TS_PROXY_PORT`**— Nomor port proxy.
- **`TS_PROXY_SCHEME`**— Skema proxy, baik `http` atau`https`.

- **TS_PROXY_USER**— Nama pengguna untuk otentikasi proxy.
- **TS_PROXY_PASSWORD**— Kata sandi pengguna untuk otentikasi proxy.
- **TS_PROXY_SSL_CERT_PATH**— File SSL Sertifikat yang digunakan untuk menghubungkan ke HTTPS proxy.
- **TS_PROXY_SSL_CERT_TYPE**— Jenis SSL sertifikat klien proxy.
- **TS_PROXY_SSL_KEY_PATH**— File kunci pribadi yang digunakan untuk menghubungkan ke HTTPS proxy.
- **TS_PROXY_SSL_KEY_TYPE**— Jenis file kunci pribadi yang digunakan untuk terhubung ke HTTPS proxy.
- **TS_PROXY_SSL_KEY_PASSWORD**— Frasa sandi ke file kunci pribadi yang digunakan untuk terhubung ke proxy. HTTPS

Contoh string koneksi untuk Timestream untuk driver LiveAnalytics ODBC

Contoh menghubungkan ke ODBC driver dengan IAM kredensil

```
Driver={Amazon Timestream ODBC Driver};Auth=IAM;AccessKeyId=(your access key
 ID);secretKey=(your secret key);SessionToken=(your session token);Region=us-east-2;
```

Contoh menghubungkan ke ODBC driver dengan profil

```
Driver={Amazon Timestream ODBC Driver};ProfileName=(the profile name);region=us-west-2;
```

Pengemudi akan mencoba untuk terhubung menggunakan kredensional yang disediakan di`~/.aws/credentials`, atau jika file ditentukan dalam variabel lingkunganAWS_SHARED_CREDENTIALS_FILE, menggunakan kredensional dalam file itu.

Contoh menghubungkan ke ODBC driver dengan Okta

```
driver={Amazon Timestream ODBC Driver};auth=okta;region=us-west-2;idPHost=(your host at
 Okta);idPUsername=(your user name);idPPassword=(your password);OktaApplicationID=(your
 Okta AppId);roleARN=(your role ARN);idPARN=(your Idp ARN);
```

Contoh menghubungkan ke ODBC driver dengan Azure Active Directory () AAD

```
driver={Amazon Timestream ODBC Driver};auth=aad;region=us-west-2;idPUsername=(your
 user name);idPPassword=(your password);aadApplicationID=(your AAD
```

```
 AppId);aadClientSecret=(your AAD client secret);aadTenant=(your AAD
 tenant);roleARN=(your role ARN);idPARN=(your idP ARN);
```

Contoh menghubungkan ke ODBC driver dengan titik akhir yang ditentukan dan level log 2 () WARNING

```
Driver={Amazon Timestream ODBC Driver};Auth=IAM;AccessKeyId=(your access
 key ID);secretKey=(your secret key);EndpointOverride=ingest.timestream.us-
west-2.amazonaws.com;Region=us-east-2;LogLevel=2;
```

Memecahkan masalah koneksi dengan driver ODBC

> **Note**
>
> Ketika nama pengguna dan kata sandi sudah ditentukan dalamDSN, tidak perlu menentukannya lagi ketika manajer ODBC driver memintanya.

Kode kesalahan `01S02` dengan pesan, `Re-writing (connection string option) (have you specified it several times?` terjadi ketika opsi string koneksi dilewatkan lebih dari sekali dalam string koneksi. Menentukan opsi lebih dari sekali menimbulkan kesalahan. Saat membuat koneksi dengan DSN dan string koneksi, jika opsi koneksi sudah ditentukan dalamDSN, jangan tentukan lagi di string koneksi.

VPCtitik akhir ()AWS PrivateLink

Anda dapat membuat koneksi pribadi antara Timestream Anda VPC dan Amazon LiveAnalytics dengan membuat titik akhir antarmuka VPC. Untuk informasi selengkapnya, lihat VPCtitik akhir ()AWS PrivateLink.

Praktik terbaik

Untuk sepenuhnya menyadari manfaat Amazon Timestream LiveAnalytics, ikuti praktik terbaik yang dijelaskan di bawah ini.

> **Note**
>
> Saat menjalankan proof-of-concept aplikasi, pertimbangkan jumlah data yang akan dikumpulkan aplikasi Anda selama beberapa bulan atau tahun sambil mengevaluasi kinerja

dan skala Timestream. LiveAnalytics Seiring pertumbuhan data Anda dari waktu ke waktu, Anda akan melihat bahwa kinerja Timestream untuk sebagian besar LiveAnalytics tetap tidak berubah karena arsitektur tanpa servernya dapat memanfaatkan paralelisme dalam jumlah besar untuk memproses volume data yang lebih besar dan secara otomatis menskalakan agar sesuai dengan kebutuhan aplikasi Anda.

Topik

Pemodelan data

Amazon Timestream for LiveAnalytics dirancang untuk mengumpulkan, menyimpan, dan menganalisis data deret waktu dari aplikasi dan perangkat yang memancarkan urutan data dengan stempel waktu. Untuk kinerja yang optimal, data yang dikirim ke Timestream LiveAnalytics harus memiliki karakteristik temporal dan waktu harus menjadi komponen klasik dari data.

Timestream untuk LiveAnalytics memberi Anda fleksibilitas untuk memodelkan data Anda dengan berbagai cara agar sesuai dengan kebutuhan aplikasi Anda. Di bagian ini, kami membahas beberapa pola ini dan memberikan panduan bagi Anda untuk mengoptimalkan biaya dan kinerja Anda. Biasakan diri Anda dengan Timestream utama untuk LiveAnalytics konsep seperti dimensi dan ukuran. Di bagian ini, Anda akan mempelajari lebih lanjut tentang:

Saat memutuskan apakah akan membuat satu tabel atau beberapa tabel untuk menyimpan data, pertimbangkan hal berikut:

- Data mana yang akan dimasukkan ke dalam tabel yang sama vs. ketika Anda ingin memisahkan data di beberapa tabel dan database.

- Cara memilih antara Timestream untuk catatan LiveAnalytics multi-ukuran vs. catatan ukuran tunggal, dan manfaat pemodelan menggunakan catatan multi-ukuran terutama saat aplikasi Anda melacak beberapa pengukuran secara instan.
- Atribut mana yang dimodelkan sebagai dimensi atau sebagai ukuran.
- Cara efektif menggunakan atribut nama ukuran untuk mengoptimalkan latensi kueri Anda.

Topik

Tabel tunggal vs. beberapa tabel

Saat Anda memodelkan data Anda dalam aplikasi, aspek penting lainnya adalah bagaimana memodelkan data ke dalam tabel dan database. Database dan tabel di Timestream for LiveAnalytics adalah abstraksi untuk kontrol akses, menentukan KMS kunci, periode retensi, dll. Timestream untuk LiveAnalytics secara otomatis mempartisi data Anda dan dirancang untuk menskalakan sumber daya agar sesuai dengan konsumsi, penyimpanan, dan beban kueri serta persyaratan untuk aplikasi Anda.

Tabel di Timestream untuk LiveAnalytics dapat menskalakan ke petabyte data yang disimpan, puluhan gigabit/detik data menulis, dan kueri dapat memproses ratusan per jam. TBs Kueri di Timestream untuk LiveAnalytics dapat menjangkau beberapa tabel dan database, menyediakan gabungan dan serikat pekerja untuk memberikan akses tanpa batas ke data Anda di beberapa tabel dan database. Jadi skala data atau volume permintaan biasanya bukan perhatian utama ketika memutuskan bagaimana mengatur data Anda di Timestream untuk LiveAnalytics. Di bawah ini adalah beberapa pertimbangan penting ketika memutuskan data mana yang akan ditempatkan bersama dalam tabel yang sama vs. dalam tabel yang berbeda, atau tabel dalam database yang berbeda.

- Kebijakan retensi data (retensi penyimpanan memori, retensi penyimpanan magnetik, dll.) didukung pada perincian tabel. Oleh karena itu, data yang memerlukan kebijakan retensi yang berbeda harus dalam tabel yang berbeda.

- AWS KMS kunci yang digunakan untuk mengenkripsi data Anda dikonfigurasi di tingkat database. Oleh karena itu, persyaratan kunci enkripsi yang berbeda menyiratkan data harus berada dalam basis data yang berbeda.
- Timestream untuk LiveAnalytics mendukung kontrol akses berbasis sumber daya pada perincian tabel dan database. Pertimbangkan persyaratan kontrol akses Anda saat memutuskan data mana yang Anda tulis ke tabel yang sama vs. tabel yang berbeda.
- Waspadai batasan jumlah dimensi, nama ukuran, dan nama atribut multi-ukuran saat memutuskan data mana yang disimpan di tabel mana.
- Pertimbangkan beban kerja kueri dan pola akses Anda saat memutuskan bagaimana Anda mengatur data Anda, karena latensi kueri dan kemudahan menulis kueri Anda akan bergantung pada hal itu.
  - Jika Anda menyimpan data yang sering Anda kueri dalam tabel yang sama, itu umumnya akan memudahkan cara Anda menulis kueri sehingga Anda sering dapat menghindari keharusan menulis gabungan, serikat pekerja, atau ekspresi tabel umum. Ini juga biasanya menghasilkan latensi kueri yang lebih rendah. Anda dapat menggunakan predikat pada dimensi dan mengukur nama untuk memfilter data yang relevan dengan kueri.

    Misalnya, pertimbangkan kasus di mana Anda menyimpan data dari perangkat yang terletak di enam benua. Jika kueri Anda sering mengakses data dari seluruh benua untuk mendapatkan tampilan agregat global, maka menyimpan data dari benua ini dalam tabel yang sama akan menghasilkan lebih mudah untuk menulis kueri. Di sisi lain, jika Anda menyimpan data pada tabel yang berbeda, Anda masih dapat menggabungkan data dalam kueri yang sama, namun, Anda perlu menulis kueri untuk menyatukan data dari seluruh tabel.
  - Timestream untuk LiveAnalytics menggunakan partisi adaptif dan pengindeksan pada data Anda. Jadi kueri hanya dikenakan biaya untuk data yang relevan dengan kueri Anda. Misalnya, jika Anda memiliki tabel yang menyimpan data dari satu juta perangkat di enam benua, jika kueri Anda memiliki predikat formulir `WHERE device_id = 'abcdef'` atau`WHERE continent = 'North America'`, maka kueri hanya dikenakan biaya untuk data untuk perangkat atau benua.
  - Jika memungkinkan, jika Anda menggunakan nama ukuran untuk memisahkan data dalam tabel yang sama yang tidak dipancarkan pada saat yang sama atau tidak sering ditanyakan, maka menggunakan predikat seperti `WHERE measure_name = 'cpu'` dalam kueri Anda, Anda tidak hanya mendapatkan manfaat pengukuran, Timestream untuk juga LiveAnalytics dapat secara efektif menghilangkan partisi yang tidak memiliki nama ukuran yang digunakan dalam predikat kueri Anda. Ini memungkinkan Anda untuk menyimpan data terkait dengan nama ukuran yang berbeda dalam tabel yang sama tanpa memengaruhi latensi atau biaya kueri, dan menghindari

penyebaran data ke beberapa tabel. Nama ukuran pada dasarnya digunakan untuk mempartisi data dan memangkas partisi yang tidak relevan dengan kueri.

Catatan multi-ukuran vs. catatan ukuran tunggal

Aliran waktu untuk LiveAnalytics memungkinkan Anda menulis data dengan beberapa ukuran per catatan (multi-ukuran) atau ukuran tunggal per catatan (ukuran tunggal).

Catatan multi-ukuran

Dalam banyak kasus penggunaan, perangkat atau aplikasi yang Anda lacak dapat memancarkan beberapa metrik atau peristiwa pada stempel waktu yang sama. Dalam kasus seperti itu, Anda dapat menyimpan semua metrik yang dipancarkan pada stempel waktu yang sama dalam catatan multi-ukuran yang sama. Artinya, semua ukuran yang disimpan dalam catatan multi-ukuran yang sama muncul sebagai kolom yang berbeda di baris data yang sama.

Pertimbangkan, misalnya, bahwa aplikasi Anda memancarkan metrik seperti cpu, memori, disk_iops dari perangkat yang diukur pada saat yang sama instan. Berikut ini adalah contoh tabel seperti itu di mana beberapa metrik yang dipancarkan pada saat yang sama instan disimpan di baris yang sama. Anda akan melihat dua host memancarkan metrik sekali setiap detik.

Nama host	ukuran_nama	Waktu	cpu	Memori	disk_iops
Host-24gju	metrik	2021-12-01 19:00:00	35	54,9	38.2
Host-24gju	metrik	2021-12-01 19:00:01	36	58	39
Host-28GJU	metrik	2021-12-01 19:00:00	15	55	92
Host-28GJU	metrik	2021-12-01 19:00:01	16	50	40

Catatan ukuran tunggal

Catatan ukuran tunggal cocok ketika perangkat Anda memancarkan metrik yang berbeda pada periode waktu yang berbeda, atau Anda menggunakan logika pemrosesan khusus yang

memancarkan metrics/events at different time periods (for instance, when a device's reading/state perubahan). Karena setiap ukuran memiliki stempel waktu yang unik, pengukuran dapat disimpan dalam catatan mereka sendiri di Timestream untuk. LiveAnalytics Misalnya, pertimbangkan sensor IoT, yang melacak suhu dan kelembaban tanah, yang memancarkan catatan hanya ketika mendeteksi perubahan dari entri yang dilaporkan sebelumnya. Contoh berikut memberikan contoh data tersebut yang dipancarkan menggunakan catatan ukuran tunggal.

device_id	ukuran_nama	Waktu	ukuran_nilai: :ganda	ukuran_nilai: :bigint
sensor-laut478	suhu	2021-12-01 19:22:32	35	NULL
sensor-laut478	suhu	2021-12-01 18:07:51	36	NULL
sensor-laut478	kelembaban	2021-12-01 19:05:30	NULL	21
sensor-laut478	kelembaban	2021-12-01 19:00:01	NULL	23

Membandingkan catatan ukuran tunggal dan multi-ukuran

Timestream for LiveAnalytics memberi Anda fleksibilitas untuk memodelkan data Anda sebagai catatan ukuran tunggal atau multi-ukuran tergantung pada persyaratan dan karakteristik aplikasi Anda. Satu tabel dapat menyimpan catatan ukuran tunggal dan multi-ukuran, jika persyaratan aplikasi Anda menginginkannya. Secara umum, ketika aplikasi Anda memancarkan beberapa ukuran/peristiwa secara instan, maka pemodelan data sebagai catatan multi-ukuran biasanya direkomendasikan untuk akses data berkinerja dan penyimpanan data yang hemat biaya.

Misalnya, jika Anda mempertimbangkan metrik dan peristiwa pelacakan kasus DevOps penggunaan dari ratusan ribu server, setiap server secara berkala memancarkan 20 metrik dan 5 peristiwa, di mana peristiwa dan metrik dipancarkan secara instan. Data tersebut dapat dimodelkan baik menggunakan catatan ukuran tunggal atau menggunakan catatan multi-ukuran (lihat generator data sumber terbuka untuk skema yang dihasilkan). Untuk kasus penggunaan ini, pemodelan data menggunakan catatan multi-ukuran dibandingkan dengan catatan ukuran tunggal menghasilkan:

- Pengukuran konsumsi - Catatan multi-ukuran menghasilkan sekitar 40% byte konsumsi yang lebih rendah yang ditulis.
- Batching konsumsi - Catatan multi-ukuran menghasilkan batch data yang lebih besar yang dikirim, yang menyiratkan klien membutuhkan lebih sedikit utas dan lebih CPU sedikit untuk memproses konsumsi.
- Pengukuran penyimpanan - Catatan multi-ukuran menghasilkan penyimpanan sekitar 8X lebih rendah, menghasilkan penghematan penyimpanan yang signifikan untuk penyimpanan memori dan magnetik.
- Latensi kueri - Rekaman multi-ukuran menghasilkan latensi kueri yang lebih rendah untuk sebagian besar jenis kueri jika dibandingkan dengan catatan ukuran tunggal.
- Query metered byte s - Untuk kueri yang memindai data kurang dari 10MB, catatan ukuran tunggal dan multi-ukuran sebanding. Untuk kueri yang mengakses ukuran tunggal dan pemindaian> data 10MB, catatan ukuran tunggal biasanya menghasilkan byte yang lebih rendah. Untuk kueri yang mereferensikan 3 ukuran atau lebih, catatan multi-ukuran menghasilkan pengukuran byte yang lebih rendah.
- Kemudahan mengekspresikan kueri multi-ukuran - Saat kueri Anda mereferensikan beberapa ukuran, pemodelan data Anda dengan catatan multi-ukuran menghasilkan lebih mudah untuk menulis kueri yang lebih ringkas.

Faktor-faktor sebelumnya akan bervariasi tergantung pada berapa banyak metrik yang Anda lacak, berapa banyak dimensi yang dimiliki data Anda, dll. Sementara contoh sebelumnya menyediakan beberapa data konkret untuk satu contoh, kita melihat di banyak skenario aplikasi dan kasus penggunaan di mana jika aplikasi Anda memancarkan beberapa ukuran pada saat yang sama, menyimpan data sebagai catatan multi-ukuran lebih efektif. Selain itu, catatan multi-ukuran memberi Anda fleksibilitas tipe data dan menyimpan beberapa nilai lain sebagai konteks (misalnya, menyimpan permintaanIDs, dan cap waktu tambahan, yang akan dibahas nanti).

Perhatikan bahwa catatan multi-ukuran juga dapat memodelkan ukuran jarang seperti contoh sebelumnya untuk catatan ukuran tunggal: Anda dapat menggunakan measure _name untuk menyimpan nama ukuran dan menggunakan nama atribut multi-ukuran generik, seperti value_double untuk menyimpan ukuran, value_bigint untuk menyimpan `DOUBLE` ukuran, value_timestamp untuk menyimpan nilai tambahan, dll. `BIGINT` `TIMESTAMP`

Dimensi dan ukuran

Tabel di Timestream untuk LiveAnalytics memungkinkan Anda menyimpan dimensi (mengidentifikasi atribut perangkat/data yang Anda simpan) dan ukuran (metrik/nilai yang Anda lacak), lihat Timestream untuk konsep untuk detail selengkapnya. LiveAnalytics Saat Anda memodelkan aplikasi di Timestream LiveAnalytics, cara Anda memetakan data ke dalam dimensi dan ukuran memengaruhi latensi konsumsi dan kueri Anda. Berikut ini adalah panduan tentang cara memodelkan data Anda sebagai dimensi dan ukuran yang dapat Anda terapkan pada kasus penggunaan Anda.

Memilih dimensi

Data yang mengidentifikasi sumber yang mengirimkan data deret waktu sangat cocok untuk dimensi, yang merupakan atribut yang tidak berubah seiring waktu. Misalnya, jika Anda memiliki metrik yang memancarkan server, maka atribut yang mengidentifikasi server, seperti nama host, Wilayah, rak, Availability Zone, adalah kandidat untuk dimensi. Demikian pula, untuk perangkat IoT dengan beberapa sensor yang melaporkan data deret waktu, id perangkat, id sensor, dll. Adalah kandidat untuk dimensi.

Jika Anda menulis data sebagai catatan multi-ukuran, dimensi dan atribut multi-ukuran muncul sebagai kolom dalam tabel saat Anda melakukan DESCRIBE atau menjalankan SELECT pernyataan di atas tabel. Oleh karena itu, saat menulis kueri Anda, Anda dapat dengan bebas menggunakan dimensi dan ukuran dalam kueri yang sama. Namun, saat Anda membuat catatan tulis untuk menyerap data, ingatlah hal berikut saat Anda memilih atribut mana yang ditentukan sebagai dimensi dan mana yang merupakan nilai ukuran:

- Nama dimensi, nilai, nama ukuran, dan stempel waktu secara unik mengidentifikasi data deret waktu. Timestream untuk LiveAnalytics menggunakan pengenal unik ini untuk menghapus duplikasi data secara otomatis. Artinya, jika Timestream untuk LiveAnalytics menerima dua titik data dengan nilai yang sama dari nama dimensi, nilai dimensi, nama ukuran, dan stempel waktu, jika nilai memiliki nomor versi yang sama, maka Timestream untuk deduplikat. LiveAnalytics Jika permintaan tulis baru memiliki versi yang lebih rendah dari data yang sudah ada di Timestream for LiveAnalytics, permintaan tulis ditolak. Jika permintaan tulis baru memiliki versi yang lebih tinggi, maka nilai baru menimpa nilai lama. Oleh karena itu, bagaimana Anda memilih nilai dimensi Anda akan memengaruhi perilaku de-duplikasi ini.
- Nama dan nilai dimensi tidak dapat diperbarui, nilai ukuran bisa. Jadi data apa pun yang mungkin memerlukan pembaruan lebih baik dimodelkan sebagai nilai ukuran. Misalnya, jika Anda memiliki mesin di lantai pabrik yang warnanya dapat berubah, Anda dapat memodelkan warna sebagai nilai ukuran, kecuali jika Anda ingin menggunakan warna juga sebagai atribut pengidentifikasi yang

diperlukan untuk deduplikasi. Artinya, nilai ukuran dapat digunakan untuk menyimpan atribut yang hanya perlahan berubah seiring waktu.

Perhatikan bahwa tabel di Timestream untuk LiveAnalytics tidak membatasi jumlah kombinasi unik nama dimensi dan nilai. Misalnya, Anda dapat memiliki miliaran kombinasi nilai unik yang disimpan dalam tabel. Namun, seperti yang akan Anda lihat dengan contoh berikut, pilihan dimensi dan ukuran yang cermat dapat mengoptimalkan latensi permintaan Anda secara signifikan, terutama untuk kueri.

Unik IDs dalam dimensi

Jika skenario aplikasi Anda mengharuskan Anda menyimpan pengenal unik untuk setiap titik data (misalnya, ID permintaan, ID transaksi, atau ID korelasi), pemodelan atribut ID sebagai nilai ukuran akan menghasilkan latensi kueri yang jauh lebih baik. Saat memodelkan data Anda dengan catatan multi-ukuran, ID akan muncul di baris yang sama dalam konteks dengan dimensi dan data deret waktu Anda yang lain, sehingga kueri Anda dapat terus menggunakannya secara efektif. Misalnya, mempertimbangkan kasus DevOps penggunaan di mana setiap titik data yang dipancarkan oleh server memiliki atribut ID permintaan unik, pemodelan ID permintaan sebagai nilai ukuran menghasilkan latensi kueri hingga 4x lebih rendah di berbagai jenis kueri, sebagai lawan pemodelan ID permintaan unik sebagai dimensi.

Anda dapat menggunakan analogi serupa untuk atribut yang tidak sepenuhnya unik untuk setiap titik data, tetapi memiliki ratusan ribu atau jutaan nilai unik. Anda dapat memodelkan atribut tersebut baik sebagai dimensi atau mengukur nilai. Anda ingin memodelkannya sebagai dimensi jika nilai diperlukan untuk de-duplikasi pada jalur tulis seperti yang dibahas sebelumnya atau Anda sering menggunakannya sebagai predikat (misalnya, dalam `WHERE` klausa dengan predikat kesetaraan pada nilai atribut itu seperti di `device_id = ‘abcde’` mana aplikasi Anda melacak jutaan perangkat) dalam kueri Anda.

Kekayaan tipe data dengan catatan multi-ukuran

Catatan multi-ukuran memberi Anda fleksibilitas untuk memodelkan data Anda secara efektif. Data yang Anda simpan dalam catatan multi-ukuran muncul sebagai kolom dalam tabel yang mirip dengan dimensi, sehingga memberikan kemudahan kueri yang sama untuk dimensi dan nilai pengukuran. Anda melihat beberapa pola ini dalam contoh yang dibahas sebelumnya. Di bawah ini Anda akan menemukan pola tambahan untuk menggunakan catatan multi-ukuran secara efektif untuk memenuhi kasus penggunaan aplikasi Anda.

Rekaman multi-ukuran mendukung atribut tipe data`DOUBLE`,,`BIGINT`, `VARCHAR``BOOLEAN`, dan`TIMESTAMP`. Oleh karena itu, mereka secara alami cocok dengan berbagai jenis atribut:

- Informasi lokasi: Misalnya, jika Anda ingin melacak lokasi (dinyatakan sebagai garis lintang dan bujur), maka pemodelannya sebagai atribut multi-ukuran akan menghasilkan latensi kueri yang lebih rendah dibandingkan dengan menyimpannya sebagai VARCHAR dimensi, terutama ketika Anda memiliki predikat pada garis lintang dan bujur.
- Beberapa stempel waktu dalam catatan: Jika skenario aplikasi Anda mengharuskan Anda melacak beberapa stempel waktu untuk catatan deret waktu, Anda dapat memodelkannya sebagai atribut tambahan dalam catatan multi-ukuran. Pola ini dapat digunakan untuk menyimpan data dengan stempel waktu masa depan atau stempel waktu sebelumnya. Perhatikan bahwa setiap catatan masih akan menggunakan stempel waktu di kolom waktu untuk mempartisi, mengindeks, dan mengidentifikasi catatan secara unik.

Secara khusus, jika Anda memiliki data numerik atau stempel waktu di mana Anda memiliki predikat dalam kueri, pemodelan atribut tersebut sebagai atribut multi-ukuran sebagai lawan dari dimensi akan menghasilkan latensi kueri yang lebih rendah. Ini karena ketika Anda memodelkan data tersebut menggunakan tipe data kaya yang didukung dalam catatan multi-ukuran, Anda dapat mengekspresikan predikat menggunakan tipe data asli alih-alih mentransmisikan nilai dari VARCHAR ke tipe data lain jika Anda memodelkan data tersebut sebagai dimensi.

Menggunakan nama ukuran dengan catatan multi-ukuran

Tabel di Timestream untuk LiveAnalytics mendukung atribut khusus (atau kolom) yang disebut nama ukuran. Anda menentukan nilai untuk atribut ini untuk setiap catatan yang Anda tulis ke Timestream. LiveAnalytics Untuk catatan ukuran tunggal, wajar untuk menggunakan nama metrik Anda (seperti cpu, memori untuk metrik server, atau suhu, tekanan untuk metrik sensor). Saat menggunakan catatan multi-ukuran, karena atribut dalam catatan multi-ukuran diberi nama (dan nama-nama ini menjadi nama kolom dalam tabel), cpu, memori atau suhu, tekanan dapat menjadi nama atribut multi-ukuran. Jadi pertanyaan alami adalah bagaimana menggunakan nama ukuran secara efektif.

Timestream untuk LiveAnalytics menggunakan nilai-nilai dalam atribut nama ukuran untuk partisi dan indeks data. Oleh karena itu, jika tabel memiliki beberapa nama ukuran yang berbeda, dan jika kueri menggunakan nilai tersebut sebagai predikat kueri, maka Timestream for LiveAnalytics dapat menggunakan partisi dan pengindeksan khusus untuk memangkas data yang tidak relevan dengan kueri. Misalnya, jika tabel Anda memiliki nama cpu dan ukuran memori, dan kueri Anda memiliki predikat`WHERE measure_name = 'cpu'`, Timestream for LiveAnalytics dapat secara efektif memangkas data untuk nama ukuran yang tidak relevan dengan kueri, misalnya, baris dengan memori nama ukuran dalam contoh ini. Pemangkasan ini berlaku bahkan saat menggunakan nama ukuran dengan catatan multi-ukuran. Anda dapat menggunakan atribut nama ukuran secara

efektif sebagai atribut partisi untuk tabel. Ukur nama bersama dengan nama dimensi dan nilai, dan waktu digunakan untuk mempartisi data dalam Timestream untuk LiveAnalytics tabel. Waspadai batasan jumlah nama ukuran unik yang diizinkan dalam Timestream untuk LiveAnalytics tabel. Perhatikan juga bahwa nama ukuran dikaitkan dengan tipe data nilai ukuran juga, misalnya, nama ukuran tunggal hanya dapat dikaitkan dengan satu jenis nilai ukuran. Jenis itu bisa menjadi salah satu`DOUBLE`,`BIGINT`,`BOOLEAN`,`VARCHAR`, dan`MULTI`. Rekaman multi-ukuran yang disimpan dengan nama ukuran akan memiliki tipe data sebagai`MULTI`. Karena satu catatan multi-ukuran dapat menyimpan beberapa metrik dengan tipe data yang berbeda (`DOUBLE`,,,`BIGINT`, `VARCHAR``BOOLEAN`, dan`TIMESTAMP`), Anda dapat mengaitkan data dari berbagai jenis dalam catatan multi-ukuran.

Bagian berikut menjelaskan beberapa contoh berbeda tentang bagaimana atribut nama ukuran dapat digunakan secara efektif untuk mengelompokkan berbagai jenis data dalam tabel yang sama.

Sensor IoT melaporkan kualitas dan nilai

Pertimbangkan Anda memiliki data pemantauan aplikasi dari sensor IoT. Setiap sensor melacak ukuran yang berbeda, seperti suhu, tekanan. Selain nilai aktual, sensor juga melaporkan kualitas pengukuran, yang merupakan ukuran seberapa akurat pembacaan, dan unit untuk pengukuran. Karena kualitas, unit, dan nilai dipancarkan bersama-sama, mereka dapat dimodelkan sebagai catatan multi-ukuran, seperti yang ditunjukkan dalam contoh data di bawah ini di mana device_id adalah dimensi, kualitas, nilai, dan unit adalah atribut multi-ukuran:

device_id	ukuran_nama	Waktu	Kualitas	Nilai	Unit
sensor-la ut478	suhu	2021-12-01 19:22:32	92	35	c
sensor-la ut478	suhu	2021-12-01 18:07:51	93	34	c
sensor-la ut478	tekanan	2021-12-01 19:05:30	98	31	psi
sensor-la ut478	tekanan	2021-12-01 19:00:01	24	132	psi

Pendekatan ini memungkinkan Anda untuk menggabungkan manfaat catatan multi-ukuran bersama dengan partisi dan pemangkasan data menggunakan nilai nama ukuran. Jika kueri mereferensikan

satu ukuran, misalnya suhu, maka Anda dapat menyertakan predikat nama ukuran dalam kueri. Berikut ini adalah contoh kueri semacam itu, yang juga memproyeksikan unit untuk pengukuran yang kualitasnya di atas 90.

```
SELECT device_id, time, value AS temperature, unit
FROM db.table
WHERE time > ago(1h)
    AND measure_name = 'temperature'
    AND quality > 90
```

Menggunakan predikat measurement _name pada kueri memungkinkan Timestream untuk secara efektif LiveAnalytics memangkas partisi dan data yang tidak relevan dengan kueri, sehingga meningkatkan latensi kueri Anda.

Dimungkinkan juga untuk menyimpan semua metrik dalam catatan multi-ukuran yang sama jika semua metrik dipancarkan pada stempel waktu yang sama dan/atau beberapa metrik ditanyakan bersama dalam kueri yang sama. Misalnya, Anda dapat membuat catatan multi-ukuran dengan atribut temperature_quality, temperature_value, temperature_unit, pressure_quality, pressure_value, pressure_unit, dll. Banyak poin yang dibahas sebelumnya tentang pemodelan data menggunakan catatan ukuran tunggal vs. multi-ukuran berlaku dalam keputusan Anda tentang cara memodelkan data. Pertimbangkan pola akses kueri Anda dan bagaimana data Anda dihasilkan untuk memilih model yang mengoptimalkan biaya, konsumsi dan latensi kueri, serta kemudahan menulis kueri Anda.

Berbagai jenis metrik dalam tabel yang sama

Kasus penggunaan lain di mana Anda dapat menggabungkan catatan multi-ukuran dengan nilai nama ukuran adalah memodelkan berbagai jenis data yang dipancarkan secara independen dari perangkat yang sama. Pertimbangkan bahwa server kasus penggunaan DevOps pemantauan memancarkan dua jenis data: metrik yang dipancarkan secara teratur dan peristiwa tidak teratur. Contoh dari pendekatan ini adalah skema yang dibahas dalam pemodelan generator data kasus DevOps penggunaan. Dalam hal ini, Anda dapat menyimpan berbagai jenis data yang dipancarkan dari server yang sama dalam tabel yang sama dengan menggunakan nama ukuran yang berbeda. Misalnya, semua metrik yang dipancarkan pada saat yang sama secara instan disimpan dengan metrik nama ukuran. Semua peristiwa yang dipancarkan pada waktu yang berbeda dari metrik disimpan dengan peristiwa nama ukuran. Skema ukuran untuk tabel (misalnya, keluaran `SHOW MEASURES` kueri) adalah:

ukuran_nama	data_type	Dimensi
peristiwa	multi	[{"data_type” :"varchar”, "dimension_name” :"availab ility_zone "}, {" data_type ” :"varchar”, "dimensio n_name” :"microservice_nam e "}, {" data_type” :"varchar”, "dimension_name” :"instanc e_name "}, {" data_type ” :"instance_name "}, {" data_type” a_type” :"varchar”, "dimension_name” :"process _name "}, {" data_type ” :"varchar”, "dimensio n_name” :"jdk_version "}, {" data_type” :"varchar”, "dimension_name” :"cell "}, {" data_type” :"varchar”, "dimension_name” :"region "}, {" data_type” :"varchar”, "dimension_name” :"silo "}]
metrik	multi	[{"data_type” :"varchar”, "dimension_name” :"availab ility_zone "}, {" data_type ” :"varchar”, "dimensio n_name” :"microservice_nam e "}, {" data_type” :"varchar”, "dimension_name” :"instanc e_name "}, {" data_type ” :"instance_name "}, {" data_type” a_type” :"varchar”, "dimension_name” :"os_vers ion "}, {" data_type” :"varchar ”, "dimension_name” :"cell

ukuran_nama	data_type	Dimensi
		"}, {" data_type” :"varchar”, "dimension_name” :"region "}, {" data_type” :"varchar”, "dimension_name” :"varchar”, "dimension_name” :"varchar” _name” :"silo "}, {" data_type ” :"varchar”, "dimensio n_name” :"instance_type "}]

Dalam hal ini, Anda dapat melihat bahwa peristiwa dan metrik juga memiliki kumpulan dimensi yang berbeda, di mana peristiwa memiliki dimensi yang berbeda jdk_version dan process_name sedangkan metrik memiliki dimensi instance_type dan os_version.

Menggunakan nama ukuran yang berbeda memungkinkan Anda menulis kueri dengan predikat seperti hanya `WHERE measure_name = 'metrics'` mendapatkan metrik. Juga memiliki semua data yang dipancarkan dari instance yang sama dalam tabel yang sama menyiratkan Anda juga dapat menulis kueri yang lebih sederhana dengan predikat instance_name untuk mendapatkan semua data untuk instance itu. Misalnya, predikat formulir `WHERE instance_name = ‘instance-1234’` tanpa predikat measurement _name akan mengembalikan semua data untuk instance server tertentu.

Rekomendasi untuk mempartisi catatan multi-ukuran

> **Important**
>
> Bagian ini sudah usang!
> Rekomendasi ini sudah ketinggalan zaman. Partisi sekarang lebih baik dikontrol menggunakan tombol partisi yang ditentukan pelanggan.

Kami telah melihat bahwa ada semakin banyak beban kerja dalam ekosistem deret waktu yang perlu menelan dan menyimpan sejumlah besar data dan pada saat yang sama membutuhkan respons kueri latensi rendah saat mengakses data dengan serangkaian nilai dimensi kardinalitas tinggi.

Karena karakteristik seperti itu, rekomendasi di bagian ini akan berguna untuk beban kerja pelanggan yang memiliki yang berikut ini.

- Diadopsi atau ingin mengadopsi catatan multi-ukuran.
- Berharap untuk memiliki volume data yang tinggi masuk ke sistem yang akan disimpan untuk waktu yang lama.
- Memerlukan waktu respons latensi rendah untuk pola akses (kueri) utama mereka.
- Ketahuilah bahwa pola kueri yang paling penting melibatkan kondisi penyaringan semacam dalam predikat. Kondisi penyaringan ini didasarkan pada dimensi kardinalitas tinggi. Misalnya, pertimbangkan peristiwa atau agregasi oleh UserId,, ServerID DeviceId, host-name, dan sebagainya.

Dalam kasus ini, satu nama untuk semua tindakan multi-ukuran tidak akan membantu, karena mesin kami menggunakan nama multi-ukuran untuk mempartisi data dan memiliki satu nilai membatasi keuntungan partisi yang Anda dapatkan. Partisi untuk catatan ini terutama didasarkan pada dua dimensi. Katakanlah waktu berada pada sumbu x dan hash nama dimensi dan `measure_name` pada sumbu y. `measure_name`Dalam kasus ini bekerja hampir seperti kunci partisi.

Rekomendasi kami adalah sebagai berikut.

- Saat memodelkan data Anda untuk kasus penggunaan seperti yang kami sebutkan, gunakan turunan langsung dari pola akses kueri utama Anda. `measure_name` Sebagai contoh:
  - Kasus penggunaan Anda mengharuskan untuk melacak kinerja aplikasi dan QoE dari sudut pandang pengguna akhir. Ini juga bisa melacak pengukuran untuk satu server atau perangkat IoT.
  - Jika Anda menanyakan dan memfilter UserId, maka Anda perlu, pada waktu konsumsi, untuk menemukan cara terbaik untuk bergaul. `measure_name` UserId
  - Karena tabel multi-ukuran hanya dapat menampung 8192 nama ukuran yang berbeda, rumus apa pun yang diadopsi tidak boleh menghasilkan lebih dari 8192 nilai yang berbeda.
- Salah satu pendekatan yang telah kami terapkan dengan sukses untuk nilai string adalah dengan menerapkan algoritma hashing ke nilai string. Kemudian lakukan operasi modulo dengan nilai absolut dari hasil hash dan 8192.

```
measure_name = getMeasureName(UserId)
int getMeasureName(value) {
    hash_value =  abs(hash(value))
    return hash_value % 8192
}
```

- Kami juga menambahkan `abs()` untuk menghapus tanda menghilangkan kemungkinan nilai berkisar dari -8192 hingga 8192. Ini harus dilakukan sebelum operasi modulo.
- Dengan menggunakan metode ini, kueri Anda dapat berjalan pada sebagian kecil dari waktu yang diperlukan untuk berjalan pada model data yang tidak dipartisi.
- Saat melakukan kueri data, pastikan Anda menyertakan kondisi pemfilteran dalam predikat yang menggunakan nilai yang baru diturunkan dari measurement _name. Sebagai contoh:
  - ```
    SELECT * FROM your_database.your_table
    WHERE host_name = 'Host-1235' time BETWEEN '2022-09-01'
        AND '2022-09-18'
        AND measure_name = (SELECT
     cast(abs(from_big_endian_64(xxhash64(CAST('HOST-1235' AS varbinary))))%8192 AS
     varchar))
    ```
  - Ini akan meminimalkan jumlah total partisi yang dipindai untuk mendapatkan data yang akan diterjemahkan dalam kueri yang lebih cepat dari waktu ke waktu.

Perlu diingat bahwa jika Anda ingin mendapatkan manfaat dari skema partisi ini, hash perlu dihitung di sisi klien dan diteruskan ke Timestream LiveAnalytics sebagai nilai statis ke mesin kueri. Contoh sebelumnya menyediakan cara untuk memvalidasi bahwa hash yang dihasilkan dapat diselesaikan oleh mesin saat diperlukan.

Waktu	host_name	lokasi	server_type	cpu_usage	available_memory	cpu_temp
2022-09-07 21:48:44 .000 0	host-1235	kami-timur1	5.8xl	55	16.2	78
R2022-09-07 21:48:44 .000 0	host-3587	kami-barat1	5.8xl	62	18.1	81
2022-09-07 21:48:45.000 000000	host-258743	eu-pusat	5.8xl	88	9.4	91
    ```

Waktu	host_name	lokasi	server_type	cpu_usage	available_memory	cpu_temp
2022-09-07 21:48:45 .000 0	host-35654	kami-timur2	5.8xl	29	24	54
R2022-09-07 21:48:45 .000 0	host-254	kami-barat1	5.8xl	44	32	48

Untuk menghasilkan yang terkait `measure_name` mengikuti rekomendasi kami, ada dua jalur yang bergantung pada pola konsumsi Anda.

1. Untuk konsumsi batch data historis —Anda dapat menambahkan transformasi ke kode tulis Anda jika Anda akan menggunakan kode Anda sendiri untuk proses batch.

   Membangun di atas contoh sebelumnya.

```
        List<String> hosts = new ArrayList<>();

        hosts.add("host-1235");
        hosts.add("host-3587");
        hosts.add("host-258743");
        hosts.add("host-35654");
        hosts.add("host-254");

        for (String h: hosts){
            ByteBuffer buf2 = ByteBuffer.wrap(h.getBytes());
            partition = abs(hasher.hash(buf2, 0L)) % 8192;
            System.out.println(h + " - " + partition);

        }
```

   Output

```
host-1235 - 6445
host-3587 - 6399
```

```
host-258743 - 640
host-35654 - 2093
host-254 - 7051
```

Dataset yang dihasilkan

Waktu	host_name	lokasi	ukuran_nama	server_type	cpu_usage	available_memory	cpu_temp
2022-09-07 21:48:44 .00	host-1235	kami-timur1	6445	5.8xl	55	16.2	78
R2022-09-07 21:48:44 .00	host-3587	kami-barat1	6399	5.8xl	62	18.1	81
2022-09-07 21:48:45.000 000000	host-258743	eu-pusat	640	5.8xl	88	9.4	91
2022-09-07 21:48:45 .00	host-35654	kami-timur2	2093	5.8xl	29	24	54
R2022-09-07 21:48:45 .00	host-254	kami-barat1	7051	5.8xl	44	32	48

2. Untuk konsumsi waktu nyata —Anda perlu menghasilkan `measure_name` dalam penerbangan saat data masuk.

Dalam kedua kasus tersebut, kami sarankan Anda menguji algoritma penghasil hash Anda di kedua ujungnya (konsumsi dan kueri) untuk memastikan Anda mendapatkan hasil yang sama.

Berikut adalah beberapa contoh kode untuk menghasilkan nilai hash berdasarkan`host_name`.

Example Python

```
>>> import xxhash
>>> from bitstring import BitArray
>>> b=xxhash.xxh64('HOST-ID-1235').digest()
>>> BitArray(b).int % 8192
### 3195
```

Example Go

```
package main

import (
    "bytes"
    "fmt"
    "github.com/cespare/xxhash"
)

func main() {
    buf := bytes.NewBufferString("HOST-ID-1235")
    x := xxhash.New()
    x.Write(buf.Bytes())
    // convert unsigned integer to signed integer before taking mod
    fmt.Printf("%f\n", abs(int64(x.Sum64())) % 8192)
}

func abs(x int64) int64 {
    if (x < 0) {
        return -x
    }
    return x
}
```

Example Java

```
import java.nio.ByteBuffer;

import net.jpountz.xxhash.XXHash64;
```

```
public class test {
    public static void main(String[] args) {
        XXHash64 hasher = net.jpountz.xxhash.XXHashFactory.fastestInstance().hash64();

        String host = "HOST-ID-1235";
        ByteBuffer buf = ByteBuffer.wrap(host.getBytes());

        Long result = Math.abs(hasher.hash(buf, 0L));
        Long partition = result % 8192;

        System.out.println(result);
        System.out.println(partition);
    }
}
```

Example ketergantungan di Maven

```
        <dependency>
            <groupId>net.jpountz.lz4</groupId>
            <artifactId>lz4</artifactId>
            <version>1.3.0</version>
        </dependency>
```

Keamanan

- Untuk akses berkelanjutan ke Timestream untuk LiveAnalytics, pastikan bahwa kunci enkripsi diamankan dan tidak dicabut atau dibuat tidak dapat diakses.
- Pantau log API akses dari AWS CloudTrail. Audit dan cabut pola akses anomali dari pengguna yang tidak sah.
- Ikuti pedoman tambahan yang dijelaskan dalamPraktik terbaik keamanan untuk Amazon Timestream untuk LiveAnalytics.

Mengkonfigurasi Amazon Timestream untuk LiveAnalytics

Konfigurasikan periode retensi data untuk penyimpanan memori dan penyimpanan magnetik agar sesuai dengan pemrosesan data, penyimpanan, kinerja kueri, dan persyaratan biaya.

- Atur retensi data penyimpanan memori agar sesuai dengan persyaratan aplikasi Anda untuk memproses data yang datang terlambat. Data yang datang terlambat adalah data yang masuk

dengan stempel waktu lebih awal dari waktu saat ini. Ini dipancarkan dari sumber daya yang mengelompokkan peristiwa untuk jangka waktu tertentu sebelum mengirim data ke Timestream LiveAnalytics, dan juga dari sumber daya dengan konektivitas intermiten misalnya sensor IoT yang online sebentar-sebentar.

- Jika Anda mengharapkan data yang datang terlambat sesekali tiba dengan stempel waktu lebih awal dari penyimpanan penyimpanan memori, Anda harus mengaktifkan penulisan penyimpanan magnetik untuk tabel Anda. Setelah Anda mengatur tabel EnableMagneticStoreWrites , tabel akan menerima data dengan stempel waktu lebih awal dari retensi penyimpanan memori Anda tetapi dalam periode retensi penyimpanan magnetik Anda. MagneticStoreWritesProperties
- Pertimbangkan karakteristik kueri yang Anda rencanakan untuk dijalankan di Timestream LiveAnalytics seperti jenis kueri, frekuensi, rentang waktu, dan persyaratan kinerja. Ini karena penyimpanan memori dan penyimpanan magnetik dioptimalkan untuk skenario yang berbeda. Penyimpanan memori dioptimalkan untuk point-in-time kueri cepat yang memproses sejumlah kecil data terbaru yang dikirim ke Timestream untuk. LiveAnalytics Penyimpanan magnetik dioptimalkan untuk kueri analitik cepat yang memproses volume data sedang hingga besar yang dikirim ke Timestream untuk. LiveAnalytics
- Periode penyimpanan data Anda juga harus dipengaruhi oleh persyaratan biaya sistem Anda.

  Misalnya, pertimbangkan skenario di mana ambang data yang datang terlambat untuk aplikasi Anda adalah 2 jam dan aplikasi Anda mengirim banyak kueri yang memproses data bernilai satu hari, bernilai seminggu, atau bernilai sebulan. Dalam hal ini, Anda mungkin ingin mengonfigurasi periode retensi yang lebih kecil untuk penyimpanan memori (2-3 jam) dan memungkinkan lebih banyak data mengalir ke penyimpanan magnetik mengingat penyimpanan magnetik dioptimalkan untuk kueri analitik cepat.

Memahami dampak peningkatan atau penurunan periode retensi data dari penyimpanan memori dan penyimpanan magnetik dari tabel yang ada.

- Ketika Anda mengurangi periode retensi penyimpanan memori, data dipindahkan dari penyimpanan memori ke penyimpanan magnetik, dan transfer data ini bersifat permanen. Timestream for LiveAnalytics tidak mengambil data dari penyimpanan magnetik untuk mengisi penyimpanan memori. Ketika Anda mengurangi periode retensi penyimpanan magnetik, data dihapus dari sistem, dan penghapusan data bersifat permanen.
- Saat Anda meningkatkan periode retensi penyimpanan memori atau penyimpanan magnetik, perubahan berlaku untuk data yang dikirim ke Timestream LiveAnalytics sejak saat itu dan seterusnya. Timestream for LiveAnalytics tidak mengambil data dari penyimpanan magnetik untuk

mengisi penyimpanan memori. Misalnya, jika periode retensi penyimpanan memori awalnya diatur ke 2 jam dan kemudian ditingkatkan menjadi 24 jam, itu akan memakan waktu 22 jam untuk penyimpanan memori untuk berisi data senilai 24 jam.

Menulis

- Pastikan bahwa stempel waktu data yang masuk tidak lebih awal dari penyimpanan data yang dikonfigurasi untuk penyimpanan memori dan selambat-lambatnya periode konsumsi masa depan yang ditentukan dalam. Kuota Mengirim data dengan stempel waktu di luar batas ini akan mengakibatkan data ditolak oleh Timestream LiveAnalytics kecuali Anda mengaktifkan penulisan penyimpanan magnetik untuk tabel Anda. Jika Anda mengaktifkan penulisan penyimpanan magnetik, pastikan stempel waktu untuk data yang masuk tidak lebih awal dari retensi data yang dikonfigurasi untuk penyimpanan magnetik.
- Jika Anda mengharapkan data yang terlambat tiba, nyalakan penulisan penyimpanan magnetik untuk tabel Anda. Ini akan memungkinkan konsumsi data dengan stempel waktu yang berada di luar periode retensi penyimpanan memori Anda tetapi masih dalam periode retensi penyimpanan magnetik Anda. Anda dapat mengatur ini dengan memperbarui `EnableMagneticStoreWrites` bendera di `MagneticStoreWritesProperties` untuk tabel Anda. Properti ini palsu secara default. Perhatikan bahwa menulis ke toko magnetik tidak akan segera tersedia untuk kueri. Mereka akan tersedia dalam waktu 6 jam.
- Targetkan beban kerja throughput tinggi ke penyimpanan memori dengan memastikan stempel waktu data yang dicerna berada dalam batas retensi penyimpanan memori. Menulis ke penyimpanan magnetik terbatas pada jumlah maksimum partisi penyimpanan magnetik aktif yang dapat menerima konsumsi bersamaan untuk database. Anda dapat melihat `ActiveMagneticStorePartitions` metrik ini di CloudWatch. Untuk mengurangi partisi penyimpanan magnetik aktif, bertujuan untuk mengurangi jumlah seri dan durasi waktu yang Anda konsumsi secara bersamaan untuk konsumsi penyimpanan magnetik.
- Saat mengirim data ke Timestream untuk LiveAnalytics, kumpulkan beberapa catatan dalam satu permintaan untuk mengoptimalkan kinerja penyerapan data.
  - Sangat bermanfaat untuk mengumpulkan catatan dari deret waktu dan catatan yang sama dengan nama ukuran yang sama.
  - Batch sebanyak mungkin catatan dalam satu permintaan selama permintaan berada dalam batas layanan yang ditentukan dalamKuota.
  - Gunakan atribut umum jika memungkinkan untuk mengurangi transfer data dan biaya konsumsi. Untuk informasi lebih lanjut, lihat WriteRecords API.

- Jika Anda mengalami kegagalan sebagian sisi klien saat menulis data ke Timestream untuk LiveAnalytics, Anda dapat mengirim ulang kumpulan catatan yang gagal tertelan setelah Anda mengatasi penyebab penolakan.
- Data yang diurutkan berdasarkan stempel waktu memiliki kinerja penulisan yang lebih baik.
- Amazon Timestream for LiveAnalytics dirancang untuk secara otomatis menskalakan kebutuhan aplikasi Anda. Ketika Timestream untuk LiveAnalytics pemberitahuan melonjak dalam permintaan tulis dari aplikasi Anda, aplikasi Anda mungkin mengalami beberapa tingkat pembatasan penyimpanan memori awal. Jika aplikasi Anda mengalami pembatasan penyimpanan memori, lanjutkan mengirim data ke Timestream dengan kecepatan yang sama (atau meningkat) untuk LiveAnalytics mengaktifkan Timestream agar dapat secara otomatis LiveAnalytics menskalakan untuk memenuhi kebutuhan aplikasi Anda. Jika Anda melihat pelambatan penyimpanan magnetik, Anda harus mengurangi tingkat konsumsi penyimpanan magnetik hingga jumlah Anda turun. `ActiveMagneticStorePartitions`

Beban batch

Praktik terbaik untuk pemuatan batch dijelaskan dalamPraktik terbaik pemuatan batch.

Kueri

Berikut ini adalah praktik terbaik yang disarankan untuk kueri dengan Amazon LiveAnalytics Timestream untuk.

- Sertakan hanya nama ukuran dan dimensi yang penting untuk kueri. Menambahkan kolom asing akan meningkatkan pemindaian data, yang berdampak pada kinerja kueri.
- Sebelum menerapkan kueri Anda dalam produksi, kami sarankan Anda meninjau wawasan kueri untuk memastikan bahwa pemangkasan spasial dan temporal optimal. Untuk informasi selengkapnya, lihat Menggunakan wawasan kueri untuk mengoptimalkan kueri di Amazon Timestream.
- Jika memungkinkan, dorong perhitungan data ke Timestream untuk LiveAnalytics menggunakan agregat bawaan dan fungsi skalar dalam klausa dan SELECT WHERE klausa sebagaimana berlaku untuk meningkatkan kinerja kueri dan mengurangi biaya. Lihat SELECT dan Fungsi agregat.
- Jika memungkinkan, gunakan fungsi perkiraan. Misalnya, gunakan APPROX _ DISTINCT alih-alih COUNT (DISTINCTcolumn_name) untuk mengoptimalkan kinerja kueri dan mengurangi biaya kueri. Lihat Fungsi agregat.

- Gunakan CASE ekspresi untuk melakukan agregasi kompleks alih-alih memilih dari tabel yang sama beberapa kali. Lihat CASEPernyataan.
- Jika memungkinkan, sertakan rentang waktu dalam WHERE klausa kueri Anda. Ini mengoptimalkan kinerja dan biaya kueri. Misalnya, jika Anda hanya membutuhkan satu jam terakhir data dalam dataset Anda, maka sertakan predikat waktu seperti time > ago (1h). Lihat SELECT dan Interval dan durasi.
- Saat kueri mengakses subset ukuran dalam tabel, selalu sertakan nama ukuran dalam WHERE klausa kueri.
- Jika memungkinkan, gunakan operator kesetaraan saat membandingkan dimensi dan ukuran dalam WHERE klausa kueri. Predikat kesetaraan pada dimensi dan nama ukuran memungkinkan peningkatan kinerja kueri dan pengurangan biaya kueri.
- Jika memungkinkan, hindari penggunaan fungsi dalam WHERE klausa untuk mengoptimalkan biaya.
- Menahan diri dari menggunakan LIKE klausa beberapa kali. Sebaliknya, gunakan ekspresi reguler saat Anda memfilter beberapa nilai pada kolom string. Lihat Fungsi ekspresi reguler.
- Hanya gunakan kolom yang diperlukan dalam klausa GROUP BY dari kueri.
- Jika hasil kueri harus dalam urutan tertentu, secara eksplisit tentukan urutan itu dalam klausa ORDER BY dari kueri terluar. Jika hasil kueri Anda tidak memerlukan pengurutan, hindari menggunakan klausa ORDER BY untuk meningkatkan kinerja kueri.
- Gunakan LIMIT klausa jika Anda hanya membutuhkan baris N pertama dalam kueri Anda.
- Jika Anda menggunakan klausa ORDER BY untuk melihat nilai N atas atau bawah, gunakan LIMIT klausa untuk mengurangi biaya kueri.
- Gunakan token pagination dari respons yang dikembalikan untuk mengambil hasil kueri. Untuk informasi selengkapnya, lihat Kueri.
- Jika Anda sudah mulai menjalankan kueri dan menyadari bahwa kueri tidak akan mengembalikan hasil yang Anda cari, batalkan kueri untuk menghemat biaya. Untuk informasi lebih lanjut, lihat CancelQuery.
- Jika aplikasi Anda mengalami pembatasan, lanjutkan pengiriman data ke Amazon Timestream dengan kecepatan yang sama LiveAnalytics untuk mengaktifkan Amazon Timestream agar dapat menskalakan otomatis LiveAnalytics untuk memenuhi kebutuhan throughput kueri aplikasi Anda.
- Jika persyaratan konkurensi kueri aplikasi Anda melebihi batas default Timestream untuk LiveAnalytics, hubungi AWS Support untuk peningkatan batas.

Pertanyaan terjadwal

Kueri terjadwal membantu Anda mengoptimalkan dasbor dengan melakukan pra-komputasi beberapa statistik agregat di seluruh armada. Jadi pertanyaan alami untuk ditanyakan adalah bagaimana Anda mengambil kasus penggunaan Anda dan mengidentifikasi hasil mana yang harus dihitung sebelumnya dan bagaimana menggunakan hasil ini yang disimpan dalam tabel turunan untuk membuat dasbor Anda. Langkah pertama dalam proses ini adalah mengidentifikasi panel mana yang harus dihitung sebelumnya. Di bawah ini adalah beberapa pedoman tingkat tinggi:

- Pertimbangkan byte yang dipindai oleh kueri yang digunakan untuk mengisi panel, frekuensi muat ulang dasbor, dan jumlah pengguna bersamaan yang akan memuat dasbor ini. Anda harus mulai dengan dasbor yang paling sering dimuat dan memindai sejumlah besar data. Dua dasbor pertama dalam contoh dasbor agregat serta dasbor agregat dalam contoh penelusuran adalah contoh yang baik dari dasbor tersebut.
- Pertimbangkan perhitungan mana yang berulang kali digunakan. Meskipun dimungkinkan untuk membuat kueri terjadwal untuk setiap panel dan setiap nilai variabel yang digunakan dalam panel, Anda dapat secara signifikan mengoptimalkan biaya dan jumlah kueri terjadwal dengan mencari jalan untuk menggunakan satu perhitungan untuk pra-menghitung data yang diperlukan untuk beberapa panel.
- Pertimbangkan frekuensi kueri terjadwal Anda untuk menyegarkan hasil yang terwujud dalam tabel turunan. Anda ingin menganalisis seberapa sering dasbor disegarkan vs. jendela waktu yang ditanyakan di dasbor vs. binning waktu yang digunakan dalam pra-komputasi serta panel di dasbor. Misalnya, jika dasbor yang merencanakan agregat per jam selama beberapa hari terakhir hanya disegarkan sekali dalam beberapa jam, Anda mungkin ingin mengonfigurasi kueri terjadwal untuk hanya menyegarkan sekali setiap 30 menit atau satu jam. Di sisi lain, jika Anda memiliki dasbor yang memplot agregat per menit dan disegarkan setiap menit atau lebih, Anda ingin kueri terjadwal Anda menyegarkan hasil setiap menit atau beberapa menit.
- Pertimbangkan pola kueri mana yang dapat dioptimalkan lebih lanjut (baik dari perspektif biaya kueri dan latensi kueri) menggunakan kueri terjadwal. Misalnya, saat menghitung nilai dimensi unik yang sering digunakan sebagai variabel di dasbor, atau mengembalikan titik data terakhir yang dipancarkan dari sensor atau titik data pertama yang dipancarkan dari sensor setelah tanggal tertentu, dll. Beberapa contoh pola ini dibahas dalam panduan ini.

Pertimbangan sebelumnya akan berdampak signifikan pada penghematan Anda saat Anda memindahkan dasbor untuk menanyakan tabel turunan, kesegaran data di dasbor Anda, dan biaya yang dikeluarkan oleh kueri terjadwal.

Aplikasi klien dan integrasi yang didukung

Jalankan aplikasi klien Anda dari Wilayah yang sama dengan Timestream LiveAnalytics untuk mengurangi latensi jaringan dan biaya transfer data. Untuk informasi selengkapnya tentang bekerja dengan layanan lain, lihatBekerja dengan layanan yang lain. Berikut ini adalah beberapa tautan bermanfaat lainnya.

- Praktik Terbaik untuk AWS Pengembangan dengan AWS SDK for Java
- Praktik terbaik untuk bekerja dengan AWS Lambda fungsi
- Praktik Terbaik untuk Amazon Managed Service untuk Apache Flink
- Praktik terbaik untuk membuat dasbor di Grafana

Umum

- Pastikan Anda mengikuti The AWS Well-Architected Framework saat menggunakan Timestream untuk. LiveAnalytics Whitepaper ini memberikan panduan seputar praktik terbaik dalam keunggulan operasional, keamanan, keandalan, efisiensi kinerja, dan pengoptimalan biaya.

Pengukuran dan optimalisasi biaya

Dengan Amazon Timestream untuk LiveAnalytics, Anda hanya membayar untuk apa yang Anda gunakan. Timestream untuk LiveAnalytics meter secara terpisah untuk penulisan, data yang disimpan, dan data yang dipindai oleh kueri. Harga setiap dimensi pengukuran ditentukan pada halaman harga. Anda dapat memperkirakan tagihan bulanan Anda menggunakan Amazon Timestream untuk Kalkulator LiveAnalytics Harga.

Bagian ini menjelaskan cara kerja pengukuran untuk penulisan, penyimpanan, dan kueri di Timestream untuk. LiveAnalytics Contoh skenario dan perhitungan juga disediakan. Selain itu, daftar praktik terbaik untuk pengoptimalan biaya disertakan. Anda dapat memilih topik di bawah ini:

Topik

- Menulis
- Penyimpanan
- Kueri
- Optimalisasi Biaya

- Pemantauan CloudWatch dengan Amazon

Menulis

Ukuran tulis dari setiap peristiwa deret waktu dihitung sebagai jumlah ukuran stempel waktu dan satu atau lebih nama dimensi, nilai dimensi, nama ukuran, dan nilai ukuran. Ukuran stempel waktu adalah 8 byte. Ukuran nama dimensi, nilai dimensi, dan nama ukuran adalah panjang dari UTF -8 byte yang dikodekan dari string yang mewakili setiap nama dimensi, nilai dimensi, dan nama ukuran. Ukuran nilai ukuran tergantung pada tipe data. Ini adalah 1 byte untuk tipe data boolean, 8 byte untuk bigint dan ganda, dan panjang UTF -8 byte yang dikodekan untuk string. Setiap tulisan dihitung dalam satuan 1 KiB.

Dua contoh perhitungan disediakan di bawah ini:

Topik

- Menghitung ukuran tulis peristiwa deret waktu
- Menghitung jumlah penulisan

Menghitung ukuran tulis peristiwa deret waktu

Pertimbangkan peristiwa deret waktu yang mewakili CPU pemanfaatan EC2 instance seperti yang ditunjukkan di bawah ini:

Waktu	region	az	vpc	Nama host	ukuran_nama	ukuran_nilai: :ganda
1602983435238563000	us-east-1	1d	vpc-1a2b3c4d	Host-24gju	pemanfaatan cpu_	35,0

Ukuran penulisan peristiwa deret waktu dapat dihitung sebagai:

- waktu = 8 byte
- dimensi pertama = 15 byte (`region+us-east-1`)
- dimensi kedua = 4 byte (`az+1d`)

- dimensi ketiga = 15 byte (`vpc+vpc-1a2b3c4d`)
- dimensi keempat = 18 byte (`hostname+host-24Gju`)
- nama ukuran = 15 byte () `cpu_utilization`
- nilai ukuran = 8 byte

Tulis ukuran peristiwa deret waktu = 83 byte

Menghitung jumlah penulisan

Sekarang pertimbangkan 100 EC2 instance, mirip dengan instance yang dijelaskan dalamMenghitung ukuran tulis peristiwa deret waktu, memancarkan metrik setiap 5 detik. Total penulisan bulanan untuk EC2 instance akan bervariasi berdasarkan berapa banyak peristiwa deret waktu yang ada per penulisan dan jika atribut umum digunakan saat mengelompokkan peristiwa deret waktu. Contoh penghitungan total penulisan bulanan disediakan untuk masing-masing skenario berikut:

Topik

- Acara satu deret waktu per tulis
- Mengelompokkan peristiwa deret waktu dalam sebuah tulisan
- Mengelompokkan peristiwa deret waktu dan menggunakan atribut umum dalam penulisan

Acara satu deret waktu per tulis

Jika setiap penulisan hanya berisi satu peristiwa deret waktu, total penulisan bulanan dihitung sebagai:

- 100 peristiwa deret waktu = 100 menulis setiap 5 detik
- x 12 tulisan/menit = 1.200 menulis
- x 60 menit/jam = 72.000 menulis
- x 24 jam/hari = 1,728.000 menulis
- x 30 hari/bulan = 51.840,000 menulis

Total penulisan bulanan = 51,840,000

Mengelompokkan peristiwa deret waktu dalam sebuah tulisan

Mengingat setiap penulisan diukur dalam satuan 1 KB, penulisan dapat berisi kumpulan 12 peristiwa deret waktu (998 byte) dan total penulisan bulanan dihitung sebagai:

- 100 peristiwa deret waktu = 9 tulis (12 peristiwa deret waktu per penulisan) setiap 5 detik
- x 12 tulisan/menit = 108 menulis
- x 60 menit/jam = 6.480 menulis
- x 24 jam/hari = 155,520 menulis
- x 30 hari/bulan = 4.665.600 menulis

Total penulisan bulanan = 4,665,600

Mengelompokkan peristiwa deret waktu dan menggunakan atribut umum dalam penulisan

Jika nama wilayah, az, vpc, dan ukuran umum di 100 EC2 instance, nilai umum dapat ditentukan hanya sekali per penulisan dan disebut sebagai atribut umum. Dalam hal ini, ukuran atribut umum adalah 52 byte, dan ukuran peristiwa deret waktu adalah 27 byte. Mengingat setiap penulisan diukur dalam satuan 1 KiB, tulisan dapat berisi 36 peristiwa deret waktu dan atribut umum, dan total penulisan bulanan dihitung sebagai:

- 100 peristiwa deret waktu = 3 tulis (36 peristiwa deret waktu per penulisan) setiap 5 detik
- x 12 tulisan/menit = 36 menulis
- x 60 menit/jam = 2.160 menulis
- x 24 jam/hari = 51,840 menulis
- x 30 hari/bulan = 1.555.200 menulis

Total penulisan bulanan = 1,555,200

> **Note**
>
> Karena penggunaan batching, atribut umum dan pembulatan penulisan ke unit 1KB, ukuran penyimpanan peristiwa deret waktu mungkin berbeda dari ukuran tulis.

Penyimpanan

Ukuran penyimpanan setiap peristiwa deret waktu di penyimpanan memori dan penyimpanan magnetik dihitung sebagai jumlah ukuran stempel waktu, nama dimensi, nilai dimensi, nama ukuran, dan nilai pengukuran. Ukuran stempel waktu adalah 8 byte. Ukuran nama dimensi, nilai dimensi, dan nama ukuran adalah panjang dari UTF -8 byte yang dikodekan dari setiap string yang mewakili nama dimensi, nilai dimensi, dan nama ukuran. Ukuran nilai ukuran tergantung pada tipe data. Ini adalah 1 byte untuk tipe data boolean, 8 byte untuk bigint dan ganda, dan panjang UTF -8 byte yang dikodekan untuk string. Setiap ukuran disimpan sebagai catatan terpisah di Amazon Timestream untuk LiveAnalytics, yaitu jika peristiwa deret waktu Anda memiliki empat ukuran, akan ada empat catatan untuk peristiwa deret waktu tersebut dalam penyimpanan.

Mempertimbangkan contoh peristiwa deret waktu yang mewakili CPU pemanfaatan sebuah EC2 instance (lihatMenghitung ukuran tulis peristiwa deret waktu), ukuran penyimpanan peristiwa deret waktu dihitung sebagai:

- waktu = 8 byte
- dimensi pertama = 15 byte (`region+us-east-1`)
- dimensi kedua = 4 byte (`az+1d`)
- dimensi ketiga = 15 byte (`vpc+vpc-1a2b3c4d`)
- dimensi keempat = 18 byte (`hostname+host-24Gju`)
- nama ukuran = 15 byte () `cpu_utilization`
- nilai ukuran = 8 byte

Ukuran penyimpanan acara deret waktu = 83 byte

> **Note**
>
> Penyimpanan memori diukur dalam GB-jam dan penyimpanan magnetik diukur dalam GB-bulan.

Kueri

Kueri dikenakan biaya berdasarkan durasi unit komputasi Timestream (TCUs) yang digunakan oleh aplikasi Anda dalam TCU -jam seperti yang ditentukan pada halaman harga Amazon Timestream.

Amazon Timestream untuk mesin LiveAnalytics kueri memangkas data yang tidak relevan saat memproses kueri. Kueri dengan proyeksi dan predikat termasuk rentang waktu, nama ukuran, dan/atau nama dimensi memungkinkan mesin pengolah kueri memangkas sejumlah besar data dan membantu menurunkan biaya kueri.

Optimalisasi Biaya

Untuk mengoptimalkan biaya penulisan, penyimpanan, dan kueri, gunakan praktik terbaik berikut dengan Amazon LiveAnalytics Timestream untuk:

- Batch beberapa peristiwa deret waktu per penulisan untuk mengurangi jumlah permintaan tulis.
- Pertimbangkan untuk menggunakan catatan Multi-ukuran, yang memungkinkan Anda menulis beberapa ukuran deret waktu dalam satu permintaan tulis dan menyimpan data Anda dengan cara yang lebih ringkas. Ini mengurangi jumlah permintaan tulis serta biaya penyimpanan data dan biaya kueri.
- Gunakan atribut umum dengan batching untuk mengumpulkan lebih banyak peristiwa deret waktu per penulisan untuk lebih mengurangi jumlah permintaan tulis.
- Atur retensi data penyimpanan memori agar sesuai dengan persyaratan aplikasi Anda untuk memproses data yang datang terlambat. Data yang datang terlambat adalah data yang masuk dengan stempel waktu lebih awal dari waktu saat ini dan di luar periode penyimpanan penyimpanan memori.
- Atur retensi data penyimpanan magnetik agar sesuai dengan persyaratan penyimpanan data jangka panjang Anda.
- Saat menulis kueri, sertakan hanya nama ukuran dan dimensi yang penting untuk kueri. Menambahkan kolom asing akan meningkatkan pemindaian data dan karenanya juga akan meningkatkan biaya kueri. Kami menyarankan Anda meninjau wawasan kueri untuk menilai efisiensi pemangkasan dari dimensi dan ukuran yang disertakan.
- Jika memungkinkan, sertakan rentang waktu dalam WHERE klausa kueri Anda. Misalnya, jika Anda hanya membutuhkan satu jam terakhir data dalam kumpulan data Anda, sertakan predikat waktu seperti. `time > ago(1h)`
- Saat kueri mengakses subset ukuran dalam tabel, selalu sertakan nama ukuran dalam WHERE klausa kueri.
- Jika Anda sudah mulai menjalankan kueri dan menyadari bahwa kueri tidak akan mengembalikan hasil yang Anda cari, batalkan kueri untuk menghemat biaya.

Pemantauan CloudWatch dengan Amazon

Anda dapat memantau Timestream untuk LiveAnalytics menggunakan Amazon CloudWatch, yang mengumpulkan dan memproses data mentah dari Timestream LiveAnalytics menjadi metrik yang dapat dibaca. near-real-time Ini mencatat statistik ini selama dua minggu sehingga Anda dapat mengakses informasi historis dan mendapatkan perspektif yang lebih baik tentang kinerja aplikasi atau layanan web Anda. Secara default, Timestream untuk data LiveAnalytics metrik secara otomatis dikirim ke CloudWatch dalam periode 1 menit atau 15 menit. Untuk informasi selengkapnya, lihat Apa itu Amazon CloudWatch? di Panduan CloudWatch Pengguna Amazon.

Topik

Bagaimana cara menggunakan Timestream untuk LiveAnalytics metrik?

Metrik yang dilaporkan oleh Timestream untuk LiveAnalytics memberikan informasi yang dapat Anda analisis dengan berbagai cara. Daftar berikut menunjukkan beberapa penggunaan umum untuk metrik. Daftar ini berisi saran agar Anda dapat memulai, bukan daftar komprehensif.

Bagaimana saya dapat melakukannya?	Metrik terkait
`How can I determine if any system errors occurred?`	Anda dapat memantau `SystemErrors` untuk menentukan apakah ada permintaan yang menghasilkan kode kesalahan server. Biasanya, metrik ini sama dengan nol. Jika tidak, Anda mungkin ingin menyelidikinya.
`How can I monitor the amount of data in the memory store?`	Anda dapat memantau `MemoryCumulativeBytesMetered` selama periode waktu yang ditentukan, untuk memantau jumlah data yang disimpan di penyimpanan memori dalam byte. Metrik ini dipancarkan setiap jam dan Anda dapat melacak byte yang disimpan di akun serta pada perincian database. Penyimpanan memori diukur dalam GB-jam (biaya penyimpanan 1GB data selama satu jam). Jadi mengalikan nilai per jam `MemoryCumulativeBytesMetered` dengan harga GB-

Bagaimana saya dapat melakukannya?	Metrik terkait
	jam di Wilayah Anda akan memberi Anda biaya yang dikeluarkan per jam. Dimensi: Operasi (penyimpanan), DatabaseName, Nama metrik
How can I monitor the amount of data in the magnetic store?	Anda dapat memantau `MagneticCumulativeBytesMetered` selama periode waktu yang ditentukan, untuk memantau jumlah data yang disimpan dalam penyimpanan magnetik dalam byte. Metrik ini dipancarkan setiap jam dan Anda dapat melacak byte yang disimpan di akun serta pada perincian database. Penyimpanan memori diukur dalam GB-bulan (biaya penyimpanan 1GB data selama satu bulan). Jadi mengalikan nilai per jam `MagneticCumulativeBytesMetered` dengan harga GB-bulan di Wilayah Anda akan memberi Anda biaya yang dikeluarkan per jam. Misalnya, jika nilainya `MagneticCumulativeBytesMetered` adalah 107374182400 byte (100GB), maka biaya per jam 1GB data di penyimpanan magnetik = (0,03) (harga us-east-1)/(30,4* 24). Mengalikan nilai ini dengan `MagneticCumulativeBytesMetered` dalam GB akan memberikan ~ $0,004 untuk jam itu. Dimensi: Operasi (Penyimpanan), DatabaseName, Nama metrik
How can I monitor the data scanned by queries?	Anda dapat memantau `CumulativeBytesMetered` selama periode waktu yang ditentukan, untuk memantau data yang dipindai oleh kueri (dalam byte) yang dikirim ke Timestream untuk. LiveAnalytics Metrik ini dipancarkan setelah eksekusi kueri dan Anda dapat melacak data yang dipindai pada perincian akun dan database. Anda dapat menghitung biaya kueri untuk periode tertentu dengan mengalikan nilai metrik dengan harga pindaian per GB di Wilayah Anda. Byte yang dipindai oleh kueri terjadwal dicatat dalam metrik ini. Dimensi: Operasi (Kueri), DatabaseName, Nama metrik

Bagaimana saya dapat melakukannya?	Metrik terkait
How can I monitor the data scanned by scheduled queries?	Anda dapat memantau `CumulativeBytesMetered` selama periode waktu yang ditentukan, untuk memantau data yang dipindai oleh kueri terjadwal (dalam byte) yang dijalankan oleh Timestream untuk. LiveAnalytics Metrik ini dipancarkan setelah eksekusi kueri dan Anda dapat melacak data yang dipindai pada perincian akun dan database. Anda dapat menghitung biaya kueri untuk periode tertentu dengan mengalikan nilai metrik dengan harga pindaian per GB di Wilayah Anda. **Note** Byte yang diukur juga diperhitungkan dalam kueri. `CumulativeBytesMetered` Dimensi: Operasi (TriggeredScheduledQuery), DatabaseName, Nama metrik

Bagaimana saya dapat melakukannya?	Metrik terkait
How can I monitor the number of records ingested?	Anda dapat memantau NumberOfRecords selama periode waktu yang ditentukan untuk memantau jumlah catatan yang dicerna. Anda dapat melacak byte yang disimpan di akun serta pada granularitas database. Anda juga dapat menggunakan metrik ini untuk memantau penulisan yang dibuat oleh Kueri Terjadwal saat hasil kueri ditulis ke dalam tabel terpisah. Saat menggunakan WriteRecords API, metrik dipancarkan untuk setiap WriteRecords permintaan, dengan dimensi CloudWatch Operasi. WriteRecords Saat menggunakan BatchLoad or ScheduledQuery APIs, metrik dipancark an pada interval yang ditentukan oleh layanan sampai tugas selesai. Dimensi CloudWatch Operasi untuk metrik ini adalah salah satu BatchLoad atauScheduledQuery , tergantung pada mana yang API digunakan. Dimensi: Operasi (WriteRecords, BatchLoad, atau Scheduled Query), DatabaseName, Nama metrik

Bagaimana saya dapat melakukannya?	Metrik terkait
How can I monitor the cost of records ingested?	Anda dapat memantau CumulativeBytesMetered untuk memantau jumlah byte yang tertelan yang menghasilkan biaya. Anda dapat melacak byte yang disimpan di akun serta pada granularitas database. Catatan yang dicerna diukur dalam byte kumulatif. Mengalikan nilai dengan harga CumulativeBytesMetered Menulis di Wilayah Anda memberi Anda biaya konsumsi yang dikeluarkan. Saat menggunakan WriteRecords API, metrik ini dipancarkan untuk setiap WriteRecords permintaan, dengan dimensi CloudWatch Operasi. WriteRecords Saat menggunakan BatchLoad or ScheduledQuery API, metrik dipancarkan pada interval yang ditentukan oleh layanan sampai tugas selesai. Dimensi CloudWatch Operasi untuk metrik ini adalah BatchLoad atau ScheduledQuery tergantung pada mana yang API digunakan.. Dimensi: Operasi (WriteRecords, BatchLoad, atau Scheduled Query), DatabaseName, Nama metrik
How can I monitor the Timestream Compute Units (TCUs) used in my account?	Anda dapat memantau QueryTCU selama periode waktu yang ditentukan, untuk memantau unit komputasi yang digunakan untuk beban kerja kueri di akun. Metrik ini dipancarkan dengan satuan komputasi maksimum dan minimum untuk setiap menit selama beban kerja kueri aktif dari akun. Unit: Count Statistik yang Valid: Minimum, Maksimum Metrik: ResourceCount Dimensi:Service: Timestream ,Resource: QueryTCU,Type: Resource, Class: OnDemand

Timestream untuk LiveAnalytics metrik dan dimensi

Saat Anda berinteraksi dengan Timestream LiveAnalytics, Timestream akan mengirimkan metrik dan dimensi berikut ke Amazon. CloudWatch Semua metrik dikumpulkan dan dilaporkan setiap menit. Anda dapat menggunakan prosedur berikut untuk melihat metrik Timestream untuk. LiveAnalytics

Untuk melihat metrik menggunakan konsol CloudWatch

Metrik dikelompokkan terlebih dahulu berdasarkan namespace layanan, lalu berdasarkan berbagai kombinasi dimensi dalam setiap namespace.

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Jika perlu, ubah Wilayah. Pada bilah navigasi, pilih Wilayah tempat AWS sumber daya Anda berada. Untuk informasi selengkapnya, lihat Titik Akhir AWS Layanan.
3. Di panel navigasi, pilih Metrik.
4. Di bawah tab Semua metrik, pilih `AWS/Timestream for LiveAnalytics.`

Untuk melihat metrik menggunakan AWS CLI

- Pada jendela perintah, gunakan perintah berikut.

```
aws cloudwatch list-metrics --namespace "AWS/Timestream"
```

Dimensi untuk Timestream untuk metrik LiveAnalytics

Metrik untuk Timestream untuk LiveAnalytics dikualifikasikan berdasarkan nilai untuk akun, nama tabel, atau operasi. Anda dapat menggunakan CloudWatch konsol untuk mengambil Timestream untuk LiveAnalytics data sepanjang salah satu dimensi dalam tabel berikut:

Dimensi	Deskripsi
`DatabaseName`	Dimensi ini membatasi data ke Timestream tertentu untuk LiveAnalytics database. Nilai ini dapat berupa database apa pun di Wilayah saat ini dan AWS akun saat ini
`Operation`	Dimensi ini membatasi data ke salah satu Timestream untuk LiveAnalytics operasi, sepertiStorage,, WriteReco

Dimensi	Deskripsi
	`rds BatchLoad` , atau`ScheduledQuery` . Lihat Timestrea m for LiveAnalytics Query API Reference untuk daftar nilai yang tersedia.
`TableName`	Dimensi ini membatasi data ke tabel tertentu dalam Timestream untuk LiveAnalyticss database.

Important

`CumulativeBytesMetered`, `UserErrors` dan `SystemErrors` metrik hanya memiliki `Operation` dimensi. `SuccessfulRequestLatency`metrik selalu memiliki `Operation` dimensi, tetapi mungkin juga memiliki `DatabaseName` dan `TableName` dimensi juga, tergantung pada nilai. `Operation` Ini karena Timestream untuk operasi LiveAnalytics tingkat tabel memiliki `DatabaseName` dan `TableName` sebagai dimensi, tetapi operasi tingkat akun tidak.

Timestream untuk metrik LiveAnalytics

Note

Amazon CloudWatch menggabungkan semua Timestream berikut untuk LiveAnalytics metrik pada interval satu menit.

Metrik umum

Metrik	Deskripsi
`SuccessfulRequestLatency`	Permintaan yang berhasil ke Timestream LiveAnalytics selama periode waktu yang ditentukan. SuccessfulRequestLatency dapat memberikan dua jenis informasi yang berbeda:

Metrik	Deskripsi
	• Waktu yang telah berlalu untuk permintaan yang berhasil (Minimum, Maksimum, Jumlah, atau Rata-rata). • Jumlah permintaan yang berhasil (SampleCount). SuccessfulRequestLatency mencerminkan aktivitas hanya dalam Timestream untuk LiveAnalytics dan tidak memperhitungkan latensi jaringan atau aktivitas sisi klien. Unit: `Milliseconds` Dimensi • `DatabaseName` • `TableName` • `Operation` Statistik Valid: • `Minimum` • `Maximum` • `Average` • `SampleCount` • `P10` • `p50` • `p90` • `p95` • `p99`

Metrik penulisan dan penyimpanan

Metrik	Deskripsi
`MagneticStoreRejectedRecordCount`	Jumlah catatan tertulis penyimpanan magnetik yang ditolak secara asinkron. Ini dapat terjadi jika catatan baru memiliki versi yang kurang dari versi saat ini atau catatan baru memiliki versi yang sama dengan versi saat ini tetapi memiliki data yang berbeda. Unit: `Count` Dimensi • `DatabaseName` • `TableName` • `Operation` Statistik Valid: • `Sum` • `SampleCount`
`MagneticStoreRejectedUpload UserFailures`	Jumlah penyimpanan magnetik menolak laporan catatan yang tidak diunggah karena kesalahan pengguna. Ini bisa disebabkan oleh IAM izin yang tidak dikonfigurasi dengan benar atau bucket S3 yang dihapus. Unit: `Count` Dimensi • `DatabaseName` • `TableName` • `Operation`

Metrik	Deskripsi
	Statistik Valid: • Sum • SampleCount
MagneticStoreRejectedUpload SystemFailures	Jumlah penyimpanan magnetik menolak laporan catatan yang tidak diunggah karena kesalahan sistem. Unit: Count Dimensi • DatabaseName • TableName • Operation Statistik Valid: • Sum • SampleCount
ActiveMagneticStorePartitions	Jumlah partisi penyimpanan magnetik yang secara aktif menelan data pada waktu tertentu. Unit: Count Dimensi • DatabaseName • Operation Statistik Valid: • Sum • SampleCount

Metrik	Deskripsi
`MagneticStorePendingRecordsLatency`	Tulisan tertua ke toko magnetik yang tidak tersedia untuk kueri. Catatan yang ditulis ke toko magnetik akan tersedia untuk pertanyaan dalam waktu 6 jam. Unit: `Milliseconds` Dimensi • `DatabaseName` • `TableName` • `Operation` Statistik Valid: • `Minimum` • `Maximum` • `Average` • `SampleCount` • `P10` • `p50` • `p90` • `p95` • `p99`
`MemoryCumulativeBytesMetered`	Jumlah data yang disimpan di penyimpanan memori, dalam byte Unit: `Bytes` Dimensi: `Operation` Statistik Valid: • `Average`

Metrik	Deskripsi
`MagneticCumulativeBytesMetered`	Jumlah data yang disimpan dalam penyimpan an magnetik, dalam byte Unit: `Bytes` Dimensi: `Operation` Statistik Valid: • `Average`
`CumulativeBytesMetered`	Jumlah data yang diukur dengan konsumsi ke Timestream untuk LiveAnalytics, dalam byte. Unit: `Bytes` Dimensi: `Operation` Statistik yang Valid: `Sum`
`NumberOfRecords`	Jumlah catatan yang dicerna ke Timestream untuk. LiveAnalytics Unit: `Count` Dimensi: `Operation` Statistik yang Valid: `Sum`

Metrik kueri

Metrik	Deskripsi
`CumulativeBytesMetered`	Jumlah data yang dipindai oleh kueri yang dikirim ke Timestream untuk LiveAnalytics, dalam byte. Unit: `Bytes`

Metrik	Deskripsi
	Dimensi: `Operation` Statistik Valid: • `Sum`
`ResourceCount`	Unit Komputasi Timestream (TCUs) digunakan untuk beban kerja kueri di akun. Metrik ini dipancarkan dengan satuan komputasi maksimum dan minimum untuk setiap menit selama beban kerja kueri aktif dari akun. Unit: `Count` Statistik yang Valid: Minimum, Maksimum Dimensi:`Service: Timestrea m` ,`Resource: QueryTCU`,`Type: Resource`, `Class: OnDemand`

Metrik kesalahan

Metrik	Deskripsi
`SystemErrors`	Permintaan ke Timestream untuk LiveAnaly tics yang menghasilkan SystemError selama periode waktu yang ditentukan. A SystemErr or biasanya menunjukkan kesalahan layanan internal. Unit: `Count` Dimensi: `Operation` Statistik Valid: • `Sum`

Metrik	Deskripsi
	• `SampleCount`
`UserErrors`	Permintaan ke Timestream untuk LiveAnaly tics itu menghasilkan InvalidRequest kesalahan selama periode waktu yang ditentukan. InvalidRequest Biasanya menunjukkan kesalahan sisi klien, seperti kombinasi parameter yang tidak valid, upaya untuk memperbarui tabel yang tidak ada, atau tanda tangan permintaan yang salah. UserErrors mewakili agregat permintaan yang tidak valid untuk AWS Wilayah saat ini dan akun saat ini. AWS Unit: `Count` Dimensi: `Operation` Statistik Valid: • `Sum` • `SampleCount`

> **Important**
>
> Tidak semua statistik, seperti `Average` atau `Sum`, berlaku untuk setiap metrik. Namun, semua nilai ini tersedia melalui Timestream untuk LiveAnalytics konsol, atau dengan menggunakan CloudWatch konsol, AWS CLI, atau AWS SDKs untuk semua metrik.

Membuat CloudWatch alarm untuk memantau Timestream LiveAnalytics

Anda dapat membuat CloudWatch alarm Amazon untuk Timestream karena LiveAnalytics mengirimkan pesan Amazon Simple Notification Service (AmazonSNS) saat alarm berubah status. Alarm mengawasi metrik tunggal selama periode waktu yang Anda tentukan. Alarm tersebut melakukan satu atau beberapa tindakan berdasarkan nilai metrik yang relatif terhadap ambang batas

tertentu selama beberapa periode waktu. Tindakan ini adalah pemberitahuan yang dikirim ke SNS topik Amazon atau kebijakan Auto Scaling.

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

Untuk informasi selengkapnya tentang membuat CloudWatch alarm, lihat Menggunakan CloudWatch Alarm Amazon di CloudWatch Panduan Pengguna Amazon.

Pemecahan Masalah

Bagian ini berisi informasi tentang pemecahan masalah Timestream untuk. LiveAnalytics

Topik

- Penanganan WriteRecords throttle
- Menangani catatan yang ditolak
- Pemecahan masalah UNLOAD dari Timestream untuk LiveAnalytics
- Timestream untuk LiveAnalytics kode kesalahan tertentu

Penanganan WriteRecords throttle

Permintaan penulisan penyimpanan memori Anda ke Timestream dapat dibatasi sebagai skala Timestream untuk beradaptasi dengan kebutuhan konsumsi data aplikasi Anda. Jika aplikasi Anda mengalami pengecualian pembatasan, Anda harus terus mengirim data pada throughput yang sama (atau lebih tinggi) untuk memungkinkan Timestream secara otomatis menskalakan kebutuhan aplikasi Anda.

Permintaan tulis penyimpanan magnetik Anda ke Timestream dapat dibatasi jika batas maksimum partisi penyimpanan magnetik menerima konsumsi. Anda akan melihat pesan throttle yang mengarahkan Anda untuk memeriksa metrik `ActiveMagneticStorePartitions` Cloudwatch untuk database ini. Throttle ini bisa memakan waktu hingga 6 jam untuk menyelesaikannya. Untuk menghindari throttle ini, Anda harus menggunakan penyimpanan memori untuk beban kerja konsumsi throughput yang tinggi. Untuk konsumsi penyimpanan magnetik, Anda dapat menargetkan menelan ke dalam partisi yang lebih sedikit dengan membatasi berapa banyak seri dan durasi waktu yang Anda konsumsi

Untuk informasi selengkapnya tentang praktik terbaik konsumsi data, lihat. Menulis

Menangani catatan yang ditolak

Jika Timestream menolak catatan, Anda akan menerima `RejectedRecordsException` rincian tentang penolakan. Silakan merujuk ke Menangani kegagalan menulis untuk informasi lebih lanjut tentang cara mengekstrak informasi ini dari WriteRecords tanggapan.

Semua penolakan akan dimasukkan dalam tanggapan ini dengan pengecualian pembaruan ke penyimpanan magnetik di mana versi rekaman baru kurang dari atau sama dengan versi rekaman yang ada. Dalam hal ini, Timestream tidak akan memperbarui catatan yang ada yang memiliki versi lebih tinggi. Timestream akan menolak catatan baru dengan versi yang lebih rendah atau sama dan menulis kesalahan ini secara asinkron ke bucket S3 Anda. Untuk menerima laporan kesalahan asinkron ini, Anda harus mengatur `MagneticStoreRejectedDataLocation` properti di `MagneticStoreWriteProperties` atas meja Anda.

Pemecahan masalah UNLOAD dari Timestream untuk LiveAnalytics

Berikut ini adalah panduan untuk pemecahan masalah yang terkait dengan perintah. UNLOAD

Kategori	Pesan kesalahan	Cara memecahkan masalah
Panjang kunci S3	UNLOADkunci berkas hasil ketika menggunakan awalan S3 [%s] yang disediakan di tujuan akan melebihi panjang kunci yang diizinkan S3. Lihat dokumentasi untuk lebih jelasnya.	Saat mengekspor hasil kueri menggunakan `UNLOAD` pernyataan, panjang kunci S3, yang terdiri dari jumlah panjang nama bucket S3 dan awalan melebihi panjang kunci S3 maksimum yang didukung. Sebaiknya kurangi prefiks atau panjang nama bucket Anda.
	UNLOADkunci berkas hasil saat menggunakan partition ed_by [%s] akan melebihi panjang kunci yang diizinkan S3. Lihat dokumentasi untuk lebih jelasnya.	Saat mengekspor hasil kueri menggunakan `UNLOAD` pernyataan, panjang Kunci S3 menggunakan kolom partition ed_by melebihi panjang kunci S3 maksimum yang didukung. Kami merekomendasikan untuk partisi dengan kolom

Kategori	Pesan kesalahan	Cara memecahkan masalah
		alternatif atau mengurangi panjang partitioned_column (jika memungkinkan).
	UNLOADkunci berkas hasil ketika menggunakan awalan S3 [%s] bersama dengan partitioned_by [%s] akan melebihi panjang kunci yang diizinkan S3. Lihat dokumentasi untuk lebih jelasnya.	Saat mengekspor hasil kueri menggunakan UNLOAD pernyataan, panjang Kunci S3, yang terdiri dari jumlah panjang nama bucket S3, awalan, dan nama kolom partitioned_by melebihi panjang kunci S3 maksimum yang didukung. Sebaiknya kurangi awalan, panjang nama bucket, atau gunakan kolom alternatif untuk mempartisi data Anda.
	Kunci objek S3 yang dihasilkan: %s terlalu panjang. Lihat dokumentasi untuk lebih jelasnya.	Saat memproses kueri Anda menggunakan UNLOAD pernyataan, salah satu nilai di kolom yang dipartisi melebihi panjang kunci S3 maksimum yang didukung. Kolom partisi dan nilai dapat ditemukan di kunci objek yang dihasilkan.

Kategori	Pesan kesalahan	Cara memecahkan masalah
S3 throttle	Kami telah mendeteksi bahwa Amazon S3 membatasi perintah write from. UNLOAD Lihat dokumentasi Amazon Timestream untuk informasi selengkapnya	Lihat dokumentasi S3 di sini. Tingkat API panggilan S3 dapat dibatasi ketika beberapa pembaca/penulis mengakses folder yang sama. Harap audit volume panggilan ke bucket yang disediakan. Jika Anda menggunakan bucket yang sama untuk beberapa `UNLOAD` kueri bersamaan, coba gunakan bucket yang berbeda untuk hal yang sama. Jika Anda menggunakan bucket yang sama untuk beberapa operasi selain Timestream LiveAnalytics `UNLOAD`, pertimbangkan untuk memindahkan `UNLOAD` hasil ke bucket terpisah.

Timestream untuk LiveAnalytics kode kesalahan tertentu

Bagian ini berisi kode kesalahan khusus untuk Timestream untuk LiveAnalytics.

Timestream untuk kesalahan LiveAnalytics penulisan API

InternalServerException

HTTPKode Status: 500

ThrottlingException

HTTPKode Status: 429

ValidationException

HTTPKode Status: 400

ConflictException

HTTPKode Status: 409

AccessDeniedException

Anda tidak memiliki akses yang memadai untuk melakukan tindakan ini.

HTTPKode Status: 403

ServiceQuotaExceededException

HTTPKode Status: 402

ResourceNotFoundException

HTTPKode Status: 404

RejectedRecordsException

HTTPKode Status: 419

InvalidEndpointException

HTTPKode Status: 421

Timestream untuk kesalahan LiveAnalytics kueri API

ValidationException

HTTPKode Status: 400

QueryExecutionException

HTTPKode Status: 400

ConflictException

HTTPKode Status: 409

ThrottlingException

HTTPKode Status: 429

InternalServerException

HTTPKode Status: 500

InvalidEndpointException

HTTPKode Status: 421

Kuota

Topik ini menjelaskan kuota saat ini, juga disebut sebagai batas, dalam Amazon LiveAnalytics Timestream untuk. Kecuali ditentukan lain, masing-masing kuota berlaku untuk setiap Wilayah.

Topik

Kuota default

Tabel berikut berisi Timestream untuk LiveAnalytics kuota dan nilai default.

displayName	Deskripsi	defaultValue
Basis data per akun	Jumlah maksimum database yang dapat Anda buat per Akun AWS.	500
Tabel per akun	Jumlah maksimum tabel yang dapat Anda buat per Akun AWS.	50000
Tingkat throttle untuk CRUD APIs	Jumlah maksimum API permintaan Create/Update/List	1

displayName	Deskripsi	defaultValue
	/Describe/Delete database/ table/scheduled kueri yang diizinkan per detik per akun, di Wilayah saat ini.	
Kueri terjadwal per akun	Jumlah maksimum kueri terjadwal yang dapat Anda buat per Akun AWS.	10000
Jumlah maksimum partisi penyimpanan magnetik aktif	Jumlah maksimum partisi penyimpanan magnetik aktif per database. Partisi mungkin tetap aktif hingga enam jam setelah menerima konsumsi.	250
maxQueryTCU	Kueri maksimum yang dapat TCUs Anda atur untuk akun Anda.	1000

Batas layanan

Tabel berikut berisi Timestream untuk batas LiveAnalytics layanan dan nilai default. Untuk mengedit penyimpanan data tabel dari konsol, lihat Mengedit tabel.

displayName	Deskripsi	defaultValue
Periode penyerapan di masa mendatang dalam hitungan menit	Waktu tunggu maksimum (dalam menit) untuk data deret waktu Anda dibandingkan dengan waktu sistem saat ini. Misalnya, jika periode konsumsi future adalah 15 menit, maka Timestream for LiveAnalytics akan menerima data yang hingga 15 menit	15

displayName	Deskripsi	defaultValue
	lebih cepat dari waktu sistem saat ini.	
Periode retensi minimum untuk penyimpanan memori dalam jam	Durasi minimum (dalam jam) di mana data harus disimpan di penyimpanan memori per tabel.	1
Periode retensi maksimum untuk penyimpanan memori dalam hitungan jam	Durasi maksimum (dalam jam) dimana data dapat disimpan di penyimpanan memori per tabel.	8766
Periode retensi minimum untuk penyimpanan magnetik dalam beberapa hari	Durasi minimum (dalam hari) di mana data harus disimpan di penyimpanan magnetik per tabel.	1
Periode retensi maksimum untuk penyimpanan magnetik dalam beberapa hari	Durasi maksimum (dalam beberapa hari) di mana data dapat disimpan di penyimpan an magnetik. Nilai ini setara dengan 200 tahun.	73000
Periode retensi default untuk penyimpanan magnetik dalam beberapa hari	Nilai default (dalam hari) yang datanya disimpan di penyimpanan magnetik per tabel. Nilai ini setara dengan 200 tahun.	73000
Periode retensi default untuk penyimpanan memori dalam hitungan jam	Durasi default (dalam jam) yang datanya disimpan di penyimpanan memori.	6
Dimensi per tabel	Jumlah maksimum dimensi per tabel.	128

displayName	Deskripsi	defaultValue
Ukur nama per tabel	Jumlah maksimum nama ukuran unik per tabel.	8192
Nama dimensi ukuran pasangan nilai dimensi per seri	Ukuran maksimum nama dimensi dan pasangan nilai dimensi per seri.	2 Kilobyte
Ukuran rekor maksimum	Ukuran maksimum catatan.	2 Kilobyte
Catatan per WriteRecords API permintaan	Jumlah maksimum catatan dalam WriteRecords API permintaan.	100
Panjang nama dimensi	Jumlah maksimum byte untuk nama Dimensi.	60 byte
Ukur panjang nama	Jumlah maksimum byte untuk nama Measure.	256 byte
Panjang nama basis data	Jumlah maksimum byte untuk nama Database.	256 byte
Panjang nama tabel	Jumlah maksimum byte untuk nama Tabel.	256 byte
QueryString panjang di KiB	Panjang maksimum (dalam KiB) dari string kueri dalam UTF -8 karakter yang dikodekan untuk kueri.	256
Durasi eksekusi untuk kueri dalam jam	Durasi eksekusi maksimum (dalam jam) untuk kueri. Kueri yang memakan waktu lebih lama akan habis waktu.	1

displayName	Deskripsi	defaultValue
Wawasan Kueri	Jumlah maksimum API permintaan Kueri yang diizinkan dengan wawasan kueri diaktifkan per detik per akun, di Wilayah saat ini.	1
Ukuran metadata untuk hasil kueri	Ukuran metadata maksimum untuk hasil kueri.	100 Kilobyte
Ukuran data untuk hasil kueri	Ukuran data maksimum untuk hasil kueri.	5 Gigabyte
Ukuran per catatan multi-ukuran	Jumlah maksimum ukuran per catatan multi-ukuran.	256
Ukur ukuran nilai per catatan multi-ukuran	Ukuran maksimum nilai ukuran per catatan multi-ukuran.	2048
Ukuran unik di seluruh catatan multi-ukuran per tabel	Ukuran unik dalam semua catatan multi-ukuran yang didefinisikan dalam satu tabel.	1024
Unit Komputasi Timestream (TCUs) per akun	Maksimum default TCUs per akun.	200

Jenis data yang didukung

Tabel berikut menjelaskan tipe data yang didukung untuk nilai ukuran dan dimensi.

Deskripsi	Timestream untuk nilai LiveAnalytics
Tipe data yang didukung untuk mengukur nilai.	Int besar, ganda, string, boolean,MULTI, Timestamp

Deskripsi	Timestream untuk nilai LiveAnalytics
Tipe data yang didukung untuk nilai dimensi.	String

Beban batch

Kuota saat ini, juga disebut sebagai batas, dalam beban batch adalah sebagai berikut.

Deskripsi	Timestream untuk nilai LiveAnalytics
Ukuran tugas pemuatan batch maks	Ukuran tugas pemuatan batch maksimum tidak boleh melebihi 100 GB.
Kuantitas file	Tugas pemuatan batch tidak dapat memiliki lebih dari 100 file.
Ukuran maksimum file	Ukuran file maksimum dalam tugas pemuatan batch tidak boleh melebihi 5 GB.
CSVukuran baris file	Baris dalam CSV file tidak boleh melebihi 16 MB. Ini adalah batas keras yang tidak dapat ditingkatkan.
Tugas pemuatan batch aktif	Sebuah tabel tidak dapat memiliki lebih dari 5 tugas pemuatan batch aktif dan akun tidak dapat memiliki lebih dari 10 tugas pemuatan batch aktif. Timestream for LiveAnalytics akan membatasi tugas pemuatan batch baru hingga lebih banyak sumber daya tersedia.

Kendala penamaan

Tabel berikut menjelaskan kendala penamaan.

Deskripsi	Timestream untuk nilai LiveAnalytics
Panjang maksimum nama dimensi.	60 byte

Deskripsi	Timestream untuk nilai LiveAnalytics
Panjang maksimum nama ukuran.	256 byte
Panjang maksimum nama tabel atau nama database.	256 byte
Nama Tabel dan Database	• Kami sarankan Anda tidak menggunakanPengidentifikasi sistem. • Dapat berisi a-z A-Z 0-9 _ (garis bawah) - (tanda hubung). (titik). • Semua nama harus dikodekan sebagai UTF -8, dan peka huruf besar/kecil. **Note** Nama tabel dan database dibandingkan menggunakan representasi biner UTF -8. Ini berarti bahwa perbandingan untuk ASCII karakter bersifat peka huruf besar/kecil.
Ukur Nama	• Tidak boleh mengandung Pengidentifikasi sistem atau titik dua ':'. • Tidak boleh dimulai dengan awalan cadangan (`ts_`,`measure_value`). **Note** Nama tabel dan database dibandingkan menggunakan representasi biner UTF -8. Ini berarti bahwa perbandingan untuk ASCII karakter bersifat peka huruf besar/kecil.

Deskripsi	Timestream untuk nilai LiveAnalytics
Nama Dimensi	• Tidak boleh berisiPengidentifikasi sistem, titik dua ':' atau tanda kutip ganda (“). • Tidak boleh dimulai dengan awalan cadangan (`ts_`,`measure_value`). • Tidak boleh mengandung karakter Unicode [0,31] yang tercantum di sini atau “\ u2028" atau “\ u2029". **Note** Nama dimensi dan ukuran dibandingkan menggunakan representasi biner UTF -8. Ini berarti bahwa perbandingan untuk ASCII karakter bersifat peka huruf besar/kecil.
Semua Nama Kolom	Nama kolom tidak dapat diduplikasi. Karena catatan multi-ukuran mewakili dimensi dan ukuran sebagai kolom, nama untuk dimensi tidak bisa sama dengan nama untuk ukuran. Nilai peka huruf besar/kecil.

Kata kunci terpesan

Semua hal berikut adalah kata kunci yang dicadangkan:

- ALTER
- AND
- AS
- BETWEEN
- OLEH
- CASE
- CAST
- CONSTRAINT
- CREATE

- CROSS
- CUBE
- CURRENT_DATE
- CURRENT_TIME
- CURRENT_TIMESTAMP
- CURRENT_USER
- DEALLOCATE
- DELETE
- DESCRIBE
- DISTINCT
- DROP
- ELSE
- END
- ESCAPE
- EXCEPT
- EXECUTE
- EXISTS
- EXTRACT
- FALSE
- FOR
- FROM
- FULL
- GROUP
- GROUPING
- HAVING
- DI DALAM
- INNER
- INSERT
- INTERSECT
- INTO

- ADALAH
- JOIN
- LEFT
- LIKE
- LOCALTIME
- LOCALTIMESTAMP
- NATURAL
- NORMALIZE
- NOT
- NULL
- PADA
- ATAU
- ORDER
- OUTER
- PREPARE
- RECURSIVE
- RIGHT
- ROLLUP
- SELECT
- TABLE
- THEN
- TRUE
- UESCAPE
- UNION
- UNNEST
- USING
- VALUES
- WHEN
- WHERE
- WITH

Pengidentifikasi sistem

Kami mencadangkan nama kolom “measure _value”, “ts_non_existent_col” dan “time” menjadi Timestream untuk pengidentifikasi sistem. LiveAnalytics Selain itu, nama kolom mungkin tidak dimulai dengan “ts_” atau “measure _name”. Pengidentifikasi sistem peka huruf besar/kecil. Pengidentifikasi dibandingkan menggunakan representasi biner UTF -8. Ini berarti bahwa perbandingan untuk pengidentifikasi peka huruf besar/kecil.

> **Note**
>
> Pengidentifikasi sistem tidak boleh digunakan untuk nama dimensi atau ukuran. Kami menyarankan Anda untuk tidak menggunakan pengidentifikasi sistem untuk database atau nama tabel.

UNLOAD

Untuk batas yang terkait dengan `UNLOAD` perintah, lihat Menggunakan UNLOAD untuk mengekspor hasil kueri ke S3 dari Timestream.

Referensi bahasa kueri

> **Note**
>
> Referensi bahasa kueri ini mencakup dokumentasi pihak ketiga berikut dari Trino Software Foundation (sebelumnya Presto Software Foundation), yang dilisensikan di bawah Lisensi Apache, Versi 2.0. Anda tidak boleh menggunakan file ini kecuali sesuai dengan lisensi ini. Untuk mendapatkan salinan Lisensi Apache, Versi 2.0, lihat situs web Apache.

Timestream untuk LiveAnalytics mendukung bahasa kueri yang kaya untuk bekerja dengan data Anda. Anda dapat melihat tipe data, operator, fungsi, dan konstruksi yang tersedia di bawah ini.

Anda juga dapat segera memulai dengan bahasa kueri Timestream di Kueri Sampel bagian ini.

Topik

- Jenis data yang didukung
- Fungsionalitas deret waktu bawaan

- SQLdukungan
- Operator logis
- Operator perbandingan
- Fungsi perbandingan
- Ekspresi bersyarat
- Fungsi konversi
- Operator matematika
- Fungsi matematika
- Operator String
- Fungsi string
- Operator array
- Fungsi array
- Fungsi bitwise
- Fungsi ekspresi reguler
- Operator tanggal/waktu
- Fungsi tanggal/waktu
- Fungsi agregat
- Fungsi jendela
- Kueri Sampel

Jenis data yang didukung

Timestream untuk LiveAnalytics bahasa query mendukung tipe data berikut.

> **Note**
>
> Tipe data yang didukung untuk penulisan dijelaskan dalam tipe Data.

Tipe data	Deskripsi
`int`	Merupakan bilangan bulat 32-bit.

Tipe data	Deskripsi
`bigint`	Merupakan integer bertanda 64-bit.
`boolean`	Salah satu dari dua nilai kebenaran logika, `True` dan`False`.
`double`	Merupakan tipe data presisi variabel 64-bit. Menerapkan IEEEStandar 754 untuk Binary Floating-Point Arithmetic. **Note** Bahasa query adalah untuk membaca data. Ada fungsi untuk `Infinity` dan nilai `NaN` ganda yang dapat digunakan dalam kueri. Tetapi Anda tidak dapat menulis nilai-nilai itu ke Timestream.
`varchar`	Data karakter panjang variabel dengan ukuran maksimum 2KB.
`array[T,...]`	Berisi satu atau lebih elemen dari tipe data tertentu *T*, di mana *T* dapat berupa salah satu tipe data yang didukung di Timestream.
`row(T,...)`	Berisi satu atau lebih bidang bernama tipe data *T*. Bidang mungkin dari tipe data apa pun yang didukung oleh Timestrea m, dan diakses dengan operator referensi bidang titik: `.`
`date`	Merupakan tanggal dalam formulir*YYYY*`-`*MM*`-`*DD*. di mana *YYYY* adalah tahun, *MM* adalah bulan, dan *DD* adalah hari, masing-masing. Rentang yang didukung adalah dari `1970-01-01` ke`2262-04-11` . Contoh: `1971-02-03`

Tipe data	Deskripsi
`time`	Merupakan waktu dalam sehari UTC. Jenis `time` data direpresentasikan dalam bentuk *HH*`.`*MM*`.`*SS*`.`*sssssssss* . Mendukung presisi nanodetik. Contoh: `17:02:07.496000000`
`timestamp`	Merupakan instance dalam waktu menggunakan waktu presisi nanodetik diUTC. *YYYY*`-`*MM*`-`*DD* *hh*`:`*mm*`:`*ss*`.`*sssssssss* Kueri mendukung stempel waktu dalam kisaran `1677-09-21 00:12:44.000000000` ke. `2262-04-11 23:47:16.854775807`

Tipe data	Deskripsi

`interval`	Merupakan interval waktu sebagai string literal*X*`t`, terdiri dari dua bagian, *X* and `t`.
	
	X adalah nilai numerik yang lebih besar dari atau sama dengan`0`, dan `t` adalah satuan waktu seperti detik atau jam. Unit ini tidak pluralisasi. Satuan waktu `t` harus menjadi salah satu dari literal string berikut:
	
	• `nanosecond`
	• `microsecond`
	• `millisecond`
	• `second`
	• `minute`
	• `hour`
	• `day`
	• `ns`(sama sepertinanosecond)
	• `us`(sama sepertimicrosecond)
	• `ms`(sama sepertimillisecond)
	• `s`(sama sepertisecond)
	• `m`(sama sepertiminute)
	• h(sama sepertihour)
	• d(sama sepertiday)
	
	Contoh:
	
	`17s`
	
	`12second`
	
	`21hour`

Tipe data	Deskripsi
	`2d`
`timeseries[row(timestamp, T,...)]`	Merupakan nilai ukuran yang direkam selama interval waktu sebagai `array` terdiri dari `row` objek. Masing-masing `row` berisi `timestamp` dan satu atau lebih nilai ukuran tipe data *T*, di mana *T* bisa salah satu dari`bigint`,`boolean`,`double`, atau`varchar`. Baris bermacam-macam dalam urutan menaik oleh. `timestamp` Tipe data timeseries mewakili nilai ukuran dari waktu ke waktu.
`unknown`	Merupakan data nol.

Fungsionalitas deret waktu bawaan

Timestream untuk LiveAnalytics menyediakan fungsionalitas deret waktu bawaan yang memperlakukan data deret waktu sebagai konsep kelas satu.

Fungsionalitas deret waktu bawaan dapat dibagi menjadi dua kategori: tampilan dan fungsi.

Anda dapat membaca tentang setiap konstruksi di bawah ini.

Topik

- Tampilan Timeseries
- Fungsi deret waktu

Tampilan Timeseries

Timestream untuk LiveAnalytics mendukung fungsi-fungsi berikut untuk mengubah data Anda ke tipe `timeseries` data:

Topik

- CREATE_TIME_SERIES
- UNNEST

CREATE_TIME_SERIES

CREATE_ TIME _ SERIES adalah fungsi agregasi yang mengambil semua pengukuran mentah dari deret waktu (nilai waktu dan ukuran) dan mengembalikan tipe data timeseries. Sintaks fungsi ini adalah sebagai berikut:

```
CREATE_TIME_SERIES(time, measure_value::<data_type>)
```

dimana `<data_type>` adalah tipe data dari nilai ukuran dan dapat menjadi salah satu dari bigint, boolean, double, atau varchar. Parameter kedua tidak bisa null.

Pertimbangkan CPU pemanfaatan EC2 instance yang disimpan dalam tabel bernama metrik seperti yang ditunjukkan di bawah ini:

Waktu	region	az	vpc	instance_id	ukuran_nama	ukuran_nilai: :ganda
2019-12-04 19:00:00.000000000	us-east-1	kami-timur-1d	vpc-1a2b3c4d	i-1234567890abcdef0	pemanfaatan cpu_	35,0
2019-12-04 19:00:01.000000000	us-east-1	kami-timur-1d	vpc-1a2b3c4d	i-1234567890abcdef0	pemanfaatan cpu_	38.2
2019-12-04 19:00:02.000000000	us-east-1	kami-timur-1d	vpc-1a2b3c4d	i-1234567890abcdef0	pemanfaatan cpu_	45.3
2019-12-04 19:00:00.000000000	us-east-1	kami-timur-1d	vpc-1a2b3c4d	i-1234567890abcdef1	pemanfaatan cpu_	54.1

Waktu	region	az	vpc	instance_id	ukuran_nama	ukuran_nilai: :ganda
2019-12-04 19:00:01.000 000000	us-east-1	kami-timur-1d	vpc-1a2b3c4d	i-1234567890abcdef1	pemanfaatan cpu_	42.5
2019-12-04 19:00:02.000 000000	us-east-1	kami-timur-1d	vpc-1a2b3c4d	i-1234567890abcdef1	pemanfaatan cpu_	33.7

Menjalankan kueri:

```
SELECT region, az, vpc, instance_id, CREATE_TIME_SERIES(time, measure_value::double) as
 cpu_utilization FROM metrics
    WHERE measure_name='cpu_utilization'
    GROUP BY region, az, vpc, instance_id
```

akan mengembalikan semua seri `cpu_utilization` yang memiliki nilai ukuran. Dalam hal ini, kami memiliki dua seri:

region	az	vpc	instance_id	pemanfaatan cpu_
us-east-1	kami-timur-1d	vpc-1a2b3c4d	i-1234567890abcdef0	[{waktu: 2019-12-04 19:00:00.000 000000, ukuran_nilai: :ganda: 35.0}, {waktu: 2019-12-04 19:00:01.000 000000, ukuran_ni

region	az	vpc	instance_id	pemanfaatan cpu_
				lai: :ganda: 38.2}, {waktu: 2019-12-0 4 19:00:02. 000 000000, ukuran_ni lai: :ganda: 45.3}]
us-east-1	kami-timur-1d	vpc-1a2b3c4d	i-1234567 890abcdef1	[{waktu: 2019-12-0 4 19:00:00. 000 000000, ukuran_ni lai: :ganda: 35.1}, {waktu: 2019-12-0 4 19:00:01. 000 000000, ukuran_ni lai: :ganda: 38.5}, {waktu: 2019-12-0 4 19:00:02. 000 000000, ukuran_ni lai: :ganda: 45.7}]

UNNEST

`UNNEST`adalah fungsi tabel yang memungkinkan Anda untuk mengubah `timeseries` data menjadi model datar. Sintaksnya adalah sebagai berikut:

UNNESTmengubah a `timeseries` menjadi dua kolom, yaitu, `time` dan`value`. Anda juga dapat menggunakan alias dengan UNNEST seperti yang ditunjukkan di bawah ini:

```
UNNEST(timeseries) AS <alias_name> (time_alias, value_alias)
```

di <alias_name> mana alias untuk tabel datar, `time_alias` adalah alias untuk `time` kolom dan `value_alias` merupakan alias untuk kolom. `value`

Misalnya, pertimbangkan skenario di mana beberapa EC2 instance di armada Anda dikonfigurasi untuk memancarkan metrik pada interval 5 detik, yang lain memancarkan metrik pada interval 15 detik, dan Anda memerlukan metrik rata-rata untuk semua instance pada granularitas 10 detik selama 6 jam terakhir. Untuk mendapatkan data ini, Anda mengubah metrik Anda ke model deret waktu menggunakan CREATE_ TIME _ SERIES. Anda kemudian dapat menggunakan INTERPOLATE_ LINEAR untuk mendapatkan nilai yang hilang pada granularitas 10 detik. Selanjutnya, Anda mengubah data kembali ke model datar menggunakan UNNEST, dan kemudian gunakan AVGuntuk mendapatkan metrik rata-rata di semua instance.

```
WITH interpolated_timeseries AS (
    SELECT region, az, vpc, instance_id,
        INTERPOLATE_LINEAR(
            CREATE_TIME_SERIES(time, measure_value::double),
                SEQUENCE(ago(6h), now(), 10s)) AS interpolated_cpu_utilization
    FROM timestreamdb.metrics
    WHERE measure_name= 'cpu_utilization' AND time >= ago(6h)
    GROUP BY region, az, vpc, instance_id
)
SELECT region, az, vpc, instance_id, avg(t.cpu_util)
FROM interpolated_timeseries
CROSS JOIN UNNEST(interpolated_cpu_utilization) AS t (time, cpu_util)
GROUP BY region, az, vpc, instance_id
```

Query di atas menunjukkan penggunaan UNNESTdengan alias. Di bawah ini adalah contoh kueri yang sama tanpa menggunakan alias untuk UNNEST:

```
WITH interpolated_timeseries AS (
    SELECT region, az, vpc, instance_id,
        INTERPOLATE_LINEAR(
            CREATE_TIME_SERIES(time, measure_value::double),
                SEQUENCE(ago(6h), now(), 10s)) AS interpolated_cpu_utilization
    FROM timestreamdb.metrics
```

```
    WHERE measure_name= 'cpu_utilization' AND time >= ago(6h)
    GROUP BY region, az, vpc, instance_id
)
SELECT region, az, vpc, instance_id, avg(value)
FROM interpolated_timeseries
CROSS JOIN UNNEST(interpolated_cpu_utilization)
GROUP BY region, az, vpc, instance_id
```

Fungsi deret waktu

Amazon Timestream untuk LiveAnalytics mendukung fungsi timeseries, seperti turunan, integral, dan korelasi, serta lainnya, untuk memperoleh wawasan yang lebih dalam dari data deret waktu Anda. Bagian ini menyediakan informasi penggunaan untuk masing-masing fungsi ini, serta contoh kueri. Pilih topik di bawah ini untuk mempelajari lebih lanjut.

Topik

- Fungsi interpolasi
- Fungsi derivatif
- Fungsi integral
- Fungsi korelasi
- Saring dan kurangi fungsi

Fungsi interpolasi

Jika data deret waktu Anda kehilangan nilai untuk peristiwa pada titik waktu tertentu, Anda dapat memperkirakan nilai peristiwa yang hilang tersebut menggunakan interpolasi. Amazon Timestream mendukung empat varian interpolasi: interpolasi linier, interpolasi spline kubik, interpolasi observasi terakhir dilakukan (locf), dan interpolasi konstan. Bagian ini menyediakan informasi penggunaan untuk Timestream untuk fungsi LiveAnalytics interpolasi, serta kueri sampel.

Informasi penggunaan

Fungsi	Tipe data keluaran	Deskripsi
`interpolate_linear (timeseries, array[timestamp])`	timeseries	Mengisi data yang hilang menggunakan interpolasi linier.

Fungsi	Tipe data keluaran	Deskripsi
`interpolate_linear(timeseries, timestamp)`	double	Mengisi data yang hilang menggunakan interpolasi linier.
`interpolate_spline_cubic(timeseries, array[timestamp])`	timeseries	Mengisi data yang hilang menggunakan interpolasi spline kubik.
`interpolate_spline_cubic(timeseries, timestamp)`	double	Mengisi data yang hilang menggunakan interpolasi spline kubik.
`interpolate_locf(timeseries, array[timestamp])`	timeseries	Mengisi data yang hilang menggunakan nilai sampel terakhir.
`interpolate_locf(timeseries, timestamp)`	double	Mengisi data yang hilang menggunakan nilai sampel terakhir.
`interpolate_fill(timeseries, array[timestamp], double)`	timeseries	Mengisi data yang hilang menggunakan nilai konstan.
`interpolate_fill(timeseries, timestamp, double)`	double	Mengisi data yang hilang menggunakan nilai konstan.

Contoh kueri

Example

Temukan CPU pemanfaatan rata-rata yang di-binned pada interval 30 detik untuk EC2 host tertentu selama 2 jam terakhir, mengisi nilai yang hilang menggunakan interpolasi linier:

```
WITH binned_timeseries AS (
```

```
SELECT hostname, BIN(time, 30s) AS binned_timestamp, ROUND(AVG(measure_value::double),
 2) AS avg_cpu_utilization
FROM "sampleDB".DevOps
WHERE measure_name = 'cpu_utilization'
    AND hostname = 'host-Hovjv'
    AND time > ago(2h)
GROUP BY hostname, BIN(time, 30s)
), interpolated_timeseries AS (
SELECT hostname,
    INTERPOLATE_LINEAR(
        CREATE_TIME_SERIES(binned_timestamp, avg_cpu_utilization),
            SEQUENCE(min(binned_timestamp), max(binned_timestamp), 15s)) AS
 interpolated_avg_cpu_utilization
FROM binned_timeseries
GROUP BY hostname
)
SELECT time, ROUND(value, 2) AS interpolated_cpu
FROM interpolated_timeseries
CROSS JOIN UNNEST(interpolated_avg_cpu_utilization)
```

Example

Temukan CPU pemanfaatan rata-rata yang di-binned pada interval 30 detik untuk EC2 host tertentu selama 2 jam terakhir, mengisi nilai yang hilang menggunakan interpolasi berdasarkan pengamatan terakhir yang dilakukan ke depan:

```
WITH binned_timeseries AS (
SELECT hostname, BIN(time, 30s) AS binned_timestamp, ROUND(AVG(measure_value::double),
 2) AS avg_cpu_utilization
FROM "sampleDB".DevOps
WHERE measure_name = 'cpu_utilization'
    AND hostname = 'host-Hovjv'
    AND time > ago(2h)
GROUP BY hostname, BIN(time, 30s)
), interpolated_timeseries AS (
SELECT hostname,
    INTERPOLATE_LOCF(
        CREATE_TIME_SERIES(binned_timestamp, avg_cpu_utilization),
            SEQUENCE(min(binned_timestamp), max(binned_timestamp), 15s)) AS
 interpolated_avg_cpu_utilization
FROM binned_timeseries
GROUP BY hostname
)
```

```
SELECT time, ROUND(value, 2) AS interpolated_cpu
FROM interpolated_timeseries
CROSS JOIN UNNEST(interpolated_avg_cpu_utilization)
```

Fungsi derivatif

Derivatif digunakan menghitung tingkat perubahan untuk metrik tertentu dan dapat digunakan untuk secara proaktif menanggapi suatu peristiwa. Misalnya, Anda menghitung turunan dari CPU pemanfaatan EC2 instance selama 5 menit terakhir, dan Anda melihat turunan positif yang signifikan. Ini bisa menjadi indikasi meningkatnya permintaan pada beban kerja Anda, jadi Anda mungkin memutuskan ingin memutar lebih banyak EC2 contoh untuk menangani beban kerja Anda dengan lebih baik.

Amazon Timestream mendukung dua varian fungsi turunan. Bagian ini menyediakan informasi penggunaan untuk Timestream untuk fungsi LiveAnalytics turunan, serta contoh kueri.

Informasi penggunaan

Fungsi	Tipe data keluaran	Deskripsi
`derivative_linear( timeseries, interval)`	timeseries	Menghitung turunan dari setiap titik dalam `timeseries` untuk yang ditentukan. `interval`
`non_negative_derivative_linear(timeseries, interval)`	timeseries	Sama seperti`derivative_linear(timeseries, interval)` , tetapi hanya mengembalikan nilai positif.

Contoh kueri

Example

Temukan tingkat perubahan dalam CPU pemanfaatan setiap 5 menit selama 1 jam terakhir:

```
SELECT DERIVATIVE_LINEAR(CREATE_TIME_SERIES(time, measure_value::double), 5m) AS
 result
```

```
FROM “sampleDB”.DevOps
WHERE measure_name = 'cpu_utilization'
AND hostname = 'host-Hovjv' and time > ago(1h)
GROUP BY hostname, measure_name
```

Example

Hitung tingkat peningkatan kesalahan yang dihasilkan oleh satu atau lebih layanan mikro:

```
WITH binned_view as (
    SELECT bin(time, 5m) as binned_timestamp, ROUND(AVG(measure_value::double), 2) as
 value
    FROM “sampleDB”.DevOps
    WHERE micro_service = 'jwt'
    AND time > ago(1h)
    AND measure_name = 'service_error'
    GROUP BY bin(time, 5m)
)
SELECT non_negative_derivative_linear(CREATE_TIME_SERIES(binned_timestamp, value), 1m)
 as rateOfErrorIncrease
FROM binned_view
```

Fungsi integral

Anda dapat menggunakan integral untuk menemukan area di bawah kurva per satuan waktu untuk peristiwa deret waktu Anda. Sebagai contoh, misalkan Anda melacak volume permintaan yang diterima oleh aplikasi Anda per unit waktu. Dalam skenario ini, Anda dapat menggunakan fungsi integral untuk menentukan total volume permintaan yang disajikan per interval tertentu selama periode waktu tertentu.

Amazon Timestream mendukung satu varian fungsi integral. Bagian ini menyediakan informasi penggunaan untuk Timestream untuk fungsi LiveAnalytics integral, serta contoh kueri.

Informasi penggunaan

Fungsi	Tipe data keluaran	Deskripsi
`integral_trapezoidal(timeseries(double))`	double	Perkiraan integral per yang ditentukan `interval day to second` untuk yang

Fungsi	Tipe data keluaran	Deskripsi

`integral_trapezoidal(timeseries(double), interval day to second)`
 	
`integral_trapezoidal(timeseries(bigint))`
 	
`integral_trapezoidal(timeseries(bigint), interval day to second)`
 	
`integral_trapezoidal(timeseries(integer), interval day to second)`
 	
`integral_trapezoidal(timeseries(integer))`
		`timeseries` disediakan, menggunakan aturan trapesium. Parameter interval hari ke kedua adalah opsional dan defaultnya adalah`1s`. Untuk informasi lebih lanjut tentang interval, lihatInterval dan durasi.

Contoh kueri

Example

Hitung total volume permintaan yang dilayani per lima menit selama satu jam terakhir oleh host tertentu:

```
SELECT INTEGRAL_TRAPEZOIDAL(CREATE_TIME_SERIES(time, measure_value::double), 5m) AS
 result FROM sample.DevOps
WHERE measure_name = 'request'
AND hostname = 'host-Hovjv'
AND time > ago (1h)
GROUP BY hostname, measure_name
```

Fungsi korelasi

Mengingat dua deret waktu panjang yang serupa, fungsi korelasi memberikan koefisien korelasi, yang menjelaskan bagaimana tren dua deret waktu dari waktu ke waktu. Koefisien korelasi berkisar dari `-1.0` ke. `1.0` `-1.0`menunjukkan bahwa tren dua deret waktu dalam arah yang berlawanan pada tingkat yang sama. sedangkan `1.0` menunjukkan bahwa tren dua timeseries dalam arah yang sama pada tingkat yang sama. Nilai `0` menunjukkan tidak ada korelasi antara dua deret waktu. Misalnya, jika harga minyak naik, dan harga saham perusahaan minyak meningkat, tren kenaikan harga minyak dan kenaikan harga perusahaan minyak akan memiliki koefisien korelasi positif. Koefisien korelasi positif yang tinggi akan menunjukkan bahwa kedua tren harga pada tingkat yang sama. Demikian pula, koefisien korelasi antara harga obligasi dan imbal hasil obligasi negatif, menunjukkan bahwa kedua nilai ini cenderung berlawanan arah dari waktu ke waktu.

Amazon Timestream mendukung dua varian fungsi korelasi. Bagian ini menyediakan informasi penggunaan untuk Timestream untuk fungsi LiveAnalytics korelasi, serta contoh kueri.

Informasi penggunaan

Fungsi	Tipe data keluaran	Deskripsi
`correlate_pearson( timeseries, timeseries)`	double	Menghitung koefisien korelasi Pearson untuk keduanya. `timeseries` Timeseries harus memiliki stempel waktu yang sama.
`correlate_spearman (timeseries, timeseries)`	double	Menghitung koefisien korelasi Spearman untuk keduanya. `timeseries` Timeseries harus memiliki stempel waktu yang sama.

Contoh kueri

Example

```
WITH cte_1 AS (
    SELECT INTERPOLATE_LINEAR(
```

```
        CREATE_TIME_SERIES(time, measure_value::double),
        SEQUENCE(min(time), max(time), 10m)) AS result
    FROM sample.DevOps
    WHERE measure_name = 'cpu_utilization'
    AND hostname = 'host-Hovjv' AND time > ago(1h)
    GROUP BY hostname, measure_name
),
cte_2 AS (
    SELECT INTERPOLATE_LINEAR(
        CREATE_TIME_SERIES(time, measure_value::double),
        SEQUENCE(min(time), max(time), 10m)) AS result
    FROM sample.DevOps
    WHERE measure_name = 'cpu_utilization'
    AND hostname = 'host-Hovjv' AND time > ago(1h)
    GROUP BY hostname, measure_name
)
SELECT correlate_pearson(cte_1.result, cte_2.result) AS result
FROM cte_1, cte_2
```

Saring dan kurangi fungsi

Amazon Timestream mendukung fungsi untuk melakukan filter dan mengurangi operasi pada data deret waktu. Bagian ini menyediakan informasi penggunaan untuk Timestream untuk LiveAnalytics memfilter dan mengurangi fungsi, serta contoh kueri.

Informasi penggunaan

Fungsi	Tipe data keluaran	Deskripsi
`filter(timeseries(T), function(T, Boolean))`	timeseries (T)	Membangun deret waktu dari deret waktu input, menggunakan nilai-nilai yang diteruskan `function` kembali`true`.
`reduce(timeseries(T), initialState S, inputFunction(S, T, S), outputFunction(S, R))`	R	Mengembalikan nilai tunggal, dikurangi dari deret waktu. `inputFunction` Akan dipanggil pada setiap elemen dalam timeseries secara berurutan. Selain mengambil

Fungsi	Tipe data keluaran	Deskripsi
		elemen saat ini, inputFunction mengambil status saat ini (awalnya `initialState`) dan mengembalikan status baru. `outputFunction` Akan dipanggil untuk mengubah status akhir menjadi nilai hasil. Itu `outputFunction` bisa menjadi fungsi identitas.

Contoh kueri

Example

Bangun deret waktu CPU pemanfaatan host dan titik filter dengan pengukuran lebih besar dari 70:

```
WITH time_series_view AS (
    SELECT INTERPOLATE_LINEAR(
        CREATE_TIME_SERIES(time, ROUND(measure_value::double,2)),
            SEQUENCE(ago(15m), ago(1m), 10s)) AS cpu_user
    FROM sample.DevOps
    WHERE hostname = 'host-Hovjv' and measure_name = 'cpu_utilization'
        AND time > ago(30m)
    GROUP BY hostname
)
SELECT FILTER(cpu_user, x -> x.value > 70.0) AS cpu_above_threshold
from time_series_view
```

Example

Bangun deret waktu CPU pemanfaatan host dan tentukan jumlah kuadrat pengukuran:

```
WITH time_series_view AS (
    SELECT INTERPOLATE_LINEAR(
        CREATE_TIME_SERIES(time, ROUND(measure_value::double,2)),
            SEQUENCE(ago(15m), ago(1m), 10s)) AS cpu_user
    FROM sample.DevOps
```

```
    WHERE hostname = 'host-Hovjv' and measure_name = 'cpu_utilization'
        AND time > ago(30m)
    GROUP BY hostname
)
SELECT REDUCE(cpu_user,
    DOUBLE '0.0',
    (s, x) -> x.value * x.value + s,
    s -> s)
from time_series_view
```

Example

Bangun deret waktu CPU pemanfaatan host dan tentukan fraksi sampel yang berada di atas ambang batas: CPU

```
WITH time_series_view AS (
    SELECT INTERPOLATE_LINEAR(
        CREATE_TIME_SERIES(time, ROUND(measure_value::double,2)),
            SEQUENCE(ago(15m), ago(1m), 10s)) AS cpu_user
    FROM sample.DevOps
    WHERE hostname = 'host-Hovjv' and measure_name = 'cpu_utilization'
        AND time > ago(30m)
    GROUP BY hostname
)
SELECT ROUND(
    REDUCE(cpu_user,
      -- initial state
      CAST(ROW(0, 0) AS ROW(count_high BIGINT, count_total BIGINT)),
      -- function to count the total points and points above a certain threshold
      (s, x) -> CAST(ROW(s.count_high + IF(x.value > 70.0, 1, 0), s.count_total + 1) AS
 ROW(count_high BIGINT, count_total BIGINT)),
      -- output function converting the counts to fraction above threshold
      s -> IF(s.count_total = 0, NULL, CAST(s.count_high AS DOUBLE) / s.count_total)),
    4) AS fraction_cpu_above_threshold
from time_series_view
```

SQLdukungan

Timestream untuk LiveAnalytics mendukung beberapa SQL konstruksi umum. Anda dapat membaca lebih lanjut di bawah ini.

Topik

- SELECT
- Dukungan subquery
- SHOWpernyataan
- DESCRIBEpernyataan
- UNLOAD

SELECT

SELECTpernyataan dapat digunakan untuk mengambil data dari satu atau lebih tabel. Bahasa query Timestream mendukung sintaks berikut untuk SELECTpernyataan:

```
[ WITH with_query [, ...] ]
            SELECT [ ALL | DISTINCT ] select_expr [, ...]
            [ function (expression) OVER (
            [ PARTITION BY partition_expr_list ]
            [ ORDER BY order_list ]
            [ frame_clause ] )
            [ FROM from_item [, ...] ]
            [ WHERE condition ]
            [ GROUP BY [ ALL | DISTINCT ] grouping_element [, ...] ]
            [ HAVING condition]
            [ { UNION | INTERSECT | EXCEPT } [ ALL | DISTINCT ] select ]
            [ ORDER BY order_list ]
            [ LIMIT [ count | ALL ] ]
```

di mana

- `function (expression)`adalah salah satu fungsi jendela yang didukung.
- `partition_expr_list`adalah:

  ```
  expression | column_name [, expr_list ]
  ```

- `order_list`adalah:

  ```
  expression | column_name [ ASC | DESC ]
  [ NULLS FIRST | NULLS LAST ]
  [, order_list ]
  ```

- `frame_clause`adalah:

```
ROWS | RANGE
{ UNBOUNDED PRECEDING | expression PRECEDING | CURRENT ROW } |
{BETWEEN
{ UNBOUNDED PRECEDING | expression { PRECEDING | FOLLOWING } |
CURRENT ROW}
AND
{ UNBOUNDED FOLLOWING | expression { PRECEDING | FOLLOWING } |
CURRENT ROW }}
```

- `from_item`adalah salah satu dari:

```
table_name [ [ AS ] alias [ ( column_alias [, ...] ) ] ]
from_item join_type from_item [ ON join_condition | USING ( join_column [, ...] ) ]
```

- `join_type`adalah salah satu dari:

```
[ INNER ] JOIN
LEFT [ OUTER ] JOIN
RIGHT [ OUTER ] JOIN
FULL [ OUTER ] JOIN
```

- `grouping_element`adalah salah satu dari:

```
()
expression
```

Dukungan subquery

Timestream mendukung subquery dalam `EXISTS` dan predikat. `IN` `EXISTS`Predikat menentukan apakah subquery mengembalikan baris apapun. `IN`Predikat menentukan apakah nilai yang dihasilkan oleh subquery cocok dengan nilai atau ekspresi dalam klausa IN. Bahasa kueri Timestream mendukung subquery berkorelasi dan lainnya.

```
SELECT t.c1
FROM (VALUES 1, 2, 3, 4, 5) AS t(c1)
WHERE EXISTS
(SELECT t.c2
 FROM (VALUES 1, 2, 3) AS t(c2)
 WHERE t.c1= t.c2
)
```

```
ORDER BY t.c1
```

c1
1
2
3

```
SELECT t.c1
FROM (VALUES 1, 2, 3, 4, 5) AS t(c1)
WHERE t.c1 IN
(SELECT t.c2
 FROM (VALUES 2, 3, 4) AS t(c2)
)
ORDER BY t.c1
```

c1
2
3
4

SHOWpernyataan

Anda dapat melihat semua database di akun dengan menggunakan `SHOW DATABASES` pernyataan. Sintaksnya adalah sebagai berikut:

```
SHOW DATABASES [LIKE pattern]
```

di mana `LIKE` klausa dapat digunakan untuk memfilter nama database.

Anda dapat melihat semua tabel di akun dengan menggunakan `SHOW TABLES` pernyataan. Sintaksnya adalah sebagai berikut:

```
SHOW TABLES [FROM database] [LIKE pattern]
```

di mana FROM klausa dapat digunakan untuk memfilter nama database dan LIKE klausa dapat digunakan untuk memfilter nama tabel.

Anda dapat melihat semua ukuran untuk tabel dengan menggunakan SHOW MEASURES pernyataan. Sintaksnya adalah sebagai berikut:

```
SHOW MEASURES FROM database.table [LIKE pattern]
```

di mana FROM klausa akan digunakan untuk menentukan database dan nama tabel dan LIKE klausa dapat digunakan untuk memfilter nama ukuran.

DESCRIBEpernyataan

Anda dapat melihat metadata untuk tabel dengan menggunakan pernyataan. DESCRIBE Sintaksnya adalah sebagai berikut:

```
DESCRIBE database.table
```

dimana table berisi nama tabel. Pernyataan describe mengembalikan nama kolom dan tipe data untuk tabel.

UNLOAD

Timestream untuk LiveAnalytics mendukung UNLOAD perintah sebagai ekstensi untuk SQL dukungannya. Tipe data UNLOAD yang didukung oleh dijelaskan dalamJenis data yang didukung. unknownJenis time dan tidak berlaku untukUNLOAD.

```
UNLOAD (SELECT statement)
 TO 's3://bucket-name/folder'
 WITH ( option = expression [, ...] )
```

di mana opsi adalah

```
{ partitioned_by = ARRAY[ col_name[,…] ]
 | format = [ '{ CSV | PARQUET }' ]
```

```
  | compression = [ '{ GZIP | NONE }' ]
  | encryption = [ '{ SSE_KMS | SSE_S3 }' ]
  | kms_key = '<string>'
  | field_delimiter ='<character>'
  | escaped_by = '<character>'
  | include_header = ['{true, false}']
  | max_file_size = '<value>'
}
```

SELECTpernyataan

Pernyataan query yang digunakan untuk memilih dan mengambil data dari satu atau lebih Timestream untuk LiveAnalytics tabel.

```
(SELECT column 1, column 2, column 3 from database.table
      where measure_name = "ABC" and timestamp between ago (1d) and now() )
```

Klausul TO

```
TO 's3://bucket-name/folder'
```

atau

```
TO 's3://access-point-alias/folder'
```

T0Klausa dalam UNL0AD pernyataan menentukan tujuan untuk output dari hasil query. Anda perlu menyediakan jalur lengkap, termasuk nama ember Amazon S3 atau Amazon S3 dengan lokasi folder di Amazon S3 access-point-alias tempat Timestream untuk menulis objek file output. LiveAnalytics Bucket S3 harus dimiliki oleh akun yang sama dan di wilayah yang sama. Selain set hasil kueri, Timestream untuk LiveAnalytics menulis file manifes dan metadata ke folder tujuan tertentu.

PARTITIONED_BY klausa

```
partitioned_by = ARRAY [col_name[,…] , (default: none)
```

`partitioned_by`Klausa ini digunakan dalam kueri untuk mengelompokkan dan menganalisis data pada tingkat granular. Saat mengekspor hasil kueri ke bucket S3, Anda dapat memilih untuk mempartisi data berdasarkan satu atau beberapa kolom dalam kueri pilih. Saat mempartisi data,

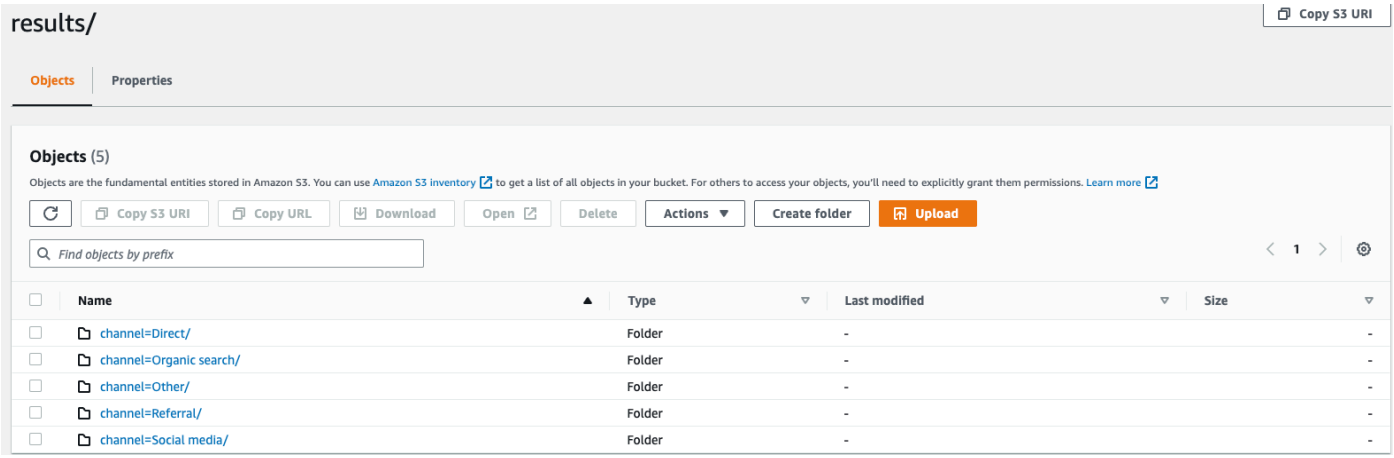
data yang diekspor dibagi menjadi himpunan bagian berdasarkan kolom partisi dan setiap subset disimpan dalam folder terpisah. Dalam folder hasil yang berisi data yang diekspor, sub-folder dibuat `folder/results/partition column = partition value/` secara otomatis. Namun, perhatikan bahwa kolom yang dipartisi tidak termasuk dalam file output.

`partitioned_by`bukan klausa wajib dalam sintaks. Jika Anda memilih untuk mengekspor data tanpa partisi apa pun, Anda dapat mengecualikan klausa dalam sintaks.

Example

Dengan asumsi Anda memantau data clickstream situs web Anda dan memiliki 5 saluran lalu lintas yaitu`direct`,,, `Social MediaOrganic Search`, `Other` dan. `Referral` Saat mengekspor data, Anda dapat memilih untuk mempartisi data menggunakan kolom`Channel`. Dalam folder data Anda`s3://bucketname/results`,, Anda akan memiliki lima folder masing-masing dengan nama saluran masing-masing, misalnya, `s3://bucketname/results/channel=Social Media/`. Dalam folder ini Anda akan menemukan data semua pelanggan yang mendarat di situs web Anda melalui `Social Media` saluran. Demikian pula, Anda akan memiliki folder lain untuk saluran yang tersisa.

Data yang diekspor dipartisi oleh kolom Saluran



FORMAT

```
format = [ '{ CSV | PARQUET }' , default: CSV
```

Kata kunci untuk menentukan format hasil kueri yang ditulis ke bucket S3 Anda. Anda dapat mengekspor data baik sebagai nilai dipisahkan koma (CSV) menggunakan koma (,) sebagai pembatas default atau dalam format Apache Parquet, format penyimpanan kolom terbuka yang efisien untuk analitik.

COMPRESSION

```
compression = [ '{ GZIP | NONE }' ], default: GZIP
```

Anda dapat mengompres data yang diekspor menggunakan algoritma kompresi GZIP atau membuatnya tidak dikompresi dengan menentukan opsi. `NONE`

ENCRYPTION

```
encryption = [ '{ SSE_KMS | SSE_S3 }' ], default: SSE_S3
```

File output di Amazon S3 dienkripsi menggunakan opsi enkripsi yang Anda pilih. Selain data Anda, file manifes dan metadata juga dienkripsi berdasarkan opsi enkripsi yang Anda pilih. Saat ini kami mendukung SSE enkripsi _S3 dan SSE _KMS. SSE_S3 adalah enkripsi sisi server dengan Amazon S3 mengenkripsi data menggunakan enkripsi standar enkripsi lanjutan () 256-bit. AES SSE_ KMS adalah enkripsi sisi server untuk mengenkripsi data menggunakan kunci yang dikelola pelanggan.

KMS_KEY

```
kms_key = '<string>'
```

KMSKunci adalah kunci yang ditentukan pelanggan untuk mengenkripsi hasil kueri yang diekspor. KMSKey dikelola dengan aman oleh AWS Key Management Service (AWS KMS) dan digunakan untuk mengenkripsi file data di Amazon S3.

FIELD_DELIMITER

```
field_delimiter ='<character>' , default: (,)
```

Saat mengekspor data dalam CSV format, bidang ini menentukan satu ASCII karakter yang digunakan untuk memisahkan bidang dalam file output, seperti karakter pipa (|), koma (,), atau tab (/t). Pembatas default untuk CSV file adalah karakter koma. Jika nilai dalam data Anda berisi pembatas yang dipilih, pembatas akan dikutip dengan karakter kutipan. Misalnya, jika nilai dalam data Anda berisi`Time,stream`, maka nilai ini akan dikutip seperti `"Time,stream"` pada data yang diekspor. Karakter kutipan yang digunakan oleh Timestream untuk LiveAnalytics adalah tanda kutip ganda (“).

Hindari menentukan karakter carriage return (ASCII13, hex`0D`, text '\ r') atau karakter line break (ASCII10, hex 0A, text'\n') sebagai `FIELD_DELIMITER` jika Anda ingin memasukkan header

dalamCSV, karena itu akan mencegah banyak parser dari dapat mengurai header dengan benar dalam output yang dihasilkan. CSV

ESCAPED_OLEH

```
escaped_by = '<character>', default: (\)
```

Saat mengekspor data dalam CSV format, bidang ini menentukan karakter yang harus diperlakukan sebagai karakter escape dalam file data yang ditulis ke bucket S3. Melarikan diri terjadi dalam skenario berikut:

1. Jika nilai itu sendiri berisi karakter kutipan (“) maka itu akan diloloskan menggunakan karakter escape. Misalnya, jika nilainya`Time"stream`, di mana (\) adalah karakter escape yang dikonfigurasi, maka itu akan lolos sebagai`Time\"stream`.
2. Jika nilai berisi karakter escape dikonfigurasi, itu akan lolos. Misalnya, jika nilainya`Time \stream`, maka itu akan lolos sebagai`Time\\stream`.

> **Note**
>
> Jika output yang diekspor berisi tipe data yang kompleks seperti Array, Rows atau Timeseries, itu akan diserialisasi sebagai string. JSON Berikut adalah contohnya.

Tipe data	Nilai aktual	Bagaimana nilai diloloskan dalam CSV format [serial stringJSON]
Array	`[ 23,24,25 ]`	`"[23,24,25]"`
Baris	`( x=23.0, y=hello )`	`"{\"x\":23.0,\"y\":\"hello\"}"`
Timeseries	`[ ( time=1970-01-01 00:00:00.000000010, value=100.0 ), ( time=1970-01-01 00:00:00.000000012, value=120.0 ) ]`	`"[{\"time\":\"1970-01-01 00:00:00.000000010Z\",\"value\":100.0},{\"time\":\"1970-01-01 00:00:00.000000012`

Tipe data	Nilai aktual	Bagaimana nilai diloloskan dalam CSV format [serial stringJSON]
		`Z\",\"value\":120.0}]"`

INCLUDE_HEADER

```
include_header = 'true' , default: 'false'
```

Saat mengekspor data dalam CSV format, bidang ini memungkinkan Anda menyertakan nama kolom sebagai baris pertama dari file CSV data yang diekspor.

Nilai yang diterima adalah 'true' dan 'false' dan nilai default adalah 'false'. Opsi transformasi teks seperti `escaped_by` dan `field_delimiter` berlaku untuk header juga.

> **Note**
>
> Saat menyertakan header, penting bahwa Anda tidak memilih karakter carriage return (ASCII13, hex 0D, text '\ r') atau karakter pemisah baris (ASCII10, hex 0A, teks'\n') sebagai`FIELD_DELIMITER`, karena itu akan mencegah banyak parser untuk dapat mengurai header dengan benar dalam output yang dihasilkan. CSV

MAX_FILE_SIZE

```
max_file_size = 'X[MB|GB]' , default: '78GB'
```

Bidang ini menentukan ukuran maksimum file yang dibuat `UNLOAD` pernyataan di Amazon S3. `UNLOAD`Pernyataan tersebut dapat membuat beberapa file tetapi ukuran maksimum setiap file yang ditulis ke Amazon S3 akan kira-kira apa yang ditentukan dalam bidang ini.

Nilai bidang harus antara 16 MB dan 78 GB, inklusif. Anda dapat menentukannya dalam bilangan bulat seperti`12GB`, atau dalam desimal seperti atau. `0.5GB` `24.7MB` Nilai default adalah 78 GB.

Ukuran file sebenarnya diperkirakan saat file sedang ditulis, sehingga ukuran maksimum sebenarnya mungkin tidak persis sama dengan angka yang Anda tentukan.

Operator logis

Timestream untuk LiveAnalytics mendukung operator logis berikut.

Operator	Deskripsi	Contoh
AND	Benar jika kedua nilai benar	a AND b
ATAU	Benar jika salah satu nilai benar	a ATAU b
NOT	Benar jika nilainya salah	NOTsebuah

- Hasil `AND` perbandingan mungkin `NULL` jika salah satu atau kedua sisi ekspresi tersebut`NULL`.
- Jika setidaknya satu sisi `AND` operator adalah ekspresi `FALSE` yang dievaluasi. `FALSE`
- Hasil `OR` perbandingan mungkin `NULL` jika salah satu atau kedua sisi ekspresi tersebut`NULL`.
- Jika setidaknya satu sisi `OR` operator adalah ekspresi `TRUE` yang dievaluasi. `TRUE`
- Pelengkap logis dari `NULL` adalah`NULL`.

Tabel kebenaran berikut menunjukkan penanganan `NULL` in `AND` dan`OR`:

A	B	A dan b	A atau b
null	null	null	nol
false	null	false	null
nol	false	false	null
true	null	nol	true
null	true	null	true
SALAH	false	false	SALAH
BETUL	SALAH	SALAH	BETUL

A	B	A dan b	A atau b
SALAH	BETUL	SALAH	true
true	true	true	true

Tabel kebenaran berikut menunjukkan penanganan NULL inNOT:

A	Bukan
null	nol
true	SALAH
SALAH	true

Operator perbandingan

Timestream untuk LiveAnalytics mendukung operator perbandingan berikut.

Operator	Deskripsi
<	Kurang dari
>	Lebih besar dari
<=	Kurang dari atau sama dengan
>=	Lebih besar dari atau sama dengan
=	Sama
<>	Tidak sama
!=	Tidak sama

> **Note**
>
> - `BETWEEN`Operator menguji apakah nilai berada dalam rentang yang ditentukan. Sintaksnya adalah sebagai berikut:
>
>   ```
>   BETWEEN min AND max
>   ```
>
>   Kehadiran `NULL` dalam `NOT BETWEEN` pernyataan `BETWEEN` atau akan menghasilkan pernyataan yang dievaluasi. `NULL`
> - `IS NULL` dan `IS NOT NULL` operator menguji apakah suatu nilai adalah null (undefined). Menggunakan `NULL` dengan `IS NULL` evaluasi menjadi benar.
> - DalamSQL, `NULL` nilai menandakan nilai yang tidak diketahui.

Fungsi perbandingan

Timestream untuk LiveAnalytics mendukung fungsi perbandingan berikut.

Topik

- terbesar ()
- paling sedikit ()
- ALL(), ANY () dan SOME ()

terbesar ()

Fungsi greatest () mengembalikan nilai terbesar dari nilai yang disediakan. Ia kembali `NULL` jika salah satu nilai yang disediakan`NULL`. Sintaksnya adalah sebagai berikut.

```
greatest(value1, value2, ..., valueN)
```

paling sedikit ()

Fungsi least () mengembalikan terkecil dari nilai yang disediakan. Ia kembali `NULL` jika salah satu nilai yang disediakan`NULL`. Sintaksnya adalah sebagai berikut.

```
least(value1, value2, ..., valueN)
```

ALL(), ANY () dan SOME ()

TheALL, ANY dan SOME quantifiers dapat digunakan bersama dengan operator perbandingan dengan cara berikut.

Ekspresi	Arti
A = ALL (...)	Mengevaluasi ke true ketika A sama dengan semua nilai.
A <> ALL (...)	Mengevaluasi ke true ketika A tidak cocok dengan nilai apa pun.
A < ALL (...)	Mengevaluasi ke true ketika A lebih kecil dari nilai terkecil.
A = ANY (...)	Mengevaluasi ke true ketika A sama dengan salah satu nilai.
A <> ANY (...)	Mengevaluasi ke true ketika A tidak cocok dengan satu atau lebih nilai.
A < ANY (...)	Mengevaluasi ke true ketika A lebih kecil dari nilai terbesar.

Contoh dan catatan penggunaan

> **Note**
>
> Saat menggunakan ALLSOME, ANY atau, kata kunci VALUES harus digunakan jika nilai perbandingan adalah daftar literal.

Contoh: **ANY()**

Contoh ANY( ) dalam pernyataan query sebagai berikut.

```
SELECT 11.7 = ANY (VALUES 12.0, 13.5, 11.7)
```

Sintaks alternatif untuk operasi yang sama adalah sebagai berikut.

```
SELECT 11.7 = ANY (SELECT 12.0 UNION ALL SELECT 13.5 UNION ALL SELECT 11.7)
```

Dalam hal ini, `ANY()` evaluasi untuk`True`.

Contoh: **`ALL()`**

Contoh `ALL()` dalam pernyataan query sebagai berikut.

```
SELECT 17 < ALL (VALUES 19, 20, 15);
```

Sintaks alternatif untuk operasi yang sama adalah sebagai berikut.

```
SELECT 17 < ALL (SELECT 19 UNION ALL SELECT 20 UNION ALL SELECT 15);
```

Dalam hal ini, `ALL()` evaluasi untuk`False`.

Contoh: **`SOME()`**

Contoh `SOME()` dalam pernyataan query sebagai berikut.

```
SELECT 50 >= SOME (VALUES 53, 77, 27);
```

Sintaks alternatif untuk operasi yang sama adalah sebagai berikut.

```
SELECT 50 >= SOME (SELECT 53 UNION ALL SELECT 77 UNION ALL SELECT 27);
```

Dalam hal ini, `SOME()` evaluasi untuk`True`.

Ekspresi bersyarat

Timestream untuk LiveAnalytics mendukung ekspresi kondisional berikut.

Topik

- CASEPernyataan
- Pernyataan IF
- COALESCEPernyataan
- NULLIFPernyataan
- TRYPernyataan

CASEPernyataan

CASEPernyataan mencari setiap ekspresi nilai dari kiri ke kanan sampai menemukan satu yang sama`expression`. Jika menemukan kecocokan, hasil untuk nilai yang cocok dikembalikan. Jika tidak ada kecocokan yang ditemukan, hasil dari `ELSE` klausa dikembalikan jika ada; jika tidak `null` dikembalikan. Sintaksnya adalah sebagai berikut:

```
CASE expression
    WHEN value THEN result
    [ WHEN ... ]
    [ ELSE result ]
END
```

Timestream juga mendukung sintaks berikut untuk CASEpernyataan. Dalam sintaks ini, formulir "dicari" mengevaluasi setiap kondisi boolean dari kiri ke kanan sampai salah satunya `true` dan mengembalikan hasil yang cocok. Jika tidak ada kondisi`true`, hasil dari `ELSE` klausa dikembalikan jika ada; jika tidak `null` dikembalikan. Lihat di bawah untuk sintaks alternatif:

```
CASE
    WHEN condition THEN result
    [ WHEN ... ]
    [ ELSE result ]
END
```

Pernyataan IF

Pernyataan IF mengevaluasi kondisi menjadi benar atau salah dan mengembalikan nilai yang sesuai. Timestream mendukung dua representasi sintaks berikut untuk IF:

```
if(condition, true_value)
```

Sintaks ini mengevaluasi dan mengembalikan `true_value` jika kondisi adalah`true`; jika tidak `null` dikembalikan dan tidak `true_value` dievaluasi.

```
if(condition, true_value, false_value)
```

Sintaks ini mengevaluasi dan mengembalikan `true_value` jika kondisi`true`, jika tidak mengevaluasi dan mengembalikan. `false_value`

Contoh

```
SELECT
  if(true, 'example 1'),
  if(false, 'example 2'),
  if(true, 'example 3 true', 'example 3 false'),
  if(false, 'example 4 true', 'example 4 false')
```

_col0	_col1	_col2	_col3
example 1	- null	example 3 true	example 4 false

COALESCEPernyataan

COALESCEmengembalikan nilai non-null pertama dalam daftar argumen. Sintaksnya adalah sebagai berikut:

```
coalesce(value1, value2[,...])
```

NULLIFPernyataan

Pernyataan IF mengevaluasi kondisi menjadi benar atau salah dan mengembalikan nilai yang sesuai. Timestream mendukung dua representasi sintaks berikut untuk IF:

NULLIFmengembalikan null jika `value1` sama`value2`; jika tidak maka kembali. `value1` Sintaksnya adalah sebagai berikut:

```
nullif(value1, value2)
```

TRYPernyataan

TRYFungsi mengevaluasi ekspresi dan menangani jenis kesalahan tertentu dengan mengembalikan`null`. Sintaksnya adalah sebagai berikut:

```
try(expression)
```

Fungsi konversi

Timestream untuk LiveAnalytics mendukung fungsi konversi berikut.

Topik

pemeran ()

Sintaks fungsi cast untuk secara eksplisit memberikan nilai sebagai tipe adalah sebagai berikut.

```
cast(value AS type)
```

try_cast ()

Timestream for LiveAnalytics juga mendukung fungsi try_cast yang mirip dengan cast tetapi mengembalikan null jika cast gagal. Sintaksnya adalah sebagai berikut.

```
try_cast(value AS type)
```

Operator matematika

Timestream untuk LiveAnalytics mendukung operator matematika berikut.

Operator	Deskripsi
+	Penambahan
-	Pengurangan
*	Perkalian
/	Pembagian (pembagian bilangan bulat melakukan pemotongan)
%	Modulus (sisa)

Fungsi matematika

Timestream untuk LiveAnalytics mendukung fungsi matematika berikut.

Fungsi	Tipe data keluaran	Deskripsi
perut (x)	[sama seperti masukan]	Mengembalikan nilai absolut x.
cbrt (x)	double	Mengembalikan akar kubus dari x.
langit-langit (x) atau ceil (x)	[sama seperti masukan]	Mengembalikan x dibulatkan ke integer terdekat.
derajat (x)	double	Mengubah sudut x dalam radian ke derajat.
e ()	double	Mengembalikan nomor Euler konstan ini.
exp (x)	double	Mengembalikan nomor Euler dinaikkan ke kekuatan x.
lantai (x)	[sama seperti masukan]	Mengembalikan x dibulatkan ke bawah ke integer terdekat.
from_base (string, radix)	bigint	Mengembalikan nilai string ditafsirkan sebagai nomor basis-radix.
ln (x)	double	Mengembalikan logaritma natural dari x.
log2 (x)	double	Mengembalikan basis 2 logaritma x.
log10 (x)	double	Mengembalikan basis 10 logaritma x.

Fungsi	Tipe data keluaran	Deskripsi
mod (n, m)	[sama seperti masukan]	Mengembalikan modulus (sisa) dari n dibagi dengan m.
pi ()	double	Mengembalikan Pi konstan.
pow (x, p) atau daya (x, p)	double	Mengembalikan x dinaikkan ke kekuatan p.
radian (x)	double	Mengubah sudut x dalam derajat menjadi radian.
rand () atau acak ()	double	Mengembalikan nilai pseudo-acak dalam kisaran 0.0 1.0.
acak (n)	[sama seperti masukan]	Mengembalikan nomor pseudo-acak antara 0 dan n (eksklusif).
bulat (x)	[sama seperti masukan]	Mengembalikan x dibulatkan ke integer terdekat.
bulat (x, d)	[sama seperti masukan]	Mengembalikan x dibulatkan ke tempat desimal d.

Fungsi	Tipe data keluaran	Deskripsi
tanda (x)	[sama seperti masukan]	Mengembalikan fungsi signum dari x, yaitu: • 0 jika argumennya adalah 0 • 1 jika argumennya lebih besar dari 0 • -1 jika argumennya kurang dari 0. Untuk argumen ganda, fungsi tambahan mengembalikan: • NaN jika argumennya NaN • 1 jika argumennya adalah +Infinity • -1 jika argumennya adalah -Infinity.
persegi (x)	double	Mengembalikan akar kuadrat dari x.
to_base (x, radix)	varchar	Mengembalikan representasi basis-radix x.
memotong (x)	double	Mengembalikan x dibulatkan ke integer dengan menjatuhk an digit setelah titik desimal.
acos (x)	double	Mengembalikan arc cosinus x.
asin (x)	double	Mengembalikan sinus busur x.
tan (x)	double	Mengembalikan tangen busur dari x.

Fungsi	Tipe data keluaran	Deskripsi
atan2 (y, x)	double	Mengembalikan tangen busur dari y/x.
cos (x)	double	Mengembalikan cosinus x.
cosh (x)	double	Mengembalikan kosinus hiperbolik x.
dosa (x)	double	Mengembalikan sinus dari x.
tan (x)	double	Mengembalikan garis singgung x.
tanh (x)	double	Mengembalikan tangen hiperbolik x.
tak terhingga ()	double	Mengembalikan konstanta yang mewakili tak terhingga positif.
adalah_terbatas (x)	boolean	Tentukan apakah x terbatas.
adalah_tak terbatas (x)	boolean	Tentukan apakah x tidak terbatas.
is_nan (x)	boolean	Tentukan apakah x adalah not-a-number.
nan ()	double	Mengembalikan konstanta mewakili not-a-number.

Operator String

Timestream untuk LiveAnalytics mendukung `||` operator untuk menggabungkan satu atau lebih string.

Fungsi string

> **Note**
>
> Tipe data input dari fungsi-fungsi ini diasumsikan varchar kecuali ditentukan lain.

Fungsi	Tipe data keluaran	Deskripsi
chr (n)	varchar	Mengembalikan kode Unicode titik n sebagai varchar.
titik kode (x)	integer	Mengembalikan titik kode Unicode dari satu-satunya karakter str.
concat (x1,..., xN)	varchar	Mengembalikan rangkaian x1, x2,..., xN.
hamming_jarak (x1, x2)	bigint	Mengembalikan jarak Hamming x1 dan x2, yaitu jumlah posisi di mana karakter yang sesuai berbeda. Perhatikan bahwa dua input varchar harus memiliki panjang yang sama.
panjang (x)	bigint	Mengembalikan panjang x dalam karakter.
levenshtein_distance (x1, x2)	bigint	Mengembalikan jarak edit Levenshtein x1 dan x2, yaitu jumlah minimum suntingan karakter tunggal (penyisipan, penghapusan atau substitusi) yang diperlukan untuk mengubah x1 menjadi x2.

Fungsi	Tipe data keluaran	Deskripsi
lebih rendah (x)	varchar	Mengkonversi x ke huruf kecil.
lpad (x1, ukuran besar, x2)	varchar	Bantalan kiri x1 untuk ukuran karakter dengan x2. Jika ukuran kurang dari panjang x1, hasilnya dipotong menjadi karakter ukuran. ukuran tidak boleh negatif dan x2 harus tidak kosong.
ltrim (x)	varchar	Menghapus spasi utama dari x.
ganti (x1, x2)	varchar	Menghapus semua contoh x2 dari x1.
ganti (x1, x2, x3)	varchar	Mengganti semua instance x2 dengan x3 di x1.
Membalikkan (x)	varchar	Mengembalikan x dengan karakter dalam urutan terbalik.
rpad (x1, ukuran besar, x2)	varchar	Bantalan kanan x1 untuk ukuran karakter dengan x2. Jika ukuran kurang dari panjang x1, hasilnya dipotong menjadi karakter ukuran. ukuran tidak boleh negatif dan x2 harus tidak kosong.
rtrim (x)	varchar	Menghapus spasi belakang dari x.
terbelah (x1, x2)	array (varchar)	Membagi x1 pada pembatas x2 dan mengembalikan array.

Fungsi	Tipe data keluaran	Deskripsi
split (x1, x2, batas besar)	array (varchar)	Membagi x1 pada pembatas x2 dan mengembalikan array. Elemen terakhir dalam array selalu berisi semua yang tersisa di x1. batas harus berupa angka positif.
split_part (x1, x2, pos kecil)	varchar	Membagi x1 pada pembatas x2 dan mengembalikan bidang varchar di pos. Indeks lapangan dimulai dengan 1. Jika pos lebih besar dari jumlah bidang, maka null dikembalikan.
strpos (x1, x2)	bigint	Mengembalikan posisi awal dari contoh pertama x2 di x1. Posisi dimulai dengan 1. Jika tidak ditemukan, 0 dikembali kan.
strpos (x1, x2, contoh bigint)	bigint	Mengembalikan posisi instance Nth dari x2 dalam x1. Contoh harus berupa angka positif. Posisi dimulai dengan 1. Jika tidak ditemukan, 0 dikembalikan.
strrpos (x1, x2)	bigint	Mengembalikan posisi awal dari contoh terakhir dari x2 di x1. Posisi dimulai dengan 1. Jika tidak ditemukan, 0 dikembalikan.

Fungsi	Tipe data keluaran	Deskripsi
strrpos (x1, x2, contoh bigint)	bigint	Mengembalikan posisi instance Nth dari x2 di x1 mulai dari akhir x1. misalnya harus angka positif. Posisi dimulai dengan 1. Jika tidak ditemukan, 0 dikembalikan.
posisi (x2 IN x1)	bigint	Mengembalikan posisi awal dari contoh pertama x2 di x1. Posisi dimulai dengan 1. Jika tidak ditemukan, 0 dikembali kan.
substr (x, awal kecil)	varchar	Mengembalikan sisa x dari awal posisi awal. Posisi dimulai dengan 1. Posisi awal negatif ditafsirkan sebagai relatif terhadap akhir x.
substr (x, awal kecil, besar len)	varchar	Mengembalikan substring dari x panjang len dari awal posisi awal. Posisi dimulai dengan 1. Posisi awal negatif ditafsirkan sebagai relatif terhadap akhir x.
memangkas (x)	varchar	Menghapus spasi putih depan dan belakang dari x.
atas (x)	varchar	Mengkonversi x ke huruf besar.

Operator array

Timestream untuk LiveAnalytics mendukung operator array berikut.

Operator	Deskripsi
[]	Akses elemen array di mana indeks pertama dimulai pada 1.
||	Menggabungkan array dengan array lain atau elemen dari jenis yang sama.

Fungsi array

Timestream untuk LiveAnalytics mendukung fungsi array berikut.

Fungsi	Tipe data keluaran	Deskripsi
array_distinct (x)	array	Hapus nilai duplikat dari array x. `SELECT array_distinct(ARRAY[1,2,2,3])` Contoh hasil: `[ 1,2,3 ]`
array_berpotongan (x, y)	array	Mengembalikan array elemen di persimpangan x dan y, tanpa duplikat. `SELECT array_intersect(ARRAY[1,2,3], ARRAY[3,4,5])` Contoh hasil: `[ 3 ]`
array_union (x, y)	array	Mengembalikan array elemen dalam penyatuan x dan y, tanpa duplikat.

Fungsi	Tipe data keluaran	Deskripsi
		`SELECT array_union(ARRAY[1,2,3], ARRAY[3,4,5])` Contoh hasil: `[ 1,2,3,4,5 ]`
array_kecuali (x, y)	array	Mengembalikan array elemen dalam x tetapi tidak di y, tanpa duplikat. `SELECT array_except(ARRAY[1,2,3], ARRAY[3,4,5])` Contoh hasil: `[ 1,2 ]`
array_join (x, pembatas, null_replacement)	varchar	Menggabungkan elemen array yang diberikan menggunakan pembatas dan string opsional untuk menggantikan nol. `SELECT array_join(ARRAY[1,2,3], ';', '')` Contoh hasil: `1;2;3`
array_maks (x)	sama seperti elemen array	Mengembalikan nilai maksimum array input. `SELECT array_max(ARRAY[1,2,3])` Contoh hasil: 3

Fungsi	Tipe data keluaran	Deskripsi
array_min (x)	sama seperti elemen array	Mengembalikan nilai minimum array input. `SELECT array_min (ARRAY[1,2,3])` Contoh hasil: 1
array_position (x, elemen)	bigint	Mengembalikan posisi kemunculan pertama dari elemen dalam array x (atau 0 jika tidak ditemukan). `SELECT array_pos ition(ARRAY[3,4,5,9], 5)` Contoh hasil: 3
array_remove (x, elemen)	array	Hapus semua elemen yang sama elemen dari array x. `SELECT array_rem ove(ARRAY[3,4,5,9], 4)` Contoh hasil: `[ 3,5,9 ]`

Fungsi	Tipe data keluaran	Deskripsi
array_sort (x)	array	Mengurutkan dan mengembalikan array x. Unsur-unsur x harus dapat diurutkan. Elemen null akan ditempatkan di akhir array dikembalikan. `SELECT array_sort(ARRAY[6,8,2,9,3])` Contoh hasil: `[ 2,3,6,8,9 ]`
arrays_tumpang tindih (x, y)	boolean	Menguji apakah array x dan y memiliki elemen non-null yang sama. Mengembalikan null jika tidak ada elemen non-null yang sama tetapi salah satu array berisi null. `SELECT arrays_overlap(ARRAY[6,8,2,9,3], ARRAY[6,8])` Contoh hasil: `true`
kardinalitas (x)	bigint	Mengembalikan ukuran array x. `SELECT cardinality(ARRAY[6,8,2,9,3])` Contoh hasil: 5

Fungsi	Tipe data keluaran	Deskripsi
concat (array1, array2,..., ArrayN)	array	Menggabungkan array array1, array2,..., ArrayN. `SELECT concat(ARRAY[6,8,2,9,3], ARRAY[11,32], ARRAY[6,8,2,0,14])` Contoh hasil: `[ 6,8,2,9,3,11,32,6,8,2,0,14 ]`
element_at (array (E), indeks)	E	Mengembalikan elemen array pada indeks yang diberikan . Jika indeks < 0, element_at mengakses elemen dari yang terakhir ke yang pertama. `SELECT element_at(ARRAY[6,8,2,9,3], 1)` Contoh hasil: 6
ulangi (elemen, hitung)	array	Ulangi elemen untuk menghitung waktu. `SELECT repeat(1, 3)` Contoh hasil: `[ 1,1,1 ]`

Fungsi	Tipe data keluaran	Deskripsi
terbalik (x)	array	Mengembalikan array yang memiliki urutan terbalik array x. `SELECT reverse(ARRAY[6,8,2,9,3])` Contoh hasil: `[ 3,9,2,8,6 ]`
urutan (mulai, berhenti)	array (kecil)	Hasilkan urutan bilangan bulat dari awal hingga berhenti, bertambah 1 jika start kurang dari atau sama dengan berhenti, jika tidak -1. `SELECT sequence(3, 8)` Contoh hasil: `[ 3,4,5,6,7,8 ]`
urutan (mulai, berhenti, langkah)	array (kecil)	Hasilkan urutan bilangan bulat dari awal hingga berhenti, bertambah demi langkah. `SELECT sequence(3, 15, 2)` Contoh hasil: `[ 3,5,7,9,11,13,15 ]`

Fungsi	Tipe data keluaran	Deskripsi
urutan (mulai, berhenti)	array (stempel waktu)	Hasilkan urutan stempel waktu dari tanggal mulai hingga tanggal berhenti, bertambah 1 hari. `SELECT sequence( '2023-04-02 19:26:12.941000000', '2023-04-06 19:26:12.941000000', 1d)` Contoh hasil: `[ 2023-04-02 19:26:12.941000000 ,2023-04-03 19:26:12.941000000 ,2023-04-04 19:26:12.941000000 ,2023-04-05 19:26:12.941000000 ,2023-04-06 19:26:12.941000000 ]`

Fungsi	Tipe data keluaran	Deskripsi
urutan (mulai, berhenti, langkah)	array (stempel waktu)	Hasilkan urutan stempel waktu dari awal hingga berhenti, bertambah demi langkah. Tipe data langkah adalah interval. `SELECT sequence( '2023-04-02 19:26:12.941000000', '2023-04-10 19:26:12.941000000', 2d)` Contoh hasil: `[ 2023-04-02 19:26:12.941000000 ,2023-04-04 19:26:12.941000000 ,2023-04-06 19:26:12.941000000 ,2023-04-08 19:26:12.941000000 ,2023-04-10 19:26:12.941000000 ]`
kocokan (x)	array	Hasilkan permutasi acak dari array x yang diberikan. `SELECT shuffle(ARRAY[6,8,2,9,3])` Contoh hasil: `[ 6,3,2,9,8 ]`

Fungsi	Tipe data keluaran	Deskripsi
irisan (x, mulai, panjang)	array	Subset array x dimulai dari indeks awal (atau mulai dari akhir jika start negatif) dengan panjang panjang. `SELECT slice(ARRAY[6,8,2,9,3], 1, 3)` Contoh hasil: `[ 6,8,2 ]`
zip (array1, array2 [,...])	array (baris)	Menggabungkan array yang diberikan, dari segi elemen, ke dalam satu array baris. Jika argumen memiliki panjang yang tidak rata, nilai yang hilang diisi denganNULL. `SELECT zip(ARRAY[6,8,2,9,3], ARRAY[15, 24])` Contoh hasil: `[ ( 6, 15 ), ( 8, 24 ),( 2, - ), ( 9, - ),( 3, - ) ]`

Fungsi bitwise

Timestream untuk LiveAnalytics mendukung fungsi bitwise berikut.

Fungsi	Tipe data keluaran	Deskripsi
bit_count (kecil, kecil)	bigint (pelengkap dua)	Mengembalikan jumlah bit dalam parameter bigint pertama di mana parameter

Fungsi	Tipe data keluaran	Deskripsi

		kedua adalah integer bertanda bit seperti 8 atau 64.
		`SELECT bit_count(19, 8)`
		Contoh hasil: `3`
		`SELECT bit_count(19, 2)`
		Contoh hasil: `Number must be representable with the bits specified . 19 can not be represented with 2 bits`

bitwise_and (kecil, kecil)	bigint (pelengkap dua)	Mengembalikan bitwise AND dari parameter bigint. `SELECT bitwise_and(12, 7)` Contoh hasil: `4`
bitwise_not (kecil)	bigint (pelengkap dua)	Mengembalikan bitwise NOT dari parameter bigint. `SELECT bitwise_not(12)` Contoh hasil: `-13`

Fungsi	Tipe data keluaran	Deskripsi
bitwise_or (kecil, kecil)	bigint (pelengkap dua)	Mengembalikan bitwise OR dari parameter bigint. `SELECT bitwise_or(12, 7)` Contoh hasil: 15
bitwise_xor (kecil, kecil)	bigint (pelengkap dua)	Mengembalikan bitwise XOR dari parameter bigint. `SELECT bitwise_xor(12, 7)` Contoh hasil: 11

Fungsi ekspresi reguler

Fungsi ekspresi reguler di Timestream untuk LiveAnalytics mendukung sintaks pola Java. Timestream untuk LiveAnalytics mendukung fungsi ekspresi reguler berikut.

Fungsi	Tipe data keluaran	Deskripsi
regexp_extract_all (string, pola)	array (varchar)	Mengembalikan substring (s) cocok dengan pola ekspresi reguler dalam string. `SELECT regexp_extract_all('example expect complex', 'ex\w')` Contoh hasil: `[ exa,exp ]`
regexp_extract_all (string, pola, grup)	array (varchar)	Menemukan semua kemunculan pola ekspresi

Fungsi	Tipe data keluaran	Deskripsi
		reguler dalam string dan mengembalikan kelompok nomor grup menangkap. `SELECT regexp_extract_all('example expect complex', '(ex)(\w)', 2)` Contoh hasil: `[ a,p ]`
regexp_extract (string, pola)	varchar	Mengembalikan substring pertama cocok dengan pola ekspresi reguler dalam string. `SELECT regexp_extract('example expect', 'ex\w')` Contoh hasil: `exa`
regexp_extract (string, pola, grup)	varchar	Menemukan kemunculan pertama dari pola ekspresi reguler dalam string dan mengembalikan kelompok nomor grup menangkap. `SELECT regexp_extract('example expect', '(ex)(\w)', 2)` Contoh hasil: a

Fungsi	Tipe data keluaran	Deskripsi
regexp_like (string, pola)	boolean	Mengevaluasi pola ekspresi reguler dan menentuka n apakah itu terkandung dalam string. Fungsi ini mirip dengan LIKE operator, kecuali bahwa pola hanya perlu terkandung dalam string, daripada perlu mencocokk an semua string. Dengan kata lain, ini melakukan operasi berisi daripada operasi pertandingan. Anda dapat mencocokkan seluruh string dengan menambatkan pola menggunakan ^ dan $. `SELECT regexp_li ke('example', 'ex')` Contoh hasil: `true`
regexp_replace (string, pola)	varchar	Menghapus setiap instance substring yang cocok dengan pola ekspresi reguler dari string. `SELECT regexp_re place('example  expect', 'expect')` Contoh hasil: `example`

Fungsi	Tipe data keluaran	Deskripsi
regexp_replace (string, pola, penggantian)	varchar	Mengganti setiap instance substring yang dicocokkan dengan pola regex dalam string dengan pengganti an. Grup penangkapan dapat direferensikan sebagai pengganti menggunakan $g untuk grup bernomor atau ${name} untuk grup bernama. Tanda dolar ($) dapat dimasukkan dalam pengganti an dengan melarikan diri dengan garis miring terbalik (\$). `SELECT regexp_replace('example expect', 'expect', 'surprise')` Contoh hasil: `example surprise`

Fungsi	Tipe data keluaran	Deskripsi
regexp_replace (string, pola, fungsi)	varchar	Menggantikan setiap instance substring yang cocok dengan pola ekspresi reguler dalam string menggunakan fungsi. Fungsi ekspresi lambda dipanggil untuk setiap kecocokan dengan grup penangkap yang diteruskan sebagai array. Menangkap nomor grup dimulai dari satu; tidak ada grup untuk seluruh pertandingan (jika Anda membutuhkan ini, kelilingi seluruh ekspresi dengan tanda kurung). `SELECT regexp_replace('example', '(\w)', x -> upper(x[1]))` Contoh hasil: `EXAMPLE`
regexp_split (string, pola)	array (varchar)	Membagi string menggunakan pola ekspresi reguler dan mengembalikan array. String kosong tertinggal dipertahankan. `SELECT regexp_split('example', 'x')` Contoh hasil: `[ e,ample ]`

Operator tanggal/waktu

> **Note**
>
> Timestream for LiveAnalytics tidak mendukung nilai waktu negatif. Setiap operasi yang mengakibatkan waktu negatif menghasilkan kesalahan.

Timestream untuk LiveAnalytics mendukung operasi berikut pada`timestamps`,`dates`, dan`intervals`.

Operator	Deskripsi
+	Penambahan
-	Pengurangan

Topik

- Operasi
- Penambahan
- Pengurangan

Operasi

Jenis hasil operasi didasarkan pada operan. Interval literal seperti `1day` dan `3s` dapat digunakan.

```
SELECT date '2022-05-21' + interval '2' day
```

```
SELECT date '2022-05-21' + 2d
```

```
SELECT date '2022-05-21' + 2day
```

Contoh hasil untuk masing-masing: `2022-05-23`

Unit interval meliputi`second`,`minute`,`hour`,`day`,`week`,`month`, dan`year`. Tetapi dalam beberapa kasus tidak semua berlaku. Misalnya detik, menit, dan jam tidak dapat ditambahkan atau dikurangi dari tanggal.

```
SELECT interval '4' year + interval '2' month
```

Contoh hasil: `4-2`

```
SELECT typeof(interval '4' year + interval '2' month)
```

Contoh hasil: `interval year to month`

Jenis hasil operasi interval mungkin `'interval year to month'` atau `'interval day to second'` tergantung pada operan. Interval dapat ditambahkan atau dikurangi dari `dates` dan. `timestamps` Tetapi a `date` atau `timestamp` tidak dapat ditambahkan atau dikurangi dari atau. `date` `timestamp` Untuk menemukan interval atau durasi yang terkait dengan tanggal atau stempel waktu, lihat `date_diff` dan fungsi terkait di. Interval dan durasi

Penambahan

Example

```
SELECT date '2022-05-21' + interval '2' day
```

Contoh hasil: `2022-05-23`

Example

```
SELECT typeof(date '2022-05-21' + interval '2' day)
```

Contoh hasil: `date`

Example

```
SELECT interval '2' year + interval '4' month
```

Contoh hasil: `2-4`

Example

```
SELECT typeof(interval '2' year + interval '4' month)
```

Contoh hasil: `interval year to month`

Pengurangan

Example

```
SELECT timestamp '2022-06-17 01:00' - interval '7' hour
```

Contoh hasil: `2022-06-16 18:00:00.000000000`

Example

```
SELECT typeof(timestamp '2022-06-17 01:00' - interval '7' hour)
```

Contoh hasil: `timestamp`

Example

```
SELECT interval '6' day - interval '4' hour
```

Contoh hasil: `5 20:00:00.000000000`

Example

```
SELECT typeof(interval '6' day - interval '4' hour)
```

Contoh hasil: `interval day to second`

Fungsi tanggal/waktu

> **Note**
>
> Timestream for LiveAnalytics tidak mendukung nilai waktu negatif. Setiap operasi yang mengakibatkan waktu negatif menghasilkan kesalahan.

Timestream untuk LiveAnalytics menggunakan UTC zona waktu untuk tanggal dan waktu. Timestream mendukung fungsi-fungsi berikut untuk tanggal dan waktu.

Topik

Umum dan konversi

Timestream untuk LiveAnalytics mendukung fungsi umum dan konversi berikut untuk tanggal dan waktu.

Fungsi	Tipe data keluaran	Deskripsi
current_date	date	Mengembalikan tanggal saat ini diUTC. Tidak ada tanda kurung yang digunakan. `SELECT current_date` Contoh hasil: `2022-07-07` **Note** Ini juga merupakan kata kunci yang dicadangkan. Untuk daftar kata kunci yang dicadangkan, lihatKata kunci terpesan.
current_time	Waktu	Mengembalikan waktu saat ini diUTC. Tidak ada tanda kurung yang digunakan.

Fungsi	Tipe data keluaran	Deskripsi
		`SELECT current_time` Contoh hasil: `17:41:52.827000000` **Note** Ini juga merupakan kata kunci yang dicadangkan. Untuk daftar kata kunci yang dicadangkan, lihatKata kunci terpesan.
current_timestamp atau now ()	timestamp	Mengembalikan stempel waktu saat ini di. UTC `SELECT current_timestamp` Contoh hasil: `2022-07-07 17:42:32.939000000` **Note** Ini juga merupakan kata kunci yang dicadangkan. Untuk daftar kata kunci yang dicadangkan, lihatKata kunci terpesan.

Fungsi	Tipe data keluaran	Deskripsi
current_timezone ()	varchar Nilainya akan menjadi 'UTC.'	Timestream menggunakan UTC zona waktu untuk tanggal dan waktu. `SELECT current_timezone()` Contoh hasil: `UTC`
tanggal (varchar (x)), tanggal (stempel waktu)	date	`SELECT date(TIMESTAMP '2022-07-07 17:44:43.771000000')` Contoh hasil: `2022-07-07`
last_day_of_month (stempel waktu), last_day_of_month (tanggal)	date	`SELECT last_day_of_month(TIMESTAMP '2022-07-07 17:44:43.771000000')` Contoh hasil: `2022-07-31`
dari_iso8601_timestamp (string)	timestamp	Mem-parsing stempel waktu ISO 8601 ke dalam format stempel waktu internal. `SELECT from_iso8601_timestamp('2022-06-17T08:04:05.000000000+05:00')` Contoh hasil: `2022-06-17 03:04:05.000000000`

Fungsi	Tipe data keluaran	Deskripsi
dari_iso8601_date (string)	date	Mem-parsing string tanggal ISO 8601 ke dalam format stempel waktu internal untuk UTC 00:00:00 dari tanggal yang ditentukan. `SELECT from_iso8601_date('2022-07-17')` Contoh hasil: `2022-07-17`
to_iso8601 (stempel waktu), to_iso8601 (tanggal)	varchar	Mengembalikan string ISO 8601 diformat untuk input. `SELECT to_iso8601(from_iso8601_date('2022-06-17'))` Contoh hasil: `2022-06-17`
dari_milidetik (bigint)	timestamp	`SELECT from_milliseconds(1)` Contoh hasil: `1970-01-01 00:00:00.001000000`
dari_nanodetik (besar)	timestamp	`select from_nanoseconds(300000001)` Contoh hasil: `1970-01-01 00:00:00.300000001`

Fungsi	Tipe data keluaran	Deskripsi
from_unixtime (ganda)	timestamp	Mengembalikan timestamp yang sesuai dengan unixtime yang disediakan. `SELECT from_unixtime(1)` Contoh hasil: `1970-01-01 00:00:01.000000000`
waktu lokal	Waktu	Mengembalikan waktu saat ini diUTC. Tidak ada tanda kurung yang digunakan. `SELECT localtime` Contoh hasil: `17:58:22.654000000` **Note** Ini juga merupakan kata kunci yang dicadangkan. Untuk daftar kata kunci yang dicadangkan, lihatKata kunci terpesan.

Fungsi	Tipe data keluaran	Deskripsi
localtimestamp	timestamp	Mengembalikan stempel waktu saat ini di. UTC Tidak ada tanda kurung yang digunakan. `SELECT localtimestamp` Contoh hasil: `2022-07-07 17:59:04.368000000` **Note** Ini juga merupakan kata kunci yang dicadangkan. Untuk daftar kata kunci yang dicadangkan, lihatKata kunci terpesan.

to_milliseconds (interval hari ke detik), to_milliseconds (stempel waktu)	bigint	`SELECT to_millis econds(INTERVAL '2' DAY + INTERVAL '3' HOUR)`
		Contoh hasil: `183600000`
		`SELECT to_millis econds(TIMESTAMP '2022-06-17 17:44:43. 771000000')`
		Contoh hasil: `1655487883771`

Fungsi	Tipe data keluaran	Deskripsi

to_nanoseconds (interval hari ke detik), to_nanoseconds (stempel waktu)	bigint	`SELECT to_nanoseconds(INTERVAL '2' DAY + INTERVAL '3' HOUR)`
		
		Contoh hasil: `183600000000000`
		
		`SELECT to_nanoseconds(TIMESTAMP '2022-06-17 17:44:43.771000678')`
		
		Contoh hasil: `1655487883771000678`

to_unixtime (stempel waktu)	double	Mengembalikan unixtime untuk stempel waktu yang disediakan. `SELECT to_unixtime('2022-06-17 17:44:43.771000000')` Contoh hasil: `1.655487883771E9`

Fungsi	Tipe data keluaran	Deskripsi
date_trunc (satuan, stempel waktu)	timestamp	Mengembalikan stempel waktu terpotong ke unit, di mana unit adalah salah satu dari [detik, menit, jam, hari, minggu, bulan, kuartal, atau tahun]. `SELECT date_trun c('minute', TIMESTAMP '2022-06-17 17:44:43. 771000000')` Contoh hasil: `2022-06-17 17:44:00.000000000`

Interval dan durasi

Timestream untuk LiveAnalytics mendukung fungsi interval dan durasi berikut untuk tanggal dan waktu.

Fungsi	Tipe data keluaran	Deskripsi
date_add (unit, bigint, tanggal), date_add (unit, bigint, waktu), date_add (varchar (x), bigint, stempel waktu)	timestamp	Menambahkan bigint unit, di mana unit adalah salah satu dari [detik, menit, jam, hari, minggu, bulan, kuartal, atau tahun]. `SELECT date_add('hour', 9, TIMESTAMP '2022-06-17 00:00:00')` Contoh hasil: `2022-06-17 09:00:00.000000000`

Fungsi	Tipe data keluaran	Deskripsi
date_diff (satuan, tanggal, tanggal), date_diff (satuan, waktu, waktu), date_diff (unit, stempel waktu, stempel waktu)	bigint	Mengembalikan perbedaan, di mana unit adalah salah satu dari [detik, menit, jam, hari, minggu, bulan, kuartal, atau tahun]. `SELECT date_diff('day', DATE '2020-03-01', DATE '2020-03-02')` Contoh hasil: 1

parse_duration (string)	interval	Mem-parsing string input untuk mengembalikan yang `interval` setara.
		`SELECT parse_duration('42.8ms')`
		Contoh hasil: `0 00:00:00.042800000`
		`SELECT typeof(parse_duration('42.8ms'))`
		Contoh hasil: `interval day to second`

Fungsi	Tipe data keluaran	Deskripsi
bin (stempel waktu, interval)	timestamp	Membulatkan nilai integer `timestamp` parameter ke kelipatan terdekat dari nilai integer `interval` parameter. Arti dari nilai pengembalian ini mungkin tidak jelas. Ini dihitung menggunakan aritmatika integer terlebih dahulu dengan membagi bilangan bulat timestamp dengan integer interval dan kemudian dengan mengalikan hasilnya dengan integer interval. Mengingat bahwa stempel waktu menentukan UTC titik waktu sebagai jumlah pecahan detik yang berlalu sejak POSIX zaman (1 Januari 1970), nilai pengembalian jarang akan sejajar dengan satuan kalender. Misalnya, jika Anda menentukan interval 30 hari, semua hari sejak zaman dibagi menjadi kenaikan 30 hari, dan awal kenaikan 30 hari terbaru dikembalikan, yang tidak memiliki hubungan dengan bulan kalender. Berikut ini adalah beberapa contohnya:

Fungsi	Tipe data keluaran	Deskripsi

		`bin(TIMESTAMP '2022-06-17 10:15:20', 5m) ==> 2022-06-17 10:15:00.000000000`
		`bin(TIMESTAMP '2022-06-17 10:15:20', 1d) ==> 2022-06-17 00:00:00.000000000`
		`bin(TIMESTAMP '2022-06-17 10:15:20', 10day) ==> 2022-06-17 00:00:00.000000000`
		`bin(TIMESTAMP '2022-06-17 10:15:20', 30day) ==> 2022-05-28 00:00:00.000000000`

lalu (interval)	timestamp	Mengembalikan nilai yang sesuai dengan `interval` current_timestamp. `SELECT ago(1d)` Contoh hasil: `2022-07-06 21:08:53.245000000`
interval literal seperti 1h, 1d, dan 30m	interval	Literal interval adalah kemudahan untuk parse_duration (string). Misalnya, `1d` sama dengan `parse_duration('1d')` . Ini memungkinkan penggunaan literal di mana pun interval digunakan. Misalnya, `ago(1d)` dan `bin(<timestamp> , 1m)`.

Beberapa literal interval bertindak sebagai singkatan untuk parse_duration. Misalnya,`parse_duration('1day')`,`1day`,`parse_duration('1d')`, dan 1d masing-masing kembali `1 00:00:00.000000000` di mana jenisnya`interval day to second`. Ruang diperbolehkan dalam format yang disediakan untuk`parse_duration`. Misalnya `parse_duration('1day')` juga kembali`00:00:00.000000000`. Tapi `1 day` ini bukan interval literal.

Satuan yang terkait dengan `interval day to second` adalah ns, nanodetik, us, mikrodetik, ms, milidetik, s, detik, m, menit, h, jam, d, dan hari.

Ada juga`interval year to month`. Satuan yang terkait dengan interval tahun ke bulan adalah y, tahun, dan bulan. Misalnya, `SELECT 1year` pengembalian`1-0`. `SELECT 12month`juga kembali`1-0`. `SELECT 8month`kembali`0-8`.

Meskipun unit juga `quarter` tersedia untuk beberapa fungsi seperti `date_trunc` dan`date_add`, tidak `quarter` tersedia sebagai bagian dari interval literal.

Memformat dan mengurai

Timestream untuk LiveAnalytics mendukung fungsi pemformatan dan penguraian berikut untuk tanggal dan waktu.

Fungsi	Tipe data keluaran	Deskripsi
date_format (stempel waktu, varchar (x))	varchar	Untuk informasi selengkapnya tentang penentu format yang digunakan oleh fungsi ini, lihat # https://trino.io/docs/current/functions/datetime.html mysql-date-functions `SELECT date_format(TIMESTAMP '2019-10-20 10:20:20', '%Y-%m-%d %H:%i:%s')` Contoh hasil: `2019-10-20 10:20:20`

Fungsi	Tipe data keluaran	Deskripsi
date_parse (varchar (x), varchar (y))	timestamp	Untuk informasi selengkapnya tentang penentu format yang digunakan oleh fungsi ini, lihat # https://trino.io/docs/current/functions/datetime.html mysql-date-functions `SELECT date_parse('2019-10-20 10:20:20', '%Y-%m-%d %H:%i:%s')` Contoh hasil: `2019-10-20 10:20:20.000000000`
format_datetime (stempel waktu, varchar (x))	varchar	Untuk informasi selengkapnya tentang string format yang digunakan oleh fungsi ini, lihat http://joda-time.sourceforge.net/apidocs/org/joda/time/format/DateTimeFormat.html `SELECT format_datetime(parse_datetime('1968-01-13 12', 'yyyy-MM-dd HH'), 'yyyy-MM-dd HH')` Contoh hasil: `1968-01-13 12`

Fungsi	Tipe data keluaran	Deskripsi
parse_datetime (varchar (x), varchar (y))	timestamp	Untuk informasi selengkap nya tentang string format yang digunakan oleh fungsi ini, lihat http://joda-time.sourceforge.net/apidocs/org/joda/time/format/DateTimeFormat.html `SELECT parse_datetime('2019-12-29 10:10 PST', 'uuuu-LL-dd HH:mm z')` Contoh hasil: `2019-12-29 18:10:00.000000000`

Ekstraksi

Timestream untuk LiveAnalytics mendukung fungsi ekstraksi berikut untuk tanggal dan waktu. Fungsi ekstrak adalah dasar untuk fungsi kenyamanan yang tersisa.

Fungsi	Tipe data keluaran	Deskripsi
sari	bigint	Mengekstrak bidang dari stempel waktu, di mana bidang adalah salah satu dari [YEAR,,QUARTER,, MONTHWEEK, DAY _OF_DAY,MONTH, DAY _OF_,WEEK, _OF_DOW,, DAY YEARDOY, YEAR atau]. WEEK YOW HOUR MINUTE SECOND

Fungsi	Tipe data keluaran	Deskripsi
		`SELECT extract(YEAR FROM '2019-10-12 23:10:34.000000000')` Contoh hasil: 2019
hari (stempel waktu), hari (tanggal), hari (interval hari ke detik)	bigint	`SELECT day('2019-10-12 23:10:34.000000000')` Contoh hasil: 12
day_of_month (stempel waktu), day_of_month (tanggal), day_of_month (interval hari ke detik)	bigint	`SELECT day_of_month('2019-10-12 23:10:34.000000000')` Contoh hasil: 12
day_of_week (stempel waktu), day_of_week (tanggal)	bigint	`SELECT day_of_week('2019-10-12 23:10:34.000000000')` Contoh hasil: 6
day_of_year (stempel waktu), day_of_year (tanggal)	bigint	`SELECT day_of_year('2019-10-12 23:10:34.000000000')` Contoh hasil: 285
dow (stempel waktu), dow (tanggal)	bigint	Alias untuk day_of_week
doy (stempel waktu), doy (tanggal)	bigint	Alias untuk day_of_year

Fungsi	Tipe data keluaran	Deskripsi
jam (stempel waktu), jam (waktu), jam (interval hari ke detik)	bigint	`SELECT hour('2019-10-12 23:10:34.000000000')` Contoh hasil: `23`
milidetik (stempel waktu), milidetik (waktu), milidetik (interval hari ke detik)	bigint	`SELECT millisecond('2019-10-12 23:10:34.000000000')` Contoh hasil: `0`
menit (timestamp), menit (waktu), menit (interval hari ke detik)	bigint	`SELECT minute('2019-10-12 23:10:34.000000000')` Contoh hasil: `10`
bulan (stempel waktu), bulan (tanggal), bulan (interval tahun ke bulan)	bigint	`SELECT month('2019-10-12 23:10:34.000000000')` Contoh hasil: `10`
nanodetik (stempel waktu), nanodetik (waktu), nanodetik (interval hari ke detik)	bigint	`SELECT nanosecond(current_timestamp)` Contoh hasil: `162000000`
kuartal (stempel waktu), kuartal (tanggal)	bigint	`SELECT quarter('2019-10-12 23:10:34.000000000')` Contoh hasil: `4`

Fungsi	Tipe data keluaran	Deskripsi
kedua (stempel waktu), detik (waktu), detik (interval hari ke detik)	bigint	`SELECT second('2019-10-12 23:10:34.000000000')` Contoh hasil: `34`
minggu (stempel waktu), minggu (tanggal)	bigint	`SELECT week('2019-10-12 23:10:34.000000000')` Contoh hasil: `41`
week_of_year (stempel waktu), week_of_year (tanggal)	bigint	Alias untuk minggu
tahun (stempel waktu), tahun (tanggal), tahun (interval tahun ke bulan)	bigint	`SELECT year('2019-10-12 23:10:34.000000000')` Contoh hasil: `2019`
year_of_week (stempel waktu), year_of_week (tanggal)	bigint	`SELECT year_of_week('2019-10-12 23:10:34.000000000')` Contoh hasil: `2019`
yow (stempel waktu), yow (tanggal)	bigint	Alias untuk year_of_week

Fungsi agregat

Timestream untuk LiveAnalytics mendukung fungsi agregat berikut.

Fungsi	Tipe data keluaran	Deskripsi
sewenang-wenang (x)	[sama seperti masukan]	Mengembalikan nilai non-null arbitrer x, jika ada. `SELECT arbitrary(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Contoh hasil: 1
array_agg (x)	array < [sama dengan masukan]	Mengembalikan array dibuat dari elemen input x. `SELECT array_agg(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Contoh hasil: `[ 1,2,3,4 ]`
rata-rata (x)	double	Mengembalikan rata-rata (rata-rata aritmatika) dari semua nilai masukan. `SELECT avg(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Contoh hasil: `2.5`
bool_and (boolean) setiap (boolean)	boolean	Mengembalikan TRUE jika setiap nilai masukanTRUE, jika tidakFALSE. `SELECT bool_and(t.c) FROM (VALUES true, true, false, true) AS t(c)`

Fungsi	Tipe data keluaran	Deskripsi
		Contoh hasil: `false`
bool_or (boolean)	boolean	Mengembalikan TRUE jika ada nilai masukanTRUE, jika tidakFALSE. `SELECT bool_or(t.c) FROM (VALUES true, true, false, true) AS t(c)` Contoh hasil: `true`
menghitung (*) hitungan (x)	bigint	count (*) mengembalikan jumlah baris input. count (x) mengembalikan jumlah nilai input non-null. `SELECT count(t.c) FROM (VALUES true, true, false, true) AS t(c)` Contoh hasil: `4`
hitungan_jika (x)	bigint	Mengembalikan jumlah nilai TRUE masukan. `SELECT count_if(t.c) FROM (VALUES true, true, false, true) AS t(c)` Contoh hasil: `3`

Fungsi	Tipe data keluaran	Deskripsi
geometrik_mean (x)	double	Mengembalikan rata-rata geometris dari semua nilai masukan. `SELECT geometric _mean(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Contoh hasil: `2.2133638 39400643`
max_by (x, y)	[sama seperti x]	Mengembalikan nilai x terkait dengan nilai maksimum y atas semua nilai masukan. `SELECT max_by(t.c1, t.c2) FROM (VALUES (('a', 1)), (('b', 2)), (('c', 3)), (('d', 4))) AS t(c1, c2)` Contoh hasil: d
maks_by (x, y, n)	array< [same as x] >	Mengembalikan n nilai x terkait dengan n terbesar dari semua nilai masukan y dalam urutan menurun y. `SELECT max_by(t.c1, t.c2, 2) FROM (VALUES (('a', 1)), (('b', 2)), (('c', 3)), (('d', 4))) AS t(c1, c2)` Contoh hasil: `[ d,c ]`

Fungsi	Tipe data keluaran	Deskripsi
min_by (x, y)	[sama seperti x]	Mengembalikan nilai x terkait dengan nilai minimum y atas semua nilai masukan. `SELECT min_by(t.c1, t.c2) FROM (VALUES (('a', 1)), (('b', 2)), (('c', 3)), (('d', 4))) AS t(c1, c2)` Contoh hasil: a
min_by (x, y, n)	array< [same as x] >	Mengembalikan n nilai x terkait dengan n terkecil dari semua nilai masukan y dalam urutan menaik y. `SELECT min_by(t.c1, t.c2, 2) FROM (VALUES (('a', 1)), (('b', 2)), (('c', 3)), (('d', 4))) AS t(c1, c2)` Contoh hasil: `[ a,b ]`
maks (x)	[sama seperti masukan]	Mengembalikan nilai maksimum dari semua nilai masukan. `SELECT max(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Contoh hasil: 4

Fungsi	Tipe data keluaran	Deskripsi
maks (x, n)	array< [same as x] >	Mengembalikan n nilai terbesar dari semua nilai masukan x. `SELECT max(t.c, 2) FROM (VALUES 1, 2, 3, 4) AS t(c)` Contoh hasil: `[ 4,3 ]`
min (x)	[sama seperti masukan]	Mengembalikan nilai minimum dari semua nilai masukan. `SELECT min(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Contoh hasil: 1
min (x, n)	array< [same as x] >	Mengembalikan n nilai terkecil dari semua nilai masukan x. `SELECT min(t.c, 2) FROM (VALUES 1, 2, 3, 4) AS t(c)` Contoh hasil: `[ 1,2 ]`
jumlah (x)	[sama seperti masukan]	Mengembalikan jumlah semua nilai masukan. `SELECT sum(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Contoh hasil: `10`

Fungsi	Tipe data keluaran	Deskripsi
bitwise_and_agg (x)	bigint	Mengembalikan bitwise AND dari semua nilai masukan dalam representasi komplemen 2s. `SELECT bitwise_and_agg(t.c) FROM (VALUES 1, -3) AS t(c)` Contoh hasil: 1
bitwise_or_agg (x)	bigint	Mengembalikan bitwise OR dari semua nilai masukan dalam representasi komplemen 2s. `SELECT bitwise_or_agg(t.c) FROM (VALUES 1, -3) AS t(c)` Contoh hasil: -3

Fungsi	Tipe data keluaran	Deskripsi
kira-kira_beda (x)	bigint	Mengembalikan perkiraan jumlah nilai masukan yang berbeda. Fungsi ini memberikan perkiraan hitungan (DISTINCTx). Nol dikembalikan jika semua nilai masukan nol. Fungsi ini harus menghasilkan kesalahan standar 2,3%, yang merupakan standar deviasi dari distribusi kesalahan (kira-kira normal) di semua set yang mungkin. Itu tidak menjamin batas atas pada kesalahan untuk setiap set input tertentu. `SELECT approx_distinct(t.c) FROM (VALUES 1, 2, 3, 4, 8) AS t(c)` Contoh hasil: 5

Fungsi	Tipe data keluaran	Deskripsi
kira-kira_berbeda (x, e)	bigint	Mengembalikan perkiraan jumlah nilai masukan yang berbeda. Fungsi ini memberikan perkiraan hitungan (DISTINCTx). Nol dikembalikan jika semua nilai masukan nol. Fungsi ini harus menghasilkan kesalahan standar tidak lebih dari e, yang merupakan standar deviasi dari distribusi kesalahan (kira-kira normal) atas semua set yang mungkin. Itu tidak menjamin batas atas pada kesalahan untuk setiap set input tertentu. Implementasi fungsi ini saat ini mengharus kan e berada dalam kisaran [0,0040625, 0,26000]. `SELECT approx_distinct(t.c, 0.2) FROM (VALUES 1, 2, 3, 4, 8) AS t(c)` Contoh hasil: 5

Fungsi	Tipe data keluaran	Deskripsi
kira-kira_persentil (x, persentase)	[sama seperti x]	Mengembalikan perkiraan persentil untuk semua nilai masukan x pada persentase yang diberikan. Nilai persentase harus antara nol dan satu dan harus konstan untuk semua baris input. `SELECT approx_percentile(t.c, 0.4) FROM (VALUES 1, 2, 3, 4) AS t(c)` Contoh hasil: 2
kira-kira_persentil (x, persentase)	array< [same as x] >	Mengembalikan perkiraan persentil untuk semua nilai masukan x pada masing-masing persentase yang ditentukan. Setiap elemen dari array persentase harus antara nol dan satu, dan array harus konstan untuk semua baris input. `SELECT approx_percentile(t.c, ARRAY[0.1, 0.8, 0.8]) FROM (VALUES 1, 2, 3, 4) AS t(c)` Contoh hasil: `[ 1,4,4 ]`

Fungsi	Tipe data keluaran	Deskripsi
kira-kira_persentil (x, w, persentase)	[sama seperti x]	Mengembalikan perkiraan persentil tertimbang untuk semua nilai masukan x menggunakan berat per item w pada persentase p. Bobot harus berupa nilai integer minimal satu. Ini secara efektif merupakan hitungan replikasi untuk nilai x dalam set persentil. Nilai p harus antara nol dan satu dan harus konstan untuk semua baris input. `SELECT approx_percentile(t.c, 1, 0.1) FROM (VALUES 1, 2, 3, 4) AS t(c)` Contoh hasil: 1

Fungsi	Tipe data keluaran	Deskripsi
kira-kira_persentil (x, w, persentase)	array< [same as x] >	Mengembalikan perkiraan persentil tertimbang untuk semua nilai masukan x menggunakan berat per item w pada masing-masing persentase yang diberikan ditentukan dalam array. Bobot harus berupa nilai integer minimal satu. Ini secara efektif merupakan hitungan replikasi untuk nilai x dalam set persentil. Setiap elemen array harus antara nol dan satu, dan array harus konstan untuk semua baris input. `SELECT approx_pe rcentile(t.c, 1, ARRAY[0.1, 0.8, 0.8]) FROM (VALUES 1, 2, 3, 4) AS t(c)` Contoh hasil: `[ 1,4,4 ]`

Fungsi	Tipe data keluaran	Deskripsi
kira-kira_persentil (x, w, persentase, akurasi)	[sama seperti x]	Mengembalikan perkiraan persentil tertimbang untuk semua nilai masukan x menggunakan berat per item w pada persentase p, dengan kesalahan peringkat akurasi maksimum. Bobot harus berupa nilai integer minimal satu. Ini secara efektif merupakan hitungan replikasi untuk nilai x dalam set persentil. Nilai p harus antara nol dan satu dan harus konstan untuk semua baris input. Akurasi harus bernilai lebih besar dari nol dan kurang dari satu, dan harus konstan untuk semua baris input. `SELECT approx_pe rcentile(t.c, 1, 0.1, 0.5) FROM (VALUES 1, 2, 3, 4) AS t(c)` Contoh hasil: 1

Fungsi	Tipe data keluaran	Deskripsi
corr (y, x)	double	Mengembalikan koefisien korelasi nilai masukan. `SELECT corr(t.c1, t.c2) FROM (VALUES ((1, 1)), ((2, 2)), ((3, 3)), ((4, 4))) AS t(c1, c2)` Contoh hasil: `1.0`
covar_pop (y, x)	double	Mengembalikan kovarians populasi dari nilai masukan. `SELECT covar_pop(t.c1, t.c2) FROM (VALUES ((1, 1)), ((2, 2)), ((3, 3)), ((4, 4))) AS t(c1, c2)` Contoh hasil: `1.25`
covar_samp (y, x)	double	Mengembalikan kovarians sampel dari nilai masukan. `SELECT covar_samp(t.c1, t.c2) FROM (VALUES ((1, 1)), ((2, 2)), ((3, 3)), ((4, 4))) AS t(c1, c2)` Contoh hasil: `1.6666666666666667`

Fungsi	Tipe data keluaran	Deskripsi
regr_intersep (y, x)	double	Mengembalikan intersep regresi linier nilai input. y adalah nilai dependen. x adalah nilai independen. `SELECT regr_intercept(t.c1, t.c2) FROM (VALUES ((1, 1)), ((2, 2)), ((3, 3)), ((4, 4))) AS t(c1, c2)` Contoh hasil: `0.0`
kemiringan regr_(y, x)	double	Mengembalikan kemiringan regresi linier nilai input. y adalah nilai dependen. x adalah nilai independen. `SELECT regr_slope(t.c1, t.c2) FROM (VALUES ((1, 1)), ((2, 2)), ((3, 3)), ((4, 4))) AS t(c1, c2)` Contoh hasil: `1.0`
kemiringan (x)	double	Mengembalikan kemiringan semua nilai masukan. `SELECT skewness(t.c1) FROM (VALUES 1, 2, 3, 4, 8) AS t(c1)` Contoh hasil: `0.8978957037987335`

Fungsi	Tipe data keluaran	Deskripsi
stddev_pop (x)	double	Mengembalikan standar deviasi populasi dari semua nilai masukan. `SELECT stddev_pop(t.c1) FROM (VALUES 1, 2, 3, 4, 8) AS t(c1)` Contoh hasil: `2.4166091947189146`
stddev_samp (x) stddev (x)	double	Mengembalikan standar deviasi sampel dari semua nilai masukan. `SELECT stddev_samp(t.c1) FROM (VALUES 1, 2, 3, 4, 8) AS t(c1)` Contoh hasil: `2.7018512172212592`
var_pop (x)	double	Mengembalikan varians populasi dari semua nilai input. `SELECT var_pop(t.c1) FROM (VALUES 1, 2, 3, 4, 8) AS t(c1)` Contoh hasil: `5.840000000000001`

Fungsi	Tipe data keluaran	Deskripsi
var_samp (x) varians (x)	double	Mengembalikan varians sampel dari semua nilai masukan. `SELECT var_samp(t.c1) FROM (VALUES 1, 2, 3, 4, 8) AS t(c1)` Contoh hasil: `7.300000000000001`

Fungsi jendela

Fungsi jendela melakukan perhitungan di seluruh baris hasil kueri. Mereka berjalan setelah HAVING klausa tetapi sebelum klausa ORDER BY. Memanggil fungsi jendela memerlukan sintaks khusus menggunakan OVER klausa untuk menentukan jendela. Sebuah jendela memiliki tiga komponen:

- Spesifikasi partisi, yang memisahkan baris input menjadi partisi yang berbeda. Ini analog dengan bagaimana klausa GROUP BY memisahkan baris ke dalam kelompok yang berbeda untuk fungsi agregat.
- Spesifikasi pemesanan, yang menentukan urutan baris input akan diproses oleh fungsi jendela.
- Bingkai jendela, yang menentukan jendela geser baris yang akan diproses oleh fungsi untuk baris tertentu. Jika frame tidak ditentukan, defaultnya RANGE UNBOUNDEDPRECEDING, yang sama dengan. RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW Bingkai ini berisi semua baris dari awal partisi hingga rekan terakhir dari baris saat ini.

Semua Fungsi Agregat dapat digunakan sebagai fungsi jendela dengan menambahkan OVER klausa. Fungsi agregat dihitung untuk setiap baris di atas baris dalam bingkai jendela baris saat ini. Selain fungsi agregat, Timestream untuk LiveAnalytics mendukung fungsi peringkat dan nilai berikut.

Fungsi	Tipe data keluaran	Deskripsi
cume_dist ()	bigint	Mengembalikan distribus i kumulatif nilai dalam

Fungsi	Tipe data keluaran	Deskripsi
		sekelompok nilai. Hasilnya adalah jumlah baris sebelumnya atau peer dengan baris di jendela urutan partisi jendela dibagi dengan jumlah total baris di partisi jendela. Dengan demikian, setiap nilai dasi dalam pemesanan akan mengevaluasi nilai distribusi yang sama.
dense_rank ()	bigint	Mengembalikan peringkat nilai dalam sekelompok nilai. Ini mirip dengan rank (), kecuali bahwa nilai dasi tidak menghasilkan celah dalam urutan.
ntil (n)	bigint	Membagi baris untuk setiap partisi jendela menjadi n ember mulai dari 1 hingga paling banyak n. Nilai bucket akan berbeda paling banyak 1. Jika jumlah baris di partisi tidak dibagi secara merata ke dalam jumlah ember, maka nilai sisanya didistrib usikan satu per ember, dimulai dengan ember pertama.

Fungsi	Tipe data keluaran	Deskripsi
percent_rank ()	double	Mengembalikan persentase peringkat nilai dalam kelompok nilai. Hasilnya adalah (r - 1)/(n - 1) di mana r adalah peringkat () dari baris dan n adalah jumlah total baris di partisi jendela.
peringkat ()	bigint	Mengembalikan peringkat nilai dalam sekelompok nilai. Peringkat adalah satu ditambah jumlah baris sebelum baris yang tidak sejajar dengan baris. Dengan demikian, nilai dasi dalam urutan akan menghasilkan celah dalam urutan. Peringkat dilakukan untuk setiap partisi jendela.
baris_number ()	bigint	Mengembalikan nomor unik dan berurutan untuk setiap baris, dimulai dengan satu, sesuai dengan urutan baris dalam partisi jendela.
nilai pertama_(x)	[sama seperti masukan]	Mengembalikan nilai pertama dari jendela. Fungsi ini tercakup pada bingkai jendela. Fungsi mengambil ekspresi atau target sebagai parameternya.

Fungsi	Tipe data keluaran	Deskripsi
nilai terakhir (x)	[sama seperti masukan]	Mengembalikan nilai terakhir dari jendela. Fungsi ini tercakup pada bingkai jendela. Fungsi mengambil ekspresi atau target sebagai parameter nya.
nth_value (x, offset)	[sama seperti masukan]	Mengembalikan nilai pada offset yang ditentukan dari awal jendela. Offset mulai dari 1. Offset dapat berupa ekspresi skalar apa pun. Jika offset adalah nol atau lebih besar dari jumlah nilai di jendela, null dikembalikan. Ini adalah kesalahan untuk offset menjadi nol atau negatif. Fungsi mengambil ekspresi atau target sebagai parameter pertamanya.
memimpin (x [, offset [, default_value]])	[sama seperti masukan]	Mengembalikan nilai pada baris offset setelah baris saat ini di jendela. Offset mulai dari 0, yang merupakan baris saat ini. Offset dapat berupa ekspresi skalar apa pun. Offset default adalah 1. Jika offset adalah null atau lebih besar dari jendela, default_v alue dikembalikan, atau jika tidak ditentukan null dikembali kan. Fungsi mengambil ekspresi atau target sebagai parameter pertamanya.

Fungsi	Tipe data keluaran	Deskripsi
lag (x [, offset [, default_v alue]])	[sama seperti masukan]	Mengembalikan nilai pada baris offset sebelum baris saat ini di jendela Offset mulai dari 0, yang merupakan baris saat ini. Offset dapat berupa ekspresi skalar apa pun. Offset default adalah 1. Jika offset adalah null atau lebih besar dari jendela, default_v alue dikembalikan, atau jika tidak ditentukan null dikembali kan. Fungsi mengambil ekspresi atau target sebagai parameter pertamanya.

Kueri Sampel

Bagian ini mencakup contoh kasus penggunaan Timestream untuk bahasa LiveAnalytics kueri.

Topik

- Pertanyaan sederhana
- Kueri dengan fungsi deret waktu
- Kueri dengan fungsi agregat

Pertanyaan sederhana

Berikut ini mendapatkan 10 titik data yang paling baru ditambahkan untuk sebuah tabel.

```
SELECT * FROM <database_name>.<table_name>
ORDER BY time DESC
LIMIT 10
```

Berikut ini mendapatkan 5 titik data tertua untuk ukuran tertentu.

```
SELECT * FROM <database_name>.<table_name>
WHERE measure_name = '<measure_name>'
ORDER BY time ASC
LIMIT 5
```

Berikut ini bekerja dengan stempel waktu granularitas nanodetik.

```
SELECT now() AS time_now
, now() - (INTERVAL '12' HOUR) AS twelve_hour_earlier -- Compatibility with ANSI SQL
, now() - 12h AS also_twelve_hour_earlier -- Convenient time interval literals
, ago(12h) AS twelve_hours_ago -- More convenience with time functionality
, bin(now(), 10m) AS time_binned -- Convenient time binning support
, ago(50ns) AS fifty_ns_ago -- Nanosecond support
, now() + (1h + 50ns) AS hour_fifty_ns_future
```

Nilai pengukuran untuk catatan multi-ukuran diidentifikasi dengan nama kolom. Nilai ukuran untuk catatan ukuran tunggal diidentifikasi olehmeasure_value::*<data_type>*, di mana *<data_type>* salah satu daridouble,, bigintboolean, atau varchar seperti yang dijelaskan dalamJenis data yang didukung. Untuk informasi selengkapnya tentang bagaimana nilai ukuran dimodelkan, lihat Tabel tunggal vs. beberapa tabel.

Berikut ini mengambil nilai untuk ukuran yang dipanggil speed dari catatan multi-ukuran dengan ameasure_name. IoTMulti-stats

```
SELECT speed FROM <database_name>.<table_name> where measure_name = 'IoTMulti-stats'
```

Berikut ini mengambil double nilai dari catatan ukuran tunggal dengan a measure_name dari. load

```
SELECT measure_value::double FROM <database_name>.<table_name> WHERE measure_name =
 'load'
```

Kueri dengan fungsi deret waktu

Topik

- Contoh dataset dan kueri

Contoh dataset dan kueri

Anda dapat menggunakan Timestream LiveAnalytics untuk memahami dan meningkatkan kinerja dan ketersediaan layanan dan aplikasi Anda. Di bawah ini adalah contoh tabel dan contoh query berjalan pada tabel itu.

Tabel `ec2_metrics` menyimpan data telemetri, seperti CPU pemanfaatan dan metrik lainnya dari instance. EC2 Anda dapat melihat tabel di bawah ini.

Waktu	region	az	Nama host	ukuran_nama	ukuran_nilai: :ganda	ukuran_nilai: :bigint
2019-12-04 19:00:00.000000000	us-east-1	us-east-1a	frontend01	pemanfaatan cpu_	35.1	null
2019-12-04 19:00:00.000000000	us-east-1	us-east-1a	frontend01	memory_utilization	55,3	null
2019-12-04 19:00:00.000000000	us-east-1	us-east-1a	frontend01	network_bytes_in	null	1.500
2019-12-04 19:00:00.000000000	us-east-1	us-east-1a	frontend01	network_bytes_out	null	6,700
2019-12-04 19:00:00.000000000	us-east-1	us-east-1b	frontend02	pemanfaatan cpu_	38,5	null
2019-12-04 19:00:00.	us-east-1	us-east-1b	frontend02	memory_utilization	58.4	null

Waktu	region	az	Nama host	ukuran_nama	ukuran_nilai: :ganda	ukuran_nilai: :bigint
000 000000						
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1b	frontend02	network_bytes_in	null	23.000
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1b	frontend02	network_bytes_out	null	12.000
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1c	frontend03	pemanfaatan cpu_	45,0	null
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1c	frontend03	memory_utilization	65.8	null
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1c	frontend03	network_bytes_in	null	15.000
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1c	frontend03	network_bytes_out	null	836.000

Waktu	region	az	Nama host	ukuran_nama	ukuran_nilai: :ganda	ukuran_nilai: :bigint
2019-12-04 19:00:05.000000000	us-east-1	us-east-1a	frontend01	pemanfaatan cpu_	55,2	null
2019-12-04 19:00:05.000000000	us-east-1	us-east-1a	frontend01	memory_utilization	75.0	null
2019-12-04 19:00:05.000000000	us-east-1	us-east-1a	frontend01	network_bytes_in	null	1,245
2019-12-04 19:00:05.000000000	us-east-1	us-east-1a	frontend01	network_bytes_out	null	68,432
2019-12-04 19:00:08.000000000	us-east-1	us-east-1b	frontend02	pemanfaatan cpu_	65.6	null
2019-12-04 19:00:08.000000000	us-east-1	us-east-1b	frontend02	memory_utilization	85.3	null
2019-12-04 19:00:08.000000000	us-east-1	us-east-1b	frontend02	network_bytes_in	null	1,245

Waktu	region	az	Nama host	ukuran_nama	ukuran_nilai: :ganda	ukuran_nilai: :bigint
2019-12-04 19:00:08.000000000	us-east-1	us-east-1b	frontend02	network_bytes_out	null	68,432
2019-12-04 19:00:20.000000000	us-east-1	us-east-1c	frontend03	pemanfaatan cpu_	12.1	null
2019-12-04 19:00:20.000000000	us-east-1	us-east-1c	frontend03	memory_utilization	32.0	null
2019-12-04 19:00:20.000000000	us-east-1	us-east-1c	frontend03	network_bytes_in	null	1.400
2019-12-04 19:00:20.000000000	us-east-1	us-east-1c	frontend03	network_bytes_out	null	345
2019-12-04 19:00:10.000000000	us-east-1	us-east-1a	frontend01	pemanfaatan cpu_	15.3	null
2019-12-04 19:00:10.000000000	us-east-1	us-east-1a	frontend01	memory_utilization	35.4	null

Waktu	region	az	Nama host	ukuran_nama	ukuran_nilai: :ganda	ukuran_nilai: :bigint
2019-12-04 19:00:10.000000000	us-east-1	us-east-1a	frontend01	network_bytes_in	null	23
2019-12-04 19:00:10.000000000	us-east-1	us-east-1a	frontend01	network_bytes_out	null	0
2019-12-04 19:00:16.000000000	us-east-1	us-east-1b	frontend02	pemanfaatan cpu_	44,0	null
2019-12-04 19:00:16.000000000	us-east-1	us-east-1b	frontend02	memory_utilization	64.2	null
2019-12-04 19:00:16.000000000	us-east-1	us-east-1b	frontend02	network_bytes_in	null	1,450
2019-12-04 19:00:16.000000000	us-east-1	us-east-1b	frontend02	network_bytes_out	null	200
2019-12-04 19:00:40.000000000	us-east-1	us-east-1c	frontend03	pemanfaatan cpu_	66.4	null

Waktu	region	az	Nama host	ukuran_nama	ukuran_nilai: :ganda	ukuran_nilai: :bigint
2019-12-04 19:00:40.000 000000	us-east-1	us-east-1c	frontend03	memory_utilization	86.3	null
2019-12-04 19:00:40.000 000000	us-east-1	us-east-1c	frontend03	network_bytes_in	null	300
2019-12-04 19:00:40.000 000000	us-east-1	us-east-1c	frontend03	network_bytes_out	null	423

Temukan CPU pemanfaatan rata-rata, p90, p95, dan p99 untuk EC2 host tertentu selama 2 jam terakhir:

```
SELECT region, az, hostname, BIN(time, 15s) AS binned_timestamp,
    ROUND(AVG(measure_value::double), 2) AS avg_cpu_utilization,
    ROUND(APPROX_PERCENTILE(measure_value::double, 0.9), 2) AS p90_cpu_utilization,
    ROUND(APPROX_PERCENTILE(measure_value::double, 0.95), 2) AS p95_cpu_utilization,
    ROUND(APPROX_PERCENTILE(measure_value::double, 0.99), 2) AS p99_cpu_utilization
FROM "sampleDB".DevOps
WHERE measure_name = 'cpu_utilization'
    AND hostname = 'host-Hovjv'
    AND time > ago(2h)
GROUP BY region, hostname, az, BIN(time, 15s)
ORDER BY binned_timestamp ASC
```

Identifikasi EC2 host dengan CPU pemanfaatan yang lebih tinggi sebesar 10% atau lebih dibandingkan dengan CPU pemanfaatan rata-rata seluruh armada selama 2 jam terakhir:

```
WITH avg_fleet_utilization AS (
    SELECT COUNT(DISTINCT hostname) AS total_host_count, AVG(measure_value::double) AS
 fleet_avg_cpu_utilization
```

```
    FROM "sampleDB".DevOps
    WHERE measure_name = 'cpu_utilization'
        AND time > ago(2h)
), avg_per_host_cpu AS (
    SELECT region, az, hostname, AVG(measure_value::double) AS avg_cpu_utilization
    FROM "sampleDB".DevOps
    WHERE measure_name = 'cpu_utilization'
        AND time > ago(2h)
    GROUP BY region, az, hostname
)
SELECT region, az, hostname, avg_cpu_utilization, fleet_avg_cpu_utilization
FROM avg_fleet_utilization, avg_per_host_cpu
WHERE avg_cpu_utilization > 1.1 * fleet_avg_cpu_utilization
ORDER BY avg_cpu_utilization DESC
```

Temukan CPU pemanfaatan rata-rata yang di-binned pada interval 30 detik untuk EC2 host tertentu selama 2 jam terakhir:

```
SELECT BIN(time, 30s) AS binned_timestamp, ROUND(AVG(measure_value::double), 2) AS
 avg_cpu_utilization
FROM "sampleDB".DevOps
WHERE measure_name = 'cpu_utilization'
    AND hostname = 'host-Hovjv'
    AND time > ago(2h)
GROUP BY hostname, BIN(time, 30s)
ORDER BY binned_timestamp ASC
```

Temukan CPU pemanfaatan rata-rata yang di-binned pada interval 30 detik untuk EC2 host tertentu selama 2 jam terakhir, mengisi nilai yang hilang menggunakan interpolasi linier:

```
WITH binned_timeseries AS (
    SELECT hostname, BIN(time, 30s) AS binned_timestamp,
 ROUND(AVG(measure_value::double), 2) AS avg_cpu_utilization
    FROM "sampleDB".DevOps
    WHERE measure_name = 'cpu_utilization'
        AND hostname = 'host-Hovjv'
        AND time > ago(2h)
    GROUP BY hostname, BIN(time, 30s)
), interpolated_timeseries AS (
    SELECT hostname,
        INTERPOLATE_LINEAR(
            CREATE_TIME_SERIES(binned_timestamp, avg_cpu_utilization),
```

```
                SEQUENCE(min(binned_timestamp), max(binned_timestamp), 15s)) AS
 interpolated_avg_cpu_utilization
    FROM binned_timeseries
    GROUP BY hostname
)
SELECT time, ROUND(value, 2) AS interpolated_cpu
FROM interpolated_timeseries
CROSS JOIN UNNEST(interpolated_avg_cpu_utilization)
```

Temukan CPU pemanfaatan rata-rata yang di-binned pada interval 30 detik untuk EC2 host tertentu selama 2 jam terakhir, mengisi nilai yang hilang menggunakan interpolasi berdasarkan pengamatan terakhir yang dilakukan ke depan:

```
WITH binned_timeseries AS (
    SELECT hostname, BIN(time, 30s) AS binned_timestamp,
 ROUND(AVG(measure_value::double), 2) AS avg_cpu_utilization
    FROM "sampleDB".DevOps
    WHERE measure_name = 'cpu_utilization'
        AND hostname = 'host-Hovjv'
        AND time > ago(2h)
    GROUP BY hostname, BIN(time, 30s)
), interpolated_timeseries AS (
    SELECT hostname,
        INTERPOLATE_LOCF(
            CREATE_TIME_SERIES(binned_timestamp, avg_cpu_utilization),
                SEQUENCE(min(binned_timestamp), max(binned_timestamp), 15s)) AS
 interpolated_avg_cpu_utilization
    FROM binned_timeseries
    GROUP BY hostname
)
SELECT time, ROUND(value, 2) AS interpolated_cpu
FROM interpolated_timeseries
CROSS JOIN UNNEST(interpolated_avg_cpu_utilization)
```

Kueri dengan fungsi agregat

Di bawah ini adalah contoh contoh skenario IoT kumpulan data untuk menggambarkan query dengan fungsi agregat.

Topik

- Contoh data

- Kueri contoh

Contoh data

Timestream memungkinkan Anda menyimpan dan menganalisis data sensor IoT seperti lokasi, konsumsi bahan bakar, kecepatan, dan kapasitas muat satu atau lebih armada truk untuk memungkinkan manajemen armada yang efektif. Di bawah ini adalah skema dan beberapa data tabel iot_trucks yang menyimpan telemetri seperti lokasi, konsumsi bahan bakar, kecepatan, dan kapasitas muat truk.

Waktu	truck_id	Membua	Model	Armada	kapasitas bahan bakar	load_capacity	ukuran_nama	ukuran_nilai: :ganc	ukuran_nilai: :varchar
2019-12-4 19:00:00.000000000	1234567	GMC	Astro	Alfa	100	500	bahan bakar_mbaca	65.2	null
2019-12-4 19:00:00.000000000	1234567	GMC	Astro	Alfa	100	500	muat	400,0	null
2019-12-4 19:00:00.000000000	1234567	GMC	Astro	Alfa	100	500	kecepata	90.2	null
2019-12-4 19:00:00.000000000	1234567	GMC	Astro	Alfa	100	500	lokasi	null	47.6062 N, 122.3321 W

Waktu	truck_id	Membua	Model	Armada	kapasitas bahan bakar	load_cap city	ukuran_r ma	ukuran_r lai: :ganc	ukuran_ni lai: :varchar
2019-12-4 19:00:00 000 000000	1234567	Kenworth	W900	Alfa	150	1000	bahan bakar_m baca	10.1	null
2019-12-4 19:00:00 000 000000	1234567	Kenworth	W900	Alfa	150	1000	muat	950,3	null
2019-12-4 19:00:00 000 000000	1234567	Kenworth	W900	Alfa	150	1000	kecepata	50,8	null
2019-12-4 19:00:00 000 000000	1234567	Kenworth	W900	Alfa	150	1000	lokasi	null	40,7128 derajat N, 74,0060 derajat W

Kueri contoh

Dapatkan daftar semua atribut sensor dan nilai yang dipantau untuk setiap truk di armada.

```
SELECT
    truck_id,
    fleet,
    fuel_capacity,
    model,
```

```
    load_capacity,
    make,
    measure_name
FROM "sampleDB".IoT
GROUP BY truck_id, fleet, fuel_capacity, model, load_capacity, make, measure_name
```

Dapatkan pembacaan bahan bakar terbaru dari setiap truk di armada dalam 24 jam terakhir.

```
WITH latest_recorded_time AS (
    SELECT
        truck_id,
        max(time) as latest_time
    FROM "sampleDB".IoT
    WHERE measure_name = 'fuel-reading'
    AND time >= ago(24h)
    GROUP BY truck_id
)
SELECT
    b.truck_id,
    b.fleet,
    b.make,
    b.model,
    b.time,
    b.measure_value::double as last_reported_fuel_reading
FROM
latest_recorded_time a INNER JOIN "sampleDB".IoT b
ON a.truck_id = b.truck_id AND b.time = a.latest_time
WHERE b.measure_name = 'fuel-reading'
AND b.time > ago(24h)
ORDER BY b.truck_id
```

Identifikasi truk yang menggunakan bahan bakar rendah (kurang dari 10%) dalam 48 jam terakhir:

```
WITH low_fuel_trucks AS (
    SELECT time, truck_id, fleet, make, model, (measure_value::double/
cast(fuel_capacity as double)*100) AS fuel_pct
    FROM "sampleDB".IoT
    WHERE time >= ago(48h)
    AND (measure_value::double/cast(fuel_capacity as double)*100) < 10
    AND measure_name = 'fuel-reading'
),
other_trucks AS (
```

```
SELECT time, truck_id, (measure_value::double/cast(fuel_capacity as double)*100) as
 remaining_fuel
    FROM "sampleDB".IoT
    WHERE time >= ago(48h)
    AND truck_id IN (SELECT truck_id FROM low_fuel_trucks)
    AND (measure_value::double/cast(fuel_capacity as double)*100) >= 10
    AND measure_name = 'fuel-reading'
),
trucks_that_refuelled AS (
    SELECT a.truck_id
    FROM low_fuel_trucks a JOIN other_trucks b
    ON a.truck_id = b.truck_id AND b.time >= a.time
)
SELECT DISTINCT truck_id, fleet, make, model, fuel_pct
FROM low_fuel_trucks
WHERE truck_id NOT IN (
    SELECT truck_id FROM trucks_that_refuelled
)
```

Temukan beban rata-rata dan kecepatan maksimal untuk setiap truk selama seminggu terakhir:

```
SELECT
    bin(time, 1d) as binned_time,
    fleet,
    truck_id,
    make,
    model,
    AVG(
        CASE WHEN measure_name = 'load' THEN measure_value::double ELSE NULL END
    ) AS avg_load_tons,
    MAX(
        CASE WHEN measure_name = 'speed' THEN measure_value::double ELSE NULL END
    ) AS max_speed_mph
FROM "sampleDB".IoT
WHERE time >= ago(7d)
AND measure_name IN ('load', 'speed')
GROUP BY fleet, truck_id, make, model, bin(time, 1d)
ORDER BY truck_id
```

Dapatkan efisiensi beban untuk setiap truk selama seminggu terakhir:

```
WITH average_load_per_truck AS (
    SELECT
```

```
        truck_id,
        avg(measure_value::double)  AS avg_load
    FROM "sampleDB".IoT
    WHERE measure_name = 'load'
    AND time >= ago(7d)
    GROUP BY truck_id, fleet, load_capacity, make, model
),
truck_load_efficiency AS (
    SELECT
        a.truck_id,
        fleet,
        load_capacity,
        make,
        model,
        avg_load,
        measure_value::double,
        time,
        (measure_value::double*100)/avg_load as load_efficiency -- ,
 approx_percentile(avg_load_pct, DOUBLE '0.9')
    FROM "sampleDB".IoT a JOIN average_load_per_truck b
    ON a.truck_id = b.truck_id
    WHERE a.measure_name = 'load'
)
SELECT
    truck_id,
    time,
    load_efficiency
FROM truck_load_efficiency
ORDER BY truck_id, time
```

APIreferensi

Bagian ini berisi dokumentasi API Referensi untuk Amazon Timestream.

Timestream memiliki duaAPIs: Query dan Write.

- Tulis API memungkinkan Anda untuk melakukan operasi seperti pembuatan tabel, penandaan sumber daya, dan penulisan catatan ke Timestream.
- Query API memungkinkan Anda untuk melakukan operasi query.

> **Note**
>
> Keduanya APIs termasuk DescribeEndpoints aksi. Untuk Query dan Write, DescribeEndpoints tindakannya identik.

Anda dapat membaca lebih lanjut tentang masing-masing API di bawah ini, bersama dengan tipe data, kesalahan umum, dan parameter.

> **Note**
>
> Untuk kode kesalahan yang umum untuk semua AWS layanan, lihat bagian AWS Support.

Topik

Tindakan

Tindakan berikut didukung oleh Amazon Timestream Write:

- CreateBatchLoadTask
- CreateDatabase
- CreateTable
- DeleteDatabase
- DeleteTable
- DescribeBatchLoadTask
- DescribeDatabase
- DescribeEndpoints
- DescribeTable
- ListBatchLoadTasks

- ListDatabases
- ListTables
- ListTagsForResource
- ResumeBatchLoadTask
- TagResource
- UntagResource
- UpdateDatabase
- UpdateTable
- WriteRecords

Tindakan berikut didukung oleh Amazon Timestream Query:

- CancelQuery
- CreateScheduledQuery
- DeleteScheduledQuery
- DescribeAccountSettings
- DescribeEndpoints
- DescribeScheduledQuery
- ExecuteScheduledQuery
- ListScheduledQueries
- ListTagsForResource
- PrepareQuery
- Query
- TagResource
- UntagResource
- UpdateAccountSettings
- UpdateScheduledQuery

Amazon Timestream Menulis

Tindakan berikut didukung oleh Amazon Timestream Write:

- CreateBatchLoadTask
- CreateDatabase
- CreateTable
- DeleteDatabase
- DeleteTable
- DescribeBatchLoadTask
- DescribeDatabase
- DescribeEndpoints
- DescribeTable
- ListBatchLoadTasks
- ListDatabases
- ListTables
- ListTagsForResource
- ResumeBatchLoadTask
- TagResource
- UntagResource
- UpdateDatabase
- UpdateTable
- WriteRecords

CreateBatchLoadTask

Layanan: Amazon Timestream Write

Membuat tugas pemuatan batch Timestream baru. Tugas pemuatan batch memproses data dari CSV sumber di lokasi S3 dan menulis ke tabel Timestream. Pemetaan dari sumber ke target didefinisikan dalam tugas pemuatan batch. Kesalahan dan peristiwa ditulis ke laporan di lokasi S3. Untuk laporan, jika AWS KMS kunci tidak ditentukan, laporan akan dienkripsi dengan kunci terkelola S3 saat opsi SSE_S3 tersebut. Jika tidak, kesalahan akan dilemparkan. Untuk informasi selengkapnya, lihat kunci AWS terkelola. Service quotas berlaku. Untuk detailnya, lihat contoh kode.

Sintaksis Permintaan

```
{
   "ClientToken": "string",
   "DataModelConfiguration": {
      "DataModel": {
         "DimensionMappings": [
            {
               "DestinationColumn": "string",
               "SourceColumn": "string"
            }
         ],
         "MeasureNameColumn": "string",
         "MixedMeasureMappings": [
            {
               "MeasureName": "string",
               "MeasureValueType": "string",
               "MultiMeasureAttributeMappings": [
                  {
                     "MeasureValueType": "string",
                     "SourceColumn": "string",
                     "TargetMultiMeasureAttributeName": "string"
                  }
               ],
               "SourceColumn": "string",
               "TargetMeasureName": "string"
            }
         ],
         "MultiMeasureMappings": {
            "MultiMeasureAttributeMappings": [
               {
                  "MeasureValueType": "string",
                  "SourceColumn": "string",
```

```
                    "TargetMultiMeasureAttributeName": "string"
                }
            ],
            "TargetMultiMeasureName": "string"
        },
        "TimeColumn": "string",
        "TimeUnit": "string"
    },
    "DataModelS3Configuration": {
        "BucketName": "string",
        "ObjectKey": "string"
    }
},
"DataSourceConfiguration": {
    "CsvConfiguration": {
        "ColumnSeparator": "string",
        "EscapeChar": "string",
        "NullValue": "string",
        "QuoteChar": "string",
        "TrimWhiteSpace": boolean
    },
    "DataFormat": "string",
    "DataSourceS3Configuration": {
        "BucketName": "string",
        "ObjectKeyPrefix": "string"
    }
},
"RecordVersion": number,
"ReportConfiguration": {
    "ReportS3Configuration": {
        "BucketName": "string",
        "EncryptionOption": "string",
        "KmsKeyId": "string",
        "ObjectKeyPrefix": "string"
    }
},
"TargetDatabaseName": "string",
"TargetTableName": "string"
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

ClientToken

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Wajib: Tidak

DataModelConfiguration

Tipe: Objek DataModelConfiguration

Wajib: Tidak

DataSourceConfiguration

Mendefinisikan detail konfigurasi tentang sumber data untuk tugas pemuatan batch.

Tipe: Objek DataSourceConfiguration

Wajib: Ya

RecordVersion

Tipe: Panjang

Wajib: Tidak

ReportConfiguration

Laporkan konfigurasi untuk tugas pemuatan batch. Ini berisi rincian tentang di mana laporan kesalahan disimpan.

Tipe: Objek ReportConfiguration

Wajib: Ya

TargetDatabaseName

Target database Timestream untuk tugas pemuatan batch.

Tipe: String

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

TargetTableName

Target tabel Timestream untuk tugas pemuatan batch.

Tipe: String

Pola: `[a-zA-Z0-9_.-]+`

Diperlukan: Ya

Sintaksis Respons

```
{
   "TaskId": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Data berikut dikembalikan dalam JSON format oleh layanan.

TaskId

ID tugas pemuatan batch.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum sebesar 32.

Pola: `[A-Z0-9]+`

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

ConflictException

Timestream tidak dapat memproses permintaan ini karena berisi sumber daya yang sudah ada.

HTTPKode Status: 400

InternalServerException

Timestream tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 500

InvalidEndpointException

Titik akhir yang diminta tidak valid.

HTTPKode Status: 400

ResourceNotFoundException

Operasi mencoba mengakses sumber daya yang tidak ada. Sumber daya mungkin tidak ditentukan dengan benar, atau statusnya mungkin tidakACTIVE.

HTTPKode Status: 400

ServiceQuotaExceededException

Kuota instance sumber daya terlampaui untuk akun ini.

HTTPKode Status: 400

ThrottlingException

Terlalu banyak permintaan yang dibuat oleh pengguna dan mereka melebihi kuota layanan. Permintaan itu dibatasi.

HTTPKode Status: 400

ValidationException

Permintaan yang tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKpara Ruby V3

CreateDatabase

Layanan: Amazon Timestream Write

Menciptakan basis data Timestream baru. Jika AWS KMS kunci tidak ditentukan, database akan dienkripsi dengan kunci terkelola Timestream yang terletak di AWS KMS akun Anda. Untuk informasi selengkapnya, lihat kunci AWS terkelola. Service quotas berlaku. Untuk detailnya, lihat contoh kode.

Sintaksis Permintaan

```
{
   "DatabaseName": "string",
   "KmsKeyId": "string",
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

DatabaseName

Nama basis data Timestream.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 256.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

KmsKeyId

AWS KMS Kunci untuk database. Jika AWS KMS kunci tidak ditentukan, database akan dienkripsi dengan kunci terkelola Timestream yang terletak di AWS KMS akun Anda. Untuk informasi selengkapnya, lihat kunci AWS terkelola.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Tidak

Tags

Daftar pasangan kunci-nilai untuk memberi label pada tabel.

Tipe: Array objek Tag

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 200 item.

Wajib: Tidak

Sintaksis Respons

```
{
   "Database": {
      "Arn": "string",
      "CreationTime": number,
      "DatabaseName": "string",
      "KmsKeyId": "string",
      "LastUpdatedTime": number,
      "TableCount": number
   }
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Data berikut dikembalikan dalam JSON format oleh layanan.

Database

Database Timestream yang baru dibuat.

Tipe: Objek Database

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

ConflictException

Timestream tidak dapat memproses permintaan ini karena berisi sumber daya yang sudah ada.

HTTPKode Status: 400

InternalServerException

Timestream tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 500

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

ServiceQuotaExceededException

Kuota instans sumber daya terlampaui untuk akun ini.

HTTPKode Status: 400

ThrottlingException

Terlalu banyak permintaan yang dibuat oleh pengguna dan mereka melebihi kuota layanan. Permintaan itu dibatasi.

HTTPKode Status: 400

ValidationException

Permintaan yang tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKpara Ruby V3

CreateTable

Layanan: Amazon Timestream Write

Menambahkan tabel baru ke database yang ada di akun Anda. Dalam sebuah Akun AWS, nama tabel harus setidaknya unik dalam setiap Wilayah jika mereka berada dalam database yang sama. Anda mungkin memiliki nama tabel yang identik di Wilayah yang sama jika tabel berada dalam database terpisah. Saat membuat tabel, Anda harus menentukan nama tabel, nama basis data, dan properti retensi. Service quotas berlaku. Lihat sampel kode untuk detail.

Sintaksis Permintaan

```
{
   "DatabaseName": "string",
   "MagneticStoreWriteProperties": {
      "EnableMagneticStoreWrites": boolean,
      "MagneticStoreRejectedDataLocation": {
         "S3Configuration": {
            "BucketName": "string",
            "EncryptionOption": "string",
            "KmsKeyId": "string",
            "ObjectKeyPrefix": "string"
         }
      }
   },
   "RetentionProperties": {
      "MagneticStoreRetentionPeriodInDays": number,
      "MemoryStoreRetentionPeriodInHours": number
   },
   "Schema": {
      "CompositePartitionKey": [
         {
            "EnforcementInRecord": "string",
            "Name": "string",
            "Type": "string"
         }
      ]
   },
   "TableName": "string",
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
```

```
    ]
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

DatabaseName

Nama basis data Timestream.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 256.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

MagneticStoreWriteProperties

Berisi properti untuk diatur di atas meja saat mengaktifkan penulisan penyimpanan magnetik.

Tipe: Objek MagneticStoreWriteProperties

Wajib: Tidak

RetentionProperties

Durasi data deret waktu Anda harus disimpan di penyimpanan memori dan penyimpanan magnetik.

Tipe: Objek RetentionProperties

Wajib: Tidak

Schema

Skema tabel.

Tipe: Objek Schema

Wajib: Tidak

TableName

Nama tabel Timestream.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 256.

Pola: `[a-zA-Z0-9_.-]+`

Wajib: Ya

Tags

Daftar pasangan kunci-nilai untuk memberi label pada tabel.

Tipe: Array objek Tag

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 200 item.

Wajib: Tidak

Sintaksis Respons

```
{
   "Table": {
      "Arn": "string",
      "CreationTime": number,
      "DatabaseName": "string",
      "LastUpdatedTime": number,
      "MagneticStoreWriteProperties": {
         "EnableMagneticStoreWrites": boolean,
         "MagneticStoreRejectedDataLocation": {
            "S3Configuration": {
               "BucketName": "string",
               "EncryptionOption": "string",
               "KmsKeyId": "string",
               "ObjectKeyPrefix": "string"
            }
         }
      },
      "RetentionProperties": {
         "MagneticStoreRetentionPeriodInDays": number,
```

```
            "MemoryStoreRetentionPeriodInHours": number
         },
         "Schema": {
            "CompositePartitionKey": [
               {
                  "EnforcementInRecord": "string",
                  "Name": "string",
                  "Type": "string"
               }
            ]
         },
         "TableName": "string",
         "TableStatus": "string"
      }
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Data berikut dikembalikan dalam JSON format oleh layanan.

Table

Tabel Timestream yang baru dibuat.

Tipe: Objek Table

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

ConflictException

Timestream tidak dapat memproses permintaan ini karena berisi sumber daya yang sudah ada.

HTTPKode Status: 400

InternalServerException

Timestream tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 500

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

ResourceNotFoundException

Operasi mencoba mengakses sumber daya yang tidak ada. Sumber daya mungkin tidak ditentukan dengan benar, atau statusnya mungkin tidakACTIVE.

HTTPKode Status: 400

ServiceQuotaExceededException

Kuota instans sumber daya terlampaui untuk akun ini.

HTTPKode Status: 400

ThrottlingException

Terlalu banyak permintaan yang dibuat oleh pengguna dan mereka melebihi kuota layanan. Permintaan itu dibatasi.

HTTPKode Status: 400

ValidationException

Permintaan yang tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKpara Ruby V3

DeleteDatabase
Layanan: Amazon Timestream Write

Menghapus database Timestream yang diberikan. Ini adalah operasi yang tidak dapat diubah. Setelah database dihapus, data deret waktu dari tabelnya tidak dapat dipulihkan.

> **Note**
>
> Semua tabel dalam database harus dihapus terlebih dahulu, atau ValidationException kesalahan akan dilemparkan.
> Karena sifat percobaan ulang yang didistribusikan, operasi dapat mengembalikan keberhasilan atau a ResourceNotFoundException. Klien harus menganggapnya setara.

Lihat sampel kode untuk detail.

Sintaksis Permintaan

```
{
   "DatabaseName": "string"
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

DatabaseName

Nama database Timestream yang akan dihapus.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 256.

Wajib: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan HTTP tubuh kosong.

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

InternalServerException

Timestream tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 500

InvalidEndpointException

Titik akhir yang diminta tidak valid.

HTTPKode Status: 400

ResourceNotFoundException

Operasi mencoba mengakses sumber daya yang tidak ada. Sumber daya mungkin tidak ditentukan dengan benar, atau statusnya mungkin tidakACTIVE.

HTTPKode Status: 400

ThrottlingException

Terlalu banyak permintaan yang dibuat oleh pengguna dan mereka melebihi kuota layanan. Permintaan itu dibatasi.

HTTPKode Status: 400

ValidationException

Permintaan yang tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKuntuk Ruby V3

DeleteTable
Layanan: Amazon Timestream Write

Menghapus tabel Timestream yang diberikan. Ini adalah operasi yang tidak dapat diubah. Setelah tabel database Timestream dihapus, data deret waktu yang disimpan dalam tabel tidak dapat dipulihkan.

> **Note**
>
> Karena sifat percobaan ulang yang didistribusikan, operasi dapat mengembalikan keberhasilan atau a ResourceNotFoundException. Klien harus menganggap mereka setara.

Lihat sampel kode untuk detail.

Sintaksis Permintaan

```
{
   "DatabaseName": "string",
   "TableName": "string"
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

DatabaseName

Nama database tempat database Timestream akan dihapus.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 256.

Wajib: Ya

TableName

Nama tabel Timestream yang akan dihapus.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 256.

Wajib: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan HTTP tubuh kosong.

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

InternalServerException

Timestream tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 500

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

ResourceNotFoundException

Operasi mencoba mengakses sumber daya yang tidak ada. Sumber daya mungkin tidak ditentukan dengan benar, atau statusnya mungkin tidakACTIVE.

HTTPKode Status: 400

ThrottlingException

Terlalu banyak permintaan yang dibuat oleh pengguna dan mereka melebihi kuota layanan. Permintaan itu dibatasi.

HTTPKode Status: 400

ValidationException

Permintaan yang tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKuntuk Ruby V3

DescribeBatchLoadTask
Layanan: Amazon Timestream Write

Mengembalikan informasi tentang tugas pemuatan batch, termasuk konfigurasi, pemetaan, kemajuan, dan detail lainnya. Service quotas berlaku. Lihat sampel kode untuk detail.

Sintaksis Permintaan

```
{
   "TaskId": "string"
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

TaskId

ID tugas pemuatan batch.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum sebesar 32.

Pola: `[A-Z0-9]+`

Diperlukan: Ya

Sintaksis Respons

```
{
   "BatchLoadTaskDescription": {
      "CreationTime": number,
      "DataModelConfiguration": {
         "DataModel": {
            "DimensionMappings": [
               {
                  "DestinationColumn": "string",
                  "SourceColumn": "string"
               }
            ],
```

```
            "MeasureNameColumn": "string",
            "MixedMeasureMappings": [
               {
                  "MeasureName": "string",
                  "MeasureValueType": "string",
                  "MultiMeasureAttributeMappings": [
                     {
                        "MeasureValueType": "string",
                        "SourceColumn": "string",
                        "TargetMultiMeasureAttributeName": "string"
                     }
                  ],
                  "SourceColumn": "string",
                  "TargetMeasureName": "string"
               }
            ],
            "MultiMeasureMappings": {
               "MultiMeasureAttributeMappings": [
                  {
                     "MeasureValueType": "string",
                     "SourceColumn": "string",
                     "TargetMultiMeasureAttributeName": "string"
                  }
               ],
               "TargetMultiMeasureName": "string"
            },
            "TimeColumn": "string",
            "TimeUnit": "string"
         },
         "DataModelS3Configuration": {
            "BucketName": "string",
            "ObjectKey": "string"
         }
      },
      "DataSourceConfiguration": {
         "CsvConfiguration": {
            "ColumnSeparator": "string",
            "EscapeChar": "string",
            "NullValue": "string",
            "QuoteChar": "string",
            "TrimWhiteSpace": boolean
         },
         "DataFormat": "string",
         "DataSourceS3Configuration": {
```

```
            "BucketName": "string",
            "ObjectKeyPrefix": "string"
         }
      },
      "ErrorMessage": "string",
      "LastUpdatedTime": number,
      "ProgressReport": {
         "BytesMetered": number,
         "FileFailures": number,
         "ParseFailures": number,
         "RecordIngestionFailures": number,
         "RecordsIngested": number,
         "RecordsProcessed": number
      },
      "RecordVersion": number,
      "ReportConfiguration": {
         "ReportS3Configuration": {
            "BucketName": "string",
            "EncryptionOption": "string",
            "KmsKeyId": "string",
            "ObjectKeyPrefix": "string"
         }
      },
      "ResumableUntil": number,
      "TargetDatabaseName": "string",
      "TargetTableName": "string",
      "TaskId": "string",
      "TaskStatus": "string"
   }
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Data berikut dikembalikan dalam JSON format oleh layanan.

BatchLoadTaskDescription

Deskripsi tugas pemuatan batch.

Tipe: Objek BatchLoadTaskDescription

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

InternalServerException

Timestream tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 500

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

ResourceNotFoundException

Operasi mencoba mengakses sumber daya yang tidak ada. Sumber daya mungkin tidak ditentukan dengan benar, atau statusnya mungkin tidakACTIVE.

HTTPKode Status: 400

ThrottlingException

Terlalu banyak permintaan yang dibuat oleh pengguna dan mereka melebihi kuota layanan. Permintaan itu dibatasi.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++

- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKuntuk Ruby V3

DescribeDatabase
Layanan: Amazon Timestream Write

Mengembalikan informasi tentang database, termasuk nama database, waktu database dibuat, dan jumlah total tabel yang ditemukan dalam database. Service quotas berlaku. Lihat sampel kode untuk detail.

Sintaksis Permintaan

```
{
   "DatabaseName": "string"
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

DatabaseName

Nama basis data Timestream.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 256.

Wajib: Ya

Sintaksis Respons

```
{
   "Database": {
      "Arn": "string",
      "CreationTime": number,
      "DatabaseName": "string",
      "KmsKeyId": "string",
      "LastUpdatedTime": number,
      "TableCount": number
   }
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Data berikut dikembalikan dalam JSON format oleh layanan.

[Database](#)

Nama tabel Timestream.

Tipe: Objek [Database](#)

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat [Kesalahan Umum](#).

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

InternalServerException

Timestream tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 500

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

ResourceNotFoundException

Operasi mencoba mengakses sumber daya yang tidak ada. Sumber daya mungkin tidak ditentukan dengan benar, atau statusnya mungkin tidakACTIVE.

HTTPKode Status: 400

ThrottlingException

Terlalu banyak permintaan yang dibuat oleh pengguna dan mereka melebihi kuota layanan. Permintaan itu dibatasi.

HTTPKode Status: 400

ValidationException

Permintaan yang tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKpara Ruby V3

DescribeEndpoints
Layanan: Amazon Timestream Write

Mengembalikan daftar endpoint yang tersedia untuk membuat API panggilan Timestream terhadap. APIOperasi ini tersedia melalui Write dan QueryAPIs.

Karena Timestream SDKs dirancang untuk bekerja secara transparan dengan arsitektur layanan, termasuk pengelolaan dan pemetaan titik akhir layanan, kami tidak menyarankan Anda menggunakan operasi ini kecuali: API

- Anda menggunakan VPCendpoints (AWS PrivateLink) dengan Timestream
- Aplikasi Anda menggunakan bahasa pemrograman yang belum memiliki SDK dukungan
- Anda memerlukan kontrol yang lebih baik atas implementasi sisi klien

Untuk informasi rinci tentang bagaimana dan kapan menggunakan dan menerapkan DescribeEndpoints, lihat The Endpoint Discovery Pattern.

Sintaksis Respons

```
{
   "Endpoints": [
      {
         "Address": "string",
         "CachePeriodInMinutes": number
      }
   ]
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Data berikut dikembalikan dalam JSON format oleh layanan.

Endpoints

Sebuah `Endpoints` objek dikembalikan ketika `DescribeEndpoints` permintaan dibuat.

Tipe: Array objek Endpoint

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

InternalServerException

Timestream tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 500

ThrottlingException

Terlalu banyak permintaan yang dibuat oleh pengguna dan mereka melebihi kuota layanan. Permintaan itu dibatasi.

HTTPKode Status: 400

ValidationException

Permintaan yang tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKpara Ruby V3

DescribeTable
Layanan: Amazon Timestream Write

Mengembalikan informasi tentang tabel, termasuk nama tabel, nama database, durasi retensi penyimpanan memori dan penyimpanan magnetik. Service quotas berlaku. Lihat sampel kode untuk detail.

Sintaksis Permintaan

```
{
   "DatabaseName": "string",
   "TableName": "string"
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

DatabaseName

Nama basis data Timestream.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 256.

Wajib: Ya

TableName

Nama tabel Timestream.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 256.

Wajib: Ya

Sintaksis Respons

```
{
   "Table": {
```

```
      "Arn": "string",
      "CreationTime": number,
      "DatabaseName": "string",
      "LastUpdatedTime": number,
      "MagneticStoreWriteProperties": {
         "EnableMagneticStoreWrites": boolean,
         "MagneticStoreRejectedDataLocation": {
            "S3Configuration": {
               "BucketName": "string",
               "EncryptionOption": "string",
               "KmsKeyId": "string",
               "ObjectKeyPrefix": "string"
            }
         }
      },
      "RetentionProperties": {
         "MagneticStoreRetentionPeriodInDays": number,
         "MemoryStoreRetentionPeriodInHours": number
      },
      "Schema": {
         "CompositePartitionKey": [
            {
               "EnforcementInRecord": "string",
               "Name": "string",
               "Type": "string"
            }
         ]
      },
      "TableName": "string",
      "TableStatus": "string"
   }
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Data berikut dikembalikan dalam JSON format oleh layanan.

Table

Tabel Timestream.

Tipe: Objek Table

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

InternalServerException

Timestream tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 500

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

ResourceNotFoundException

Operasi mencoba mengakses sumber daya yang tidak ada. Sumber daya mungkin tidak ditentukan dengan benar, atau statusnya mungkin tidakACTIVE.

HTTPKode Status: 400

ThrottlingException

Terlalu banyak permintaan yang dibuat oleh pengguna dan mereka melebihi kuota layanan. Permintaan itu dibatasi.

HTTPKode Status: 400

ValidationException

Permintaan yang tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKuntuk Ruby V3

ListBatchLoadTasks
Layanan: Amazon Timestream Write

Menyediakan daftar tugas pemuatan batch, bersama dengan nama, status, kapan tugas dapat dilanjutkan hingga, dan detail lainnya. Lihat sampel kode untuk detail.

Sintaksis Permintaan

```
{
   "MaxResults": number,
   "NextToken": "string",
   "TaskStatus": "string"
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

MaxResults

Jumlah total item yang akan dikembalikan dalam output. Jika jumlah total item yang tersedia lebih dari nilai yang ditentukan, a NextToken disediakan dalam output. Untuk melanjutkan pagination, berikan NextToken nilai sebagai argumen dari API pemanggilan berikutnya.

Jenis: Integer

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 100.

Wajib: Tidak

NextToken

Token untuk menentukan di mana harus memulai paginating. Ini adalah NextToken dari respons yang sebelumnya terpotong.

Tipe: String

Wajib: Tidak

TaskStatus

Status tugas pemuatan batch.

Tipe: String

Nilai yang Valid: `CREATED | IN_PROGRESS | FAILED | SUCCEEDED | PROGRESS_STOPPED | PENDING_RESUME`

Wajib: Tidak

Sintaksis Respons

```
{
   "BatchLoadTasks": [
      {
         "CreationTime": number,
         "DatabaseName": "string",
         "LastUpdatedTime": number,
         "ResumableUntil": number,
         "TableName": "string",
         "TaskId": "string",
         "TaskStatus": "string"
      }
   ],
   "NextToken": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Data berikut dikembalikan dalam JSON format oleh layanan.

BatchLoadTasks

Daftar detail tugas pemuatan batch.

Tipe: Array objek BatchLoadTask

NextToken

Token untuk menentukan di mana harus memulai paginating. Berikan yang berikutnya ListBatchLoadTasksRequest.

Tipe: String

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

InternalServerException

Timestream tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 500

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

ThrottlingException

Terlalu banyak permintaan yang dibuat oleh pengguna dan mereka melebihi kuota layanan. Permintaan itu dibatasi.

HTTPKode Status: 400

ValidationException

Permintaan yang tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2

- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKpara Ruby V3

ListDatabases
Layanan: Amazon Timestream Write

Mengembalikan daftar database Timestream Anda. Service quotas berlaku. Lihat sampel kode untuk detail.

Sintaksis Permintaan

```
{
   "MaxResults": number,
   "NextToken": "string"
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

MaxResults

Jumlah total item yang akan dikembalikan dalam output. Jika jumlah total item yang tersedia lebih dari nilai yang ditentukan, a NextToken disediakan dalam output. Untuk melanjutkan pagination, berikan NextToken nilai sebagai argumen dari API pemanggilan berikutnya.

Jenis: Integer

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 20.

Wajib: Tidak

NextToken

Token pagination. Untuk melanjutkan pagination, berikan NextToken nilai sebagai argumen dari API pemanggilan berikutnya.

Tipe: String

Wajib: Tidak

Sintaksis Respons

```
{
```

```
   "Databases": [
      {
         "Arn": "string",
         "CreationTime": number,
         "DatabaseName": "string",
         "KmsKeyId": "string",
         "LastUpdatedTime": number,
         "TableCount": number
      }
   ],
   "NextToken": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Data berikut dikembalikan dalam JSON format oleh layanan.

Databases

Daftar nama database.

Tipe: Array objek Database

NextToken

Token pagination. Parameter ini dikembalikan ketika respon terpotong.

Tipe: String

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

InternalServerException

Timestream tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 500

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

ThrottlingException

Terlalu banyak permintaan yang dibuat oleh pengguna dan mereka melebihi kuota layanan. Permintaan itu dibatasi.

HTTPKode Status: 400

ValidationException

Permintaan yang tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKuntuk Ruby V3

ListTables
Layanan: Amazon Timestream Write

Menyediakan daftar tabel, bersama dengan nama, status, dan properti retensi dari setiap tabel. Lihat sampel kode untuk detail.

Sintaksis Permintaan

```
{
   "DatabaseName": "string",
   "MaxResults": number,
   "NextToken": "string"
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

DatabaseName

Nama basis data Timestream.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 256.

Wajib: Tidak

MaxResults

Jumlah total item yang akan dikembalikan dalam output. Jika jumlah total item yang tersedia lebih dari nilai yang ditentukan, a NextToken disediakan dalam output. Untuk melanjutkan pagination, berikan NextToken nilai sebagai argumen dari API pemanggilan berikutnya.

Jenis: Integer

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 20.

Wajib: Tidak

NextToken

Token pagination. Untuk melanjutkan pagination, berikan NextToken nilai sebagai argumen dari API pemanggilan berikutnya.

Tipe: String

Wajib: Tidak

Sintaksis Respons

```
{
   "NextToken": "string",
   "Tables": [
      {
         "Arn": "string",
         "CreationTime": number,
         "DatabaseName": "string",
         "LastUpdatedTime": number,
         "MagneticStoreWriteProperties": {
            "EnableMagneticStoreWrites": boolean,
            "MagneticStoreRejectedDataLocation": {
               "S3Configuration": {
                  "BucketName": "string",
                  "EncryptionOption": "string",
                  "KmsKeyId": "string",
                  "ObjectKeyPrefix": "string"
               }
            }
         },
         "RetentionProperties": {
            "MagneticStoreRetentionPeriodInDays": number,
            "MemoryStoreRetentionPeriodInHours": number
         },
         "Schema": {
            "CompositePartitionKey": [
               {
                  "EnforcementInRecord": "string",
                  "Name": "string",
                  "Type": "string"
               }
            ]
         },
```

```
            "TableName": "string",
            "TableStatus": "string"
        }
    ]
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Data berikut dikembalikan dalam JSON format oleh layanan.

NextToken

Token untuk menentukan di mana harus memulai paginating. Ini adalah NextToken dari respons yang sebelumnya terpotong.

Tipe: String

Tables

Daftar tabel.

Tipe: Array objek Table

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

InternalServerException

Timestream tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 500

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

ResourceNotFoundException

Operasi mencoba mengakses sumber daya yang tidak ada. Sumber daya mungkin tidak ditentukan dengan benar, atau statusnya mungkin tidakACTIVE.

HTTPKode Status: 400

ThrottlingException

Terlalu banyak permintaan yang dibuat oleh pengguna dan mereka melebihi kuota layanan. Permintaan itu dibatasi.

HTTPKode Status: 400

ValidationException

Permintaan yang tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKuntuk Ruby V3

ListTagsForResource
Layanan: Amazon Timestream Write

Daftar semua tag pada sumber daya Timestream.

Sintaksis Permintaan

```
{
   "ResourceARN": "string"
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

ResourceARN

Sumber daya Timestream dengan tag yang akan dicantumkan. Nilai ini adalah Nama Sumber Daya Amazon (ARN).

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 1011.

Wajib: Ya

Sintaksis Respons

```
{
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Data berikut dikembalikan dalam JSON format oleh layanan.

Tags

Tag yang saat ini terkait dengan sumber daya Timestream.

Tipe: Array objek Tag

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 200 item.

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

ResourceNotFoundException

Operasi mencoba mengakses sumber daya yang tidak ada. Sumber daya mungkin tidak ditentukan dengan benar, atau statusnya mungkin tidakACTIVE.

HTTPKode Status: 400

ThrottlingException

Terlalu banyak permintaan yang dibuat oleh pengguna dan mereka melebihi kuota layanan. Permintaan itu dibatasi.

HTTPKode Status: 400

ValidationException

Permintaan yang tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKpara Ruby V3

ResumeBatchLoadTask
Layanan: Amazon Timestream Write

Sintaksis Permintaan

```
{
   "TaskId": "string"
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

TaskId

ID tugas pemuatan batch untuk dilanjutkan.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum sebesar 32.

Pola: `[A-Z0-9]+`

Diperlukan: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan HTTP tubuh kosong.

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

InternalServerException

Timestream tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 500

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

ResourceNotFoundException

Operasi mencoba mengakses sumber daya yang tidak ada. Sumber daya mungkin tidak ditentukan dengan benar, atau statusnya mungkin tidakACTIVE.

HTTPKode Status: 400

ThrottlingException

Terlalu banyak permintaan yang dibuat oleh pengguna dan mereka melebihi kuota layanan. Permintaan itu dibatasi.

HTTPKode Status: 400

ValidationException

Permintaan yang tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3

- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKpara Ruby V3

TagResource
Layanan: Amazon Timestream Write

Mengaitkan satu set tag dengan sumber daya Timestream. Anda kemudian dapat mengaktifkan tag yang ditentukan pengguna ini sehingga muncul di konsol Billing and Cost Management untuk melacak alokasi biaya.

Sintaksis Permintaan

```
{
   "ResourceARN": "string",
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

ResourceARN

Mengidentifikasi sumber daya Timestream yang tag harus ditambahkan. Nilai ini adalah Nama Sumber Daya Amazon (ARN).

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 1011.

Wajib: Ya

Tags

Tag yang akan ditetapkan ke sumber daya Timestream.

Tipe: Array objek Tag

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 200 item.

Wajib: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan HTTP tubuh kosong.

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

ResourceNotFoundException

Operasi mencoba mengakses sumber daya yang tidak ada. Sumber daya mungkin tidak ditentukan dengan benar, atau statusnya mungkin tidakACTIVE.

HTTPKode Status: 400

ServiceQuotaExceededException

Kuota instans sumber daya terlampaui untuk akun ini.

HTTPKode Status: 400

ThrottlingException

Terlalu banyak permintaan yang dibuat oleh pengguna dan mereka melebihi kuota layanan. Permintaan itu dibatasi.

HTTPKode Status: 400

ValidationException

Permintaan yang tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKpara Ruby V3

UntagResource
Layanan: Amazon Timestream Write

Menghapus asosiasi tag dari sumber daya Timestream.

Sintaksis Permintaan

```
{
   "ResourceARN": "string",
   "TagKeys": [ "string" ]
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

ResourceARN

Sumber daya Timestream tempat tag akan dihapus. Nilai ini adalah Nama Sumber Daya Amazon (ARN).

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 1011.

Wajib: Ya

TagKeys

Daftar kunci tag. Tag yang ada dari sumber daya yang kuncinya adalah anggota daftar ini akan dihapus dari sumber daya Timestream.

Tipe: Array string.

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 200 item.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Wajib: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan HTTP tubuh kosong.

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

InvalidEndpointException

Titik akhir yang diminta tidak valid.

HTTPKode Status: 400

ResourceNotFoundException

Operasi mencoba mengakses sumber daya yang tidak ada. Sumber daya mungkin tidak ditentukan dengan benar, atau statusnya mungkin tidakACTIVE.

HTTPKode Status: 400

ServiceQuotaExceededException

Kuota instance sumber daya terlampaui untuk akun ini.

HTTPKode Status: 400

ThrottlingException

Terlalu banyak permintaan yang dibuat oleh pengguna dan mereka melebihi kuota layanan. Permintaan itu dibatasi.

HTTPKode Status: 400

ValidationException

Permintaan yang tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKuntuk Ruby V3

UpdateDatabase
Layanan: Amazon Timestream Write

Memodifikasi AWS KMS kunci untuk database yang ada. Saat memperbarui database, Anda harus menentukan nama database dan pengidentifikasi AWS KMS kunci baru yang akan digunakan (`KmsKeyId`). Jika ada `UpdateDatabase` permintaan bersamaan, penulis pertama menang.

Lihat sampel kode untuk detail.

Sintaksis Permintaan

```
{
   "DatabaseName": "string",
   "KmsKeyId": "string"
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

DatabaseName

Nama basis data.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 256.

Wajib: Ya

KmsKeyId

Pengidentifikasi AWS KMS kunci baru (`KmsKeyId`) yang akan digunakan untuk mengenkripsi data yang disimpan dalam database. Jika yang `KmsKeyId` saat ini terdaftar dengan database sama dengan permintaan, tidak akan ada pembaruan apa pun. `KmsKeyId`

Anda dapat menentukan `KmsKeyId` menggunakan salah satu hal berikut:

- ID kunci: `1234abcd-12ab-34cd-56ef-1234567890ab`
- KunciARN: `arn:aws:kms:us-east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Nama alias: `alias/ExampleAlias`

- AliasARN: `arn:aws:kms:us-east-1:111122223333:alias/ExampleAlias`

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Ya

Sintaksis Respons

```
{
   "Database": {
      "Arn": "string",
      "CreationTime": number,
      "DatabaseName": "string",
      "KmsKeyId": "string",
      "LastUpdatedTime": number,
      "TableCount": number
   }
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Data berikut dikembalikan dalam JSON format oleh layanan.

Database

Wadah tingkat atas untuk sebuah meja. Database dan tabel adalah konsep manajemen dasar di Amazon Timestream. Semua tabel dalam database dienkripsi dengan kunci yang sama AWS KMS .

Tipe: Objek Database

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

InternalServerException

Timestream tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 500

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

ResourceNotFoundException

Operasi mencoba mengakses sumber daya yang tidak ada. Sumber daya mungkin tidak ditentukan dengan benar, atau statusnya mungkin tidakACTIVE.

HTTPKode Status: 400

ServiceQuotaExceededException

Kuota instans sumber daya terlampaui untuk akun ini.

HTTPKode Status: 400

ThrottlingException

Terlalu banyak permintaan yang dibuat oleh pengguna dan mereka melebihi kuota layanan. Permintaan itu dibatasi.

HTTPKode Status: 400

ValidationException

Permintaan yang tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah

- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKpara Ruby V3

UpdateTable

Layanan: Amazon Timestream Write

Memodifikasi durasi retensi penyimpanan memori dan penyimpanan magnetik untuk tabel Timestream Anda. Perhatikan bahwa perubahan durasi retensi segera berlaku. Misalnya, jika periode retensi penyimpanan memori awalnya diatur ke 2 jam dan kemudian diubah menjadi 24 jam, penyimpanan memori akan mampu menyimpan 24 jam data, tetapi akan diisi dengan 24 jam data 22 jam setelah perubahan ini dilakukan. Timestream tidak mengambil data dari penyimpanan magnetik untuk mengisi penyimpanan memori.

Lihat sampel kode untuk detail.

Sintaksis Permintaan

```
{
   "DatabaseName": "string",
   "MagneticStoreWriteProperties": {
      "EnableMagneticStoreWrites": boolean,
      "MagneticStoreRejectedDataLocation": {
         "S3Configuration": {
            "BucketName": "string",
            "EncryptionOption": "string",
            "KmsKeyId": "string",
            "ObjectKeyPrefix": "string"
         }
      }
   },
   "RetentionProperties": {
      "MagneticStoreRetentionPeriodInDays": number,
      "MemoryStoreRetentionPeriodInHours": number
   },
   "Schema": {
      "CompositePartitionKey": [
         {
            "EnforcementInRecord": "string",
            "Name": "string",
            "Type": "string"
         }
      ]
   },
   "TableName": "string"
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

DatabaseName

Nama basis data Timestream.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 256.

Wajib: Ya

MagneticStoreWriteProperties

Berisi properti untuk diatur di atas meja saat mengaktifkan penulisan penyimpanan magnetik.

Tipe: Objek MagneticStoreWriteProperties

Wajib: Tidak

RetentionProperties

Durasi retensi penyimpanan memori dan penyimpanan magnetik.

Tipe: Objek RetentionProperties

Wajib: Tidak

Schema

Skema tabel.

Tipe: Objek Schema

Wajib: Tidak

TableName

Nama tabel Timestream.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 256.

Wajib: Ya

Sintaksis Respons

```
{
   "Table": {
      "Arn": "string",
      "CreationTime": number,
      "DatabaseName": "string",
      "LastUpdatedTime": number,
      "MagneticStoreWriteProperties": {
         "EnableMagneticStoreWrites": boolean,
         "MagneticStoreRejectedDataLocation": {
            "S3Configuration": {
               "BucketName": "string",
               "EncryptionOption": "string",
               "KmsKeyId": "string",
               "ObjectKeyPrefix": "string"
            }
         }
      },
      "RetentionProperties": {
         "MagneticStoreRetentionPeriodInDays": number,
         "MemoryStoreRetentionPeriodInHours": number
      },
      "Schema": {
         "CompositePartitionKey": [
            {
               "EnforcementInRecord": "string",
               "Name": "string",
               "Type": "string"
            }
         ]
      },
      "TableName": "string",
      "TableStatus": "string"
   }
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Data berikut dikembalikan dalam JSON format oleh layanan.

Table

Tabel Timestream yang diperbarui.

Tipe: Objek Table

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

InternalServerException

Timestream tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 500

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

ResourceNotFoundException

Operasi mencoba mengakses sumber daya yang tidak ada. Sumber daya mungkin tidak ditentukan dengan benar, atau statusnya mungkin tidakACTIVE.

HTTPKode Status: 400

ThrottlingException

Terlalu banyak permintaan yang dibuat oleh pengguna dan mereka melebihi kuota layanan. Permintaan itu dibatasi.

HTTPKode Status: 400

ValidationException

Permintaan yang tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKpara Ruby V3

WriteRecords
Layanan: Amazon Timestream Write

Memungkinkan Anda untuk menulis data deret waktu Anda ke Timestream. Anda dapat menentukan satu titik data atau sekumpulan titik data yang akan dimasukkan ke dalam sistem. Timestream menawarkan skema fleksibel yang secara otomatis mendeteksi nama kolom dan tipe data untuk tabel Timestream Anda berdasarkan nama dimensi dan tipe data dari titik data yang Anda tentukan saat menjalankan penulisan ke dalam database.

Timestream mendukung semantik baca konsistensi akhirnya. Ini berarti bahwa ketika Anda melakukan kueri data segera setelah menulis kumpulan data ke Timestream, hasil kueri mungkin tidak mencerminkan hasil operasi penulisan yang baru saja selesai. Hasilnya mungkin juga mencakup beberapa data basi. Jika Anda mengulangi permintaan kueri setelah waktu yang singkat, hasilnya akan mengembalikan data terbaru. Service quotas berlaku.

Lihat sampel kode untuk detail.

Upserts

Anda dapat menggunakan `Version` parameter dalam `WriteRecords` permintaan untuk memperbarui titik data. Timestream melacak nomor versi dengan setiap catatan. `Version`default 1 ketika tidak ditentukan untuk catatan dalam permintaan. Timestream memperbarui nilai ukuran rekaman yang ada bersama dengan `Version` saat menerima permintaan tulis dengan `Version` angka yang lebih tinggi untuk catatan itu. Ketika menerima permintaan pembaruan di mana nilai ukurannya sama dengan catatan yang ada, Timestream masih memperbarui`Version`, jika lebih besar dari `Version` nilai yang ada. Anda dapat memperbarui titik data sebanyak yang diinginkan, selama nilai `Version` terus meningkat.

Misalnya, Anda menulis catatan baru tanpa menunjukkan `Version` dalam permintaan. Timestream menyimpan catatan ini, dan diatur `Version` ke`1`. Sekarang, misalkan Anda mencoba memperbarui catatan ini dengan `WriteRecords` permintaan catatan yang sama dengan nilai ukuran yang berbeda tetapi, seperti sebelumnya, jangan berikan`Version`. Dalam hal ini, Timestream akan menolak pembaruan ini dengan `RejectedRecordsException` karena versi rekaman yang diperbarui tidak lebih besar dari nilai Versi yang ada.

Namun, jika Anda mengirim ulang permintaan pembaruan dengan `Version` set ke`2`, Timestream kemudian akan berhasil memperbarui nilai catatan, dan `Version` akan disetel ke. 2 Selanjutnya, misalkan Anda mengirim `WriteRecords` permintaan dengan catatan yang sama dan nilai ukuran yang identik, tetapi dengan `Version` set ke`3`. Dalam hal ini, Timestream hanya akan memperbarui

`Version` ke`3`. Setiap pembaruan lebih lanjut perlu mengirim nomor versi yang lebih besar dari`3`, atau permintaan pembaruan akan menerima file`RejectedRecordsException`.

Sintaksis Permintaan

```
{
   "CommonAttributes": {
      "Dimensions": [
         {
            "DimensionValueType": "string",
            "Name": "string",
            "Value": "string"
         }
      ],
      "MeasureName": "string",
      "MeasureValue": "string",
      "MeasureValues": [
         {
            "Name": "string",
            "Type": "string",
            "Value": "string"
         }
      ],
      "MeasureValueType": "string",
      "Time": "string",
      "TimeUnit": "string",
      "Version": number
   },
   "DatabaseName": "string",
   "Records": [
      {
         "Dimensions": [
            {
               "DimensionValueType": "string",
               "Name": "string",
               "Value": "string"
            }
         ],
         "MeasureName": "string",
         "MeasureValue": "string",
         "MeasureValues": [
            {
               "Name": "string",
               "Type": "string",
```

```
                "Value": "string"
            }
        ],
        "MeasureValueType": "string",
        "Time": "string",
        "TimeUnit": "string",
        "Version": number
    }
  ],
  "TableName": "string"
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

CommonAttributes

Catatan yang berisi atribut ukuran, dimensi, waktu, dan versi umum yang dibagikan di semua catatan dalam permintaan. Atribut ukuran dan dimensi yang ditentukan akan digabungkan dengan atribut ukuran dan dimensi dalam objek rekaman saat data ditulis ke Timestream. Dimensi mungkin tidak tumpang tindih, atau `ValidationException` akan dilemparkan. Dengan kata lain, catatan harus berisi dimensi dengan nama unik.

Tipe: Objek Record

Wajib: Tidak

DatabaseName

Nama basis data Timestream.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 256.

Wajib: Ya

Records

Larik catatan yang berisi atribut ukuran, dimensi, waktu, dan versi unik untuk setiap titik data deret waktu.

Tipe: Array objek Record

Anggota Array: Jumlah minimum 1 item. Jumlah maksimum 100 item.

Wajib: Ya

TableName

Nama tabel Timestream.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 256.

Wajib: Ya

Sintaksis Respons

```
{
   "RecordsIngested": {
      "MagneticStore": number,
      "MemoryStore": number,
      "Total": number
   }
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Data berikut dikembalikan dalam JSON format oleh layanan.

RecordsIngested

Informasi tentang catatan yang dicerna oleh permintaan ini.

Tipe: Objek RecordsIngested

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

InternalServerException

Timestream tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 500

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

RejectedRecordsException

WriteRecords akan membuang pengecualian ini dalam kasus berikut:

- Rekaman dengan data duplikat di mana ada beberapa catatan dengan dimensi, stempel waktu, dan nama ukuran yang sama tetapi:
  - Nilai ukur berbeda
  - Versi tidak hadir dalam permintaan atau nilai versi dalam catatan baru sama dengan atau lebih rendah dari nilai yang ada

  Dalam hal ini, jika Timestream menolak data, `ExistingVersion` bidang dalam `RejectedRecords` respons akan menunjukkan versi rekaman saat ini. Untuk memaksa pembaruan, Anda dapat mengirim ulang permintaan dengan versi untuk catatan yang ditetapkan ke nilai yang lebih besar dari`ExistingVersion`.
- Catatan dengan stempel waktu yang berada di luar durasi retensi penyimpanan memori.
- Rekaman dengan dimensi atau ukuran yang melebihi batas yang ditentukan Timestream.

Untuk informasi selengkapnya, lihat Kuota di Panduan Pengembang Amazon Timestream.

HTTPKode Status: 400

ResourceNotFoundException

Operasi mencoba mengakses sumber daya yang tidak ada. Sumber daya mungkin tidak ditentukan dengan benar, atau statusnya mungkin tidakACTIVE.

HTTPKode Status: 400

ThrottlingException

Terlalu banyak permintaan yang dibuat oleh pengguna dan mereka melebihi kuota layanan. Permintaan itu dibatasi.

HTTPKode Status: 400

ValidationException

Permintaan yang tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKuntuk Ruby V3

Kueri Amazon Timestream

Tindakan berikut didukung oleh Amazon Timestream Query:

- CancelQuery
- CreateScheduledQuery
- DeleteScheduledQuery
- DescribeAccountSettings

- DescribeEndpoints
- DescribeScheduledQuery
- ExecuteScheduledQuery
- ListScheduledQueries
- ListTagsForResource
- PrepareQuery
- Query
- TagResource
- UntagResource
- UpdateAccountSettings
- UpdateScheduledQuery

CancelQuery

Layanan: Amazon Timestream Query

Membatalkan kueri yang telah dikeluarkan. Pembatalan diberikan hanya jika kueri belum selesai berjalan sebelum permintaan pembatalan dikeluarkan. Karena pembatalan adalah operasi idempoten, permintaan pembatalan berikutnya akan mengembalikan a`CancellationMessage`, yang menunjukkan bahwa kueri telah dibatalkan. Lihat sampel kode untuk detail.

Sintaksis Permintaan

```
{
   "QueryId": "string"
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

QueryId

ID kueri yang perlu dibatalkan. `QueryID`dikembalikan sebagai bagian dari hasil kueri.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `[a-zA-Z0-9]+`

Diperlukan: Ya

Sintaksis Respons

```
{
   "CancellationMessage": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Data berikut dikembalikan dalam JSON format oleh layanan.

CancellationMessage

A `CancellationMessage` dikembalikan ketika `CancelQuery` permintaan untuk kueri yang ditentukan oleh `QueryId` telah dikeluarkan.

Tipe: String

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

InternalServerException

Layanan tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 400

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

ThrottlingException

Permintaan ditolak karena throttling permintaan.

HTTPKode Status: 400

ValidationException

Permintaan tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKpara Ruby V3

CreateScheduledQuery

Layanan: Amazon Timestream Query

Buat kueri terjadwal yang akan dijalankan atas nama Anda pada jadwal yang dikonfigurasi. Timestream mengasumsikan peran eksekusi yang disediakan sebagai bagian dari `ScheduledQueryExecutionRoleArn` parameter untuk menjalankan kueri. Anda dapat menggunakan `NotificationConfiguration` parameter untuk mengonfigurasi notifikasi untuk operasi kueri terjadwal Anda.

Sintaksis Permintaan

```
{
   "ClientToken": "string",
   "ErrorReportConfiguration": {
      "S3Configuration": {
         "BucketName": "string",
         "EncryptionOption": "string",
         "ObjectKeyPrefix": "string"
      }
   },
   "KmsKeyId": "string",
   "Name": "string",
   "NotificationConfiguration": {
      "SnsConfiguration": {
         "TopicArn": "string"
      }
   },
   "QueryString": "string",
   "ScheduleConfiguration": {
      "ScheduleExpression": "string"
   },
   "ScheduledQueryExecutionRoleArn": "string",
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ],
   "TargetConfiguration": {
      "TimestreamConfiguration": {
         "DatabaseName": "string",
         "DimensionMappings": [
            {
```

```
                "DimensionValueType": "string",
                "Name": "string"
            }
         ],
         "MeasureNameColumn": "string",
         "MixedMeasureMappings": [
            {
               "MeasureName": "string",
               "MeasureValueType": "string",
               "MultiMeasureAttributeMappings": [
                  {
                     "MeasureValueType": "string",
                     "SourceColumn": "string",
                     "TargetMultiMeasureAttributeName": "string"
                  }
               ],
               "SourceColumn": "string",
               "TargetMeasureName": "string"
            }
         ],
         "MultiMeasureMappings": {
            "MultiMeasureAttributeMappings": [
               {
                  "MeasureValueType": "string",
                  "SourceColumn": "string",
                  "TargetMultiMeasureAttributeName": "string"
               }
            ],
            "TargetMultiMeasureName": "string"
         },
         "TableName": "string",
         "TimeColumn": "string"
      }
   }
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

ClientToken

Menggunakan ClientToken membuat panggilan ke CreateScheduledQuery idempoten, dengan kata lain, membuat permintaan yang sama berulang kali akan menghasilkan hasil yang sama. Membuat beberapa CreateScheduledQuery permintaan identik memiliki efek yang sama seperti membuat satu permintaan.

- Jika CreateScheduledQuery dipanggil tanpa a`ClientToken`, Query SDK menghasilkan `ClientToken` atas nama Anda.
- Setelah 8 jam, permintaan apa pun dengan hal `ClientToken` yang sama diperlakukan sebagai permintaan baru.

Tipe: String

Kendala Panjang: Panjang minimum 32. Panjang maksimum 128.

Wajib: Tidak

ErrorReportConfiguration

Konfigurasi untuk pelaporan kesalahan. Laporan kesalahan akan dihasilkan ketika masalah ditemui saat menulis hasil kueri.

Tipe: Objek ErrorReportConfiguration

Wajib: Ya

KmsKeyId

KMSKunci Amazon digunakan untuk mengenkripsi sumber daya kueri terjadwal, saat istirahat. Jika KMS kunci Amazon tidak ditentukan, sumber kueri terjadwal akan dienkripsi dengan kunci Amazon milik Timestream. KMS Untuk menentukan KMS kunci, gunakan ID kunci, kunciARN, nama alias, atau aliasARN. Saat menggunakan nama alias, awali nama dengan alias/

Jika ErrorReportConfiguration digunakan `SSE_KMS` sebagai jenis enkripsi, hal yang sama KmsKeyId digunakan untuk mengenkripsi laporan kesalahan saat istirahat.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Tidak

Name

Nama kueri terjadwal.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Wajib: Ya

NotificationConfiguration

Konfigurasi pemberitahuan untuk kueri terjadwal. Pemberitahuan dikirim oleh Timestream saat proses kueri selesai, saat status diperbarui atau saat Anda menghapusnya.

Tipe: Objek NotificationConfiguration

Wajib: Ya

QueryString

String query untuk dijalankan. Nama parameter dapat ditentukan dalam @ karakter string query diikuti oleh identifier. Parameter bernama `@scheduled_runtime` dicadangkan dan dapat digunakan dalam kueri untuk mendapatkan waktu di mana kueri dijadwalkan untuk dijalankan.

Stempel waktu yang dihitung sesuai dengan ScheduleConfiguration parameter, akan menjadi nilai `@scheduled_runtime` paramater untuk setiap kueri yang dijalankan. Misalnya, pertimbangkan instance kueri terjadwal yang dijalankan pada 2021-12-01 00:00:00. Untuk contoh ini, `@scheduled_runtime` parameter diinisialisasi ke stempel waktu 2021-12-01 00:00:00 saat menjalankan kueri.

Tipe: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 262144.

Wajib: Ya

ScheduleConfiguration

Konfigurasi jadwal untuk kueri.

Tipe: Objek ScheduleConfiguration

Wajib: Ya

ScheduledQueryExecutionRoleArn

ARNUntuk IAM peran yang Timestream akan asumsikan saat menjalankan kueri terjadwal.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Ya

Tags

Daftar pasangan kunci-nilai untuk memberi label pada kueri terjadwal.

Tipe: Array objek Tag

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 200 item.

Wajib: Tidak

TargetConfiguration

Konfigurasi yang digunakan untuk menulis hasil query.

Tipe: Objek TargetConfiguration

Wajib: Tidak

Sintaksis Respons

```
{
   "Arn": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Data berikut dikembalikan dalam JSON format oleh layanan.

Arn

ARNuntuk kueri terjadwal yang dibuat.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

ConflictException

Tidak dapat melakukan polling hasil untuk kueri yang dibatalkan.

HTTPKode Status: 400

InternalServerException

Layanan tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 400

InvalidEndpointException

Titik akhir yang diminta tidak valid.

HTTPKode Status: 400

ServiceQuotaExceededException

Anda telah melampaui kuota layanan.

HTTPKode Status: 400

ThrottlingException

Permintaan ditolak karena throttling permintaan.

HTTPKode Status: 400

ValidationException

Permintaan tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKpara Ruby V3

DeleteScheduledQuery
Layanan: Amazon Timestream Query

Menghapus kueri terjadwal yang diberikan. Ini adalah operasi yang tidak dapat diubah.

Sintaksis Permintaan

```
{
   "ScheduledQueryArn": "string"
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

ScheduledQueryArn

Kueri yang dijadwalkan. ARN

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan HTTP tubuh kosong.

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

InternalServerException

Layanan tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 400

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

ResourceNotFoundException

Sumber daya yang diminta tidak dapat ditemukan.

HTTPKode Status: 400

ThrottlingException

Permintaan ditolak karena throttling permintaan.

HTTPKode Status: 400

ValidationException

Permintaan tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3

- AWS SDKuntuk Python
- AWS SDKpara Ruby V3

DescribeAccountSettings
Layanan: Amazon Timestream Query

Menjelaskan pengaturan untuk akun Anda yang menyertakan model harga kueri dan maksimum yang dikonfigurasi TCUs yang dapat digunakan layanan untuk beban kerja kueri Anda.

Anda dikenakan biaya hanya untuk durasi unit komputasi yang digunakan untuk beban kerja Anda.

Sintaksis Respons

```
{
   "MaxQueryTCU": number,
   "QueryPricingModel": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Data berikut dikembalikan dalam JSON format oleh layanan.

MaxQueryTCU

Jumlah maksimum unit komputasi Timestream (TCUs) yang akan digunakan layanan kapan saja untuk melayani kueri Anda.

Jenis: Integer

QueryPricingModel

Model harga untuk kueri di akun Anda.

Tipe: String

Nilai yang Valid: `BYTES_SCANNED | COMPUTE_UNITS`

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

InternalServerException

Layanan tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 400

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

ThrottlingException

Permintaan ditolak karena throttling permintaan.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKuntuk Ruby V3

DescribeEndpoints
Layanan: Amazon Timestream Query

DescribeEndpoints mengembalikan daftar endpoint yang tersedia untuk membuat API panggilan Timestream terhadap. Ini API tersedia melalui Write dan Query.

Karena Timestream SDKs dirancang untuk bekerja secara transparan dengan arsitektur layanan, termasuk pengelolaan dan pemetaan titik akhir layanan, Anda tidak disarankan untuk menggunakan ini kecuali: API

- Anda menggunakan VPCendpoints (AWS PrivateLink) dengan Timestream
- Aplikasi Anda menggunakan bahasa pemrograman yang belum memiliki SDK dukungan
- Anda memerlukan kontrol yang lebih baik atas implementasi sisi klien

Untuk informasi rinci tentang bagaimana dan kapan menggunakan dan menerapkan DescribeEndpoints, lihat The Endpoint Discovery Pattern.

Sintaksis Respons

```
{
   "Endpoints": [
      {
         "Address": "string",
         "CachePeriodInMinutes": number
      }
   ]
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Data berikut dikembalikan dalam JSON format oleh layanan.

Endpoints

Sebuah `Endpoints` objek dikembalikan ketika `DescribeEndpoints` permintaan dibuat.

Tipe: Array objek Endpoint

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

InternalServerException

Layanan tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 400

ThrottlingException

Permintaan ditolak karena throttling permintaan.

HTTPKode Status: 400

ValidationException

Permintaan tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKpara Ruby V3

DescribeScheduledQuery
Layanan: Amazon Timestream Query

Memberikan informasi rinci tentang kueri terjadwal.

Sintaksis Permintaan

```
{
   "ScheduledQueryArn": "string"
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

ScheduledQueryArn

Kueri yang dijadwalkan. ARN

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Ya

Sintaksis Respons

```
{
   "ScheduledQuery": {
      "Arn": "string",
      "CreationTime": number,
      "ErrorReportConfiguration": {
         "S3Configuration": {
            "BucketName": "string",
            "EncryptionOption": "string",
            "ObjectKeyPrefix": "string"
         }
      },
      "KmsKeyId": "string",
      "LastRunSummary": {
         "ErrorReportLocation": {
```

```
            "S3ReportLocation": {
               "BucketName": "string",
               "ObjectKey": "string"
            }
         },
         "ExecutionStats": {
            "BytesMetered": number,
            "CumulativeBytesScanned": number,
            "DataWrites": number,
            "ExecutionTimeInMillis": number,
            "QueryResultRows": number,
            "RecordsIngested": number
         },
         "FailureReason": "string",
         "InvocationTime": number,
         "QueryInsightsResponse": {
            "OutputBytes": number,
            "OutputRows": number,
            "QuerySpatialCoverage": {
               "Max": {
                  "PartitionKey": [ "string" ],
                  "TableArn": "string",
                  "Value": number
               }
            },
            "QueryTableCount": number,
            "QueryTemporalRange": {
               "Max": {
                  "TableArn": "string",
                  "Value": number
               }
            }
         },
         "RunStatus": "string",
         "TriggerTime": number
      },
      "Name": "string",
      "NextInvocationTime": number,
      "NotificationConfiguration": {
         "SnsConfiguration": {
            "TopicArn": "string"
         }
      },
      "PreviousInvocationTime": number,
```

```
      "QueryString": "string",
      "RecentlyFailedRuns": [
         {
            "ErrorReportLocation": {
               "S3ReportLocation": {
                  "BucketName": "string",
                  "ObjectKey": "string"
               }
            },
            "ExecutionStats": {
               "BytesMetered": number,
               "CumulativeBytesScanned": number,
               "DataWrites": number,
               "ExecutionTimeInMillis": number,
               "QueryResultRows": number,
               "RecordsIngested": number
            },
            "FailureReason": "string",
            "InvocationTime": number,
            "QueryInsightsResponse": {
               "OutputBytes": number,
               "OutputRows": number,
               "QuerySpatialCoverage": {
                  "Max": {
                     "PartitionKey": [ "string" ],
                     "TableArn": "string",
                     "Value": number
                  }
               },
               "QueryTableCount": number,
               "QueryTemporalRange": {
                  "Max": {
                     "TableArn": "string",
                     "Value": number
                  }
               }
            },
            "RunStatus": "string",
            "TriggerTime": number
         }
      ],
      "ScheduleConfiguration": {
         "ScheduleExpression": "string"
      },
```

```
      "ScheduledQueryExecutionRoleArn": "string",
      "State": "string",
      "TargetConfiguration": {
         "TimestreamConfiguration": {
            "DatabaseName": "string",
            "DimensionMappings": [
               {
                  "DimensionValueType": "string",
                  "Name": "string"
               }
            ],
            "MeasureNameColumn": "string",
            "MixedMeasureMappings": [
               {
                  "MeasureName": "string",
                  "MeasureValueType": "string",
                  "MultiMeasureAttributeMappings": [
                     {
                        "MeasureValueType": "string",
                        "SourceColumn": "string",
                        "TargetMultiMeasureAttributeName": "string"
                     }
                  ],
                  "SourceColumn": "string",
                  "TargetMeasureName": "string"
               }
            ],
            "MultiMeasureMappings": {
               "MultiMeasureAttributeMappings": [
                  {
                     "MeasureValueType": "string",
                     "SourceColumn": "string",
                     "TargetMultiMeasureAttributeName": "string"
                  }
               ],
               "TargetMultiMeasureName": "string"
            },
            "TableName": "string",
            "TimeColumn": "string"
         }
      }
   }
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Data berikut dikembalikan dalam JSON format oleh layanan.

[ScheduledQuery](#)

Permintaan yang dijadwalkan.

Tipe: Objek [ScheduledQueryDescription](#)

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat [Kesalahan Umum](#).

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

InternalServerException

Layanan tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 400

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

ResourceNotFoundException

Sumber daya yang diminta tidak dapat ditemukan.

HTTPKode Status: 400

ThrottlingException

Permintaan ditolak karena throttling permintaan.

HTTPKode Status: 400

ValidationException

Permintaan tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKpara Ruby V3

ExecuteScheduledQuery
Layanan: Amazon Timestream Query

Anda dapat menggunakan ini API untuk menjalankan kueri terjadwal secara manual.

Jika Anda mengaktifkan`QueryInsights`, ini API juga menampilkan wawasan dan metrik yang terkait dengan kueri yang Anda jalankan sebagai bagian dari notifikasi AmazonSNS. `QueryInsights`membantu penyetelan kinerja kueri Anda. Untuk informasi selengkapnya`QueryInsights`, lihat Menggunakan wawasan kueri untuk mengoptimalkan kueri di Amazon Timestream.

Sintaksis Permintaan

```
{
   "ClientToken": "string",
   "InvocationTime": number,
   "QueryInsights": {
      "Mode": "string"
   },
   "ScheduledQueryArn": "string"
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

ClientToken

Tidak digunakan.

Tipe: String

Kendala Panjang: Panjang minimum 32. Panjang maksimum 128.

Wajib: Tidak

InvocationTime

Stempel waktu masuk. UTC Kueri akan dijalankan seolah-olah dipanggil pada stempel waktu ini.

Tipe: Timestamp

Wajib: Ya

QueryInsights

Merangkum pengaturan untuk mengaktifkan. `QueryInsights`

Mengaktifkan `QueryInsights` mengembalikan wawasan dan metrik sebagai bagian dari SNS notifikasi Amazon untuk kueri yang Anda jalankan. Anda dapat menggunakan `QueryInsights` untuk menyetel kinerja dan biaya kueri Anda.

Tipe: Objek ScheduledQueryInsights

Wajib: Tidak

ScheduledQueryArn

ARNdari query yang dijadwalkan.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan HTTP tubuh kosong.

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

InternalServerException

Layanan tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 400

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

ResourceNotFoundException

Sumber daya yang diminta tidak dapat ditemukan.

HTTPKode Status: 400

ThrottlingException

Permintaan ditolak karena throttling permintaan.

HTTPKode Status: 400

ValidationException

Permintaan tidak valid atau cacat.

HTTPKode Status: 400

Contoh

Pesan pemberitahuan kueri terjadwal untuk CONTROL mode ENABLED WITH _ RATE _ _

Contoh berikut menunjukkan pesan notifikasi kueri terjadwal yang berhasil untuk `ENABLED_WITH_RATE_CONTROL` mode `QueryInsights` parameter.

```
"SuccessNotificationMessage": {
    "type": "MANUAL_TRIGGER_SUCCESS",
    "arn": "arn:aws:timestream:<Region>:<Account>:scheduled-query/sq-test-49c6ed55-
c2e7-4cc2-9956-4a0ecea13420-80e05b035236a4c3",
    "scheduledQueryRunSummary": {
        "invocationEpochSecond": 1723710546,
        "triggerTimeMillis": 1723710547490,
        "runStatus": "MANUAL_TRIGGER_SUCCESS",
        "executionStats": {
            "executionTimeInMillis": 17343,
            "dataWrites": 1024,
```

```
            "bytesMetered": 0,
            "cumulativeBytesScanned": 600,
            "recordsIngested": 1,
            "queryResultRows": 1
        },
        "queryInsightsResponse": {
            "querySpatialCoverage": {
                "max": {
                    "value": 1.0,
                    "tableArn": "arn:aws:timestream:<Region>:<Account>:database/BaseDb/
table/BaseTable",
                    "partitionKey": [
                        "measure_name"
                    ]
                }
            },
            "queryTemporalRange": {
                "max": {
                    "value": 2399999999999,
                    "tableArn": "arn:aws:timestream:<Region>:<Account>:database/BaseDb/
table/BaseTable"
                }
            },
            "queryTableCount": 1,
            "outputRows": 1,
            "outputBytes": 59
        }
    }
}
```

Pesan pemberitahuan kueri terjadwal untuk DISABLED mode

Contoh berikut menunjukkan pesan notifikasi kueri terjadwal yang berhasil untuk `DISABLED` mode `QueryInsights` parameter.

```
"SuccessNotificationMessage": {
    "type": "MANUAL_TRIGGER_SUCCESS",
    "arn": "arn:aws:timestream:<Region>:<Account>:scheduled-query/sq-test-
fa109d9e-6528-4a0d-ac40-482fa05e657f-140faaeecdc5b2a7",
    "scheduledQueryRunSummary": {
        "invocationEpochSecond": 1723711401,
        "triggerTimeMillis": 1723711402144,
        "runStatus": "MANUAL_TRIGGER_SUCCESS",
```

```
        "executionStats": {
            "executionTimeInMillis": 17992,
            "dataWrites": 1024,
            "bytesMetered": 0,
            "cumulativeBytesScanned": 600,
            "recordsIngested": 1,
            "queryResultRows": 1
        }
    }
}
```

Pesan pemberitahuan kegagalan untuk CONTROL mode ENABLED WITH _ RATE _ _

Contoh berikut menunjukkan pesan pemberitahuan kueri terjadwal gagal untuk `ENABLED_WITH_RATE_CONTROL` mode `QueryInsights` parameter.

```
"FailureNotificationMessage": {
    "type": "MANUAL_TRIGGER_FAILURE",
    "arn": "arn:aws:timestream:<Region>:<Account>:scheduled-query/sq-test-
b261670d-790c-4116-9db5-0798071b18b1-b7e27a1d79be226d",
    "scheduledQueryRunSummary": {
        "invocationEpochSecond": 1727915513,
        "triggerTimeMillis": 1727915513894,
        "runStatus": "MANUAL_TRIGGER_FAILURE",
        "executionStats": {
            "executionTimeInMillis": 10777,
            "dataWrites": 0,
            "bytesMetered": 0,
            "cumulativeBytesScanned": 0,
            "recordsIngested": 0,
            "queryResultRows": 4
        },
        "errorReportLocation": {
            "s3ReportLocation": {
                "bucketName": "my-amzn-s3-demo-bucket",
                "objectKey": "4my-organization-f7a3c5d065a1a95e/1727915513/
MANUAL/1727915513894/5e14b3df-b147-49f4-9331-784f749b68ae"
            }
        },
        "failureReason": "Schedule encountered some errors and is incomplete. Please
 take a look at error report for further details"
    }
}
```

Pesan pemberitahuan kegagalan untuk DISABLED mode

Contoh berikut menunjukkan pesan pemberitahuan kueri terjadwal gagal untuk `DISABLED` mode `QueryInsights` parameter.

```
"FailureNotificationMessage": {
    "type": "MANUAL_TRIGGER_FAILURE",
    "arn": "arn:aws:timestream:<Region>:<Account>:scheduled-query/sq-test-
b261670d-790c-4116-9db5-0798071b18b1-b7e27a1d79be226d",
    "scheduledQueryRunSummary": {
        "invocationEpochSecond": 1727915194,
        "triggerTimeMillis": 1727915195119,
        "runStatus": "MANUAL_TRIGGER_FAILURE",
        "executionStats": {
            "executionTimeInMillis": 10777,
            "dataWrites": 0,
            "bytesMetered": 0,
            "cumulativeBytesScanned": 0,
            "recordsIngested": 0,
            "queryResultRows": 4
        },
        "errorReportLocation": {
            "s3ReportLocation": {
                "bucketName": "my-amzn-s3-demo-bucket",
                "objectKey": "4my-organization-b7e27a1d79be226d/1727915194/
MANUAL/1727915195119/08dea9f5-9a0a-4e63-a5f7-ded23247bb98"
            }
        },
        "failureReason": "Schedule encountered some errors and is incomplete. Please
 take a look at error report for further details"
    }
}
```

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2

- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKpara Ruby V3

ListScheduledQueries
Layanan: Amazon Timestream Query

Mendapat daftar semua kueri terjadwal di akun Amazon dan Wilayah pemanggil. `ListScheduledQueries`Akhirnya konsisten.

Sintaksis Permintaan

```
{
   "MaxResults": number,
   "NextToken": "string"
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

MaxResults

Jumlah maksimum item untuk dikembalikan dalam output. Jika jumlah total item yang tersedia lebih dari nilai yang ditentukan, a `NextToken` disediakan dalam output. Untuk melanjutkan pagination, berikan `NextToken` nilai sebagai argumen untuk `ListScheduledQueriesRequest` panggilan berikutnya.

Jenis: Integer

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 1000.

Wajib: Tidak

NextToken

Token pagination untuk melanjutkan pagination.

Tipe: String

Wajib: Tidak

Sintaksis Respons

```
{
```

```
   "NextToken": "string",
   "ScheduledQueries": [
      {
         "Arn": "string",
         "CreationTime": number,
         "ErrorReportConfiguration": {
            "S3Configuration": {
               "BucketName": "string",
               "EncryptionOption": "string",
               "ObjectKeyPrefix": "string"
            }
         },
         "LastRunStatus": "string",
         "Name": "string",
         "NextInvocationTime": number,
         "PreviousInvocationTime": number,
         "State": "string",
         "TargetDestination": {
            "TimestreamDestination": {
               "DatabaseName": "string",
               "TableName": "string"
            }
         }
      }
   ]
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Data berikut dikembalikan dalam JSON format oleh layanan.

NextToken

Token untuk menentukan di mana harus memulai paginating. Ini adalah NextToken dari respons yang sebelumnya terpotong.

Tipe: String

ScheduledQueries

Daftar kueri terjadwal.

Tipe: Array objek ScheduledQuery

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

InternalServerException

Layanan tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 400

InvalidEndpointException

Titik akhir yang diminta tidak valid.

HTTPKode Status: 400

ThrottlingException

Permintaan ditolak karena throttling permintaan.

HTTPKode Status: 400

ValidationException

Permintaan tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2

- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKuntuk Ruby V3

ListTagsForResource
Layanan: Amazon Timestream Query

Daftar semua tag pada sumber kueri Timestream.

Sintaksis Permintaan

```
{
   "MaxResults": number,
   "NextToken": "string",
   "ResourceARN": "string"
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

MaxResults

Jumlah maksimum tag untuk dikembalikan.

Jenis: Integer

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 200.

Wajib: Tidak

NextToken

Token pagination untuk melanjutkan pagination.

Tipe: String

Wajib: Tidak

ResourceARN

Sumber daya Timestream dengan tag yang akan dicantumkan. Nilai ini adalah Nama Sumber Daya Amazon (ARN).

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Ya

Sintaksis Respons

```
{
   "NextToken": "string",
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Data berikut dikembalikan dalam JSON format oleh layanan.

NextToken

Token pagination untuk melanjutkan pagination dengan panggilan berikutnya ke. `ListTagsForResourceResponse`

Tipe: String

Tags

Tag yang saat ini terkait dengan sumber daya Timestream.

Tipe: Array objek Tag

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 200 item.

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

ResourceNotFoundException

Sumber daya yang diminta tidak dapat ditemukan.

HTTPKode Status: 400

ThrottlingException

Permintaan ditolak karena throttling permintaan.

HTTPKode Status: 400

ValidationException

Permintaan tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKpara Ruby V3

PrepareQuery
Layanan: Amazon Timestream Query

Operasi sinkron yang memungkinkan Anda mengirimkan kueri dengan parameter yang akan disimpan oleh Timestream untuk dijalankan nanti. Timestream hanya mendukung penggunaan operasi ini dengan `ValidateOnly` set to`true`.

Sintaksis Permintaan

```
{
   "QueryString": "string",
   "ValidateOnly": boolean
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

QueryString

String query Timestream yang ingin Anda gunakan sebagai pernyataan siap. Nama parameter dapat ditentukan dalam @ karakter string query diikuti oleh identifier.

Tipe: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 262144.

Wajib: Ya

ValidateOnly

Dengan menyetel nilai ini`true`, Timestream hanya akan memvalidasi bahwa string kueri adalah kueri Timestream yang valid, dan tidak menyimpan kueri yang disiapkan untuk digunakan nanti.

Tipe: Boolean

Wajib: Tidak

Sintaksis Respons

```
{
   "Columns": [
```

```
      {
         "Aliased": boolean,
         "DatabaseName": "string",
         "Name": "string",
         "TableName": "string",
         "Type": {
            "ArrayColumnInfo": {
               "Name": "string",
               "Type": "Type"
            },
            "RowColumnInfo": [
               {
                  "Name": "string",
                  "Type": "Type"
               }
            ],
            "ScalarType": "string",
            "TimeSeriesMeasureValueColumnInfo": {
               "Name": "string",
               "Type": "Type"
            }
         }
      }
   ],
   "Parameters": [
      {
         "Name": "string",
         "Type": {
            "ArrayColumnInfo": {
               "Name": "string",
               "Type": "Type"
            },
            "RowColumnInfo": [
               {
                  "Name": "string",
                  "Type": "Type"
               }
            ],
            "ScalarType": "string",
            "TimeSeriesMeasureValueColumnInfo": {
               "Name": "string",
               "Type": "Type"
            }
         }
```

```
        }
    ],
    "QueryString": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Data berikut dikembalikan dalam JSON format oleh layanan.

Columns

Daftar kolom SELECT klausa dari string kueri yang dikirimkan.

Tipe: Array objek SelectColumn

Parameters

Daftar parameter yang digunakan dalam string kueri yang dikirimkan.

Tipe: Array objek ParameterMapping

QueryString

String kueri yang ingin Anda siapkan.

Tipe: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 262144.

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

InternalServerException

Layanan tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 400

InvalidEndpointException

Titik akhir yang diminta tidak valid.

HTTPKode Status: 400

ThrottlingException

Permintaan ditolak karena throttling permintaan.

HTTPKode Status: 400

ValidationException

Permintaan tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKpara Ruby V3

Query
Layanan: Amazon Timestream Query

`Query`adalah operasi sinkron yang memungkinkan Anda menjalankan kueri terhadap data Amazon Timestream Anda.

Jika Anda mengaktifkan`QueryInsights`, ini API juga menampilkan wawasan dan metrik yang terkait dengan kueri yang Anda jalankan. `QueryInsights`membantu penyetelan kinerja kueri Anda. Untuk informasi selengkapnya`QueryInsights`, lihat Menggunakan wawasan kueri untuk mengoptimalkan kueri di Amazon Timestream.

> **Note**
>
> Jumlah maksimum `Query` API permintaan yang diizinkan untuk Anda buat dengan `QueryInsights` diaktifkan adalah 1 kueri per detik (QPS). Jika Anda melebihi tingkat kueri ini, itu mungkin mengakibatkan pelambatan.

`Query`akan habis setelah 60 detik. Anda harus memperbarui batas waktu default SDK untuk mendukung batas waktu 60 detik. Lihat contoh kode untuk detailnya.

Permintaan kueri Anda akan gagal dalam kasus berikut:

- Jika Anda mengirimkan `Query` permintaan dengan token klien yang sama di luar jendela idempotensi 5 menit.
- Jika Anda mengirimkan `Query` permintaan dengan token klien yang sama, tetapi mengubah parameter lain, dalam jendela idempotensi 5 menit.
- Jika ukuran baris (termasuk metadata kueri) melebihi 1 MB, maka kueri akan gagal dengan pesan kesalahan berikut:

  ```
  Query aborted as max page response size has been exceeded by the output result row
  ```

- Jika IAM prinsipal inisiator kueri dan pembaca hasil tidak sama dan/atau inisiator kueri dan pembaca hasil tidak memiliki string kueri yang sama dalam permintaan kueri, kueri akan gagal dengan kesalahan`Invalid pagination token`.

Sintaksis Permintaan

```
{
   "ClientToken": "string",
   "MaxRows": number,
   "NextToken": "string",
   "QueryInsights": {
      "Mode": "string"
   },
   "QueryString": "string"
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

ClientToken

String unik dan peka huruf besar/kecil hingga 64 ASCII karakter yang ditentukan saat `Query` permintaan dibuat. `ClientToken`Memberikan panggilan untuk `Query` idempoten. Ini berarti bahwa menjalankan kueri yang sama berulang kali akan menghasilkan hasil yang sama. Dengan kata lain, membuat beberapa `Query` permintaan identik memiliki efek yang sama seperti membuat permintaan tunggal. Saat menggunakan `ClientToken` dalam kueri, perhatikan hal berikut:

- Jika Query API dipakai tanpa a`ClientToken`, Query akan SDK menghasilkan a `ClientToken` atas nama Anda.
- Jika `Query` pemanggilan hanya berisi `ClientToken` tetapi tidak menyertakan a`NextToken`, pemanggilan tersebut diasumsikan sebagai `Query` kueri baru yang dijalankan.
- Jika pemanggilan berisi`NextToken`, pemanggilan tertentu diasumsikan sebagai pemanggilan berikutnya dari panggilan sebelumnya ke KueriAPI, dan kumpulan hasil dikembalikan.
- Setelah 4 jam, permintaan apa pun dengan hal `ClientToken` yang sama diperlakukan sebagai permintaan baru.

Tipe: String

Kendala Panjang: Panjang minimum 32. Panjang maksimum 128.

Wajib: Tidak

MaxRows

Jumlah total baris yang akan dikembalikan dalam `Query` output. Jalankan awal `Query` dengan `MaxRows` nilai yang ditentukan akan mengembalikan kumpulan hasil kueri dalam dua kasus:

- Ukuran hasilnya kurang dari`1MB`.
- Jumlah baris dalam set hasil kurang dari nilai`maxRows`.

Jika tidak, pemanggilan awal `Query` hanya mengembalikan a`NextToken`, yang kemudian dapat digunakan dalam panggilan berikutnya untuk mengambil set hasil. Untuk melanjutkan pagination, berikan `NextToken` nilai dalam perintah berikutnya.

Jika ukuran baris besar (misalnya baris memiliki banyak kolom), Timestream dapat mengembalikan lebih sedikit baris agar ukuran respons tidak melebihi batas 1 MB. Jika tidak `MaxRows` disediakan, Timestream akan mengirimkan jumlah baris yang diperlukan untuk memenuhi batas 1 MB.

Jenis: Integer

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 1000.

Wajib: Tidak

NextToken

Token pagination digunakan untuk mengembalikan serangkaian hasil. Ketika `Query` API dipanggil menggunakan`NextToken`, pemanggilan tertentu diasumsikan sebagai pemanggilan berikutnya dari panggilan sebelumnya ke`Query`, dan kumpulan hasil dikembalikan. Namun, jika `Query` pemanggilan hanya berisi`ClientToken`, pemanggilan tersebut `Query` diasumsikan sebagai kueri baru yang dijalankan.

Perhatikan hal berikut saat menggunakan NextToken dalam kueri:

- Token pagination dapat digunakan hingga lima `Query` pemanggilan, ATAU untuk durasi hingga 1 jam — mana yang lebih dulu.
- Menggunakan yang sama `NextToken` akan mengembalikan set catatan yang sama. Untuk terus melakukan paginasi melalui set hasil, Anda harus menggunakan yang terbaru. `nextToken`
- Misalkan `Query` pemanggilan mengembalikan dua `NextToken` nilai, `TokenA` dan. `TokenB` Jika `TokenB` digunakan dalam `Query` pemanggilan berikutnya, maka tidak valid dan `TokenA` tidak dapat digunakan kembali.

- Untuk meminta set hasil sebelumnya dari kueri setelah pagination dimulai, Anda harus memanggil ulang Query. API
- Yang terbaru `NextToken` harus digunakan untuk paginasi sampai `null` dikembalikan, pada titik mana yang baru `NextToken` harus digunakan.
- Jika IAM prinsipal inisiator kueri dan pembaca hasil tidak sama dan/atau inisiator kueri dan pembaca hasil tidak memiliki string kueri yang sama dalam permintaan kueri, kueri akan gagal dengan kesalahan`Invalid pagination token`.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Tidak

QueryInsights

Merangkum pengaturan untuk mengaktifkan. `QueryInsights`

Mengaktifkan `QueryInsights` mengembalikan wawasan dan metrik selain hasil kueri untuk kueri yang Anda jalankan. Anda dapat menggunakan `QueryInsights` untuk menyetel kinerja kueri Anda.

Tipe: Objek QueryInsights

Wajib: Tidak

QueryString

Query yang akan dijalankan oleh Timestream.

Tipe: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 262144.

Wajib: Ya

Sintaksis Respons

```
{
   "ColumnInfo": [
      {
         "Name": "string",
         "Type": {
            "ArrayColumnInfo": "ColumnInfo",
```

```
            "RowColumnInfo": [
               "ColumnInfo"
            ],
            "ScalarType": "string",
            "TimeSeriesMeasureValueColumnInfo": "ColumnInfo"
         }
      }
   ],
   "NextToken": "string",
   "QueryId": "string",
   "QueryInsightsResponse": {
      "OutputBytes": number,
      "OutputRows": number,
      "QuerySpatialCoverage": {
         "Max": {
            "PartitionKey": [ "string" ],
            "TableArn": "string",
            "Value": number
         }
      },
      "QueryTableCount": number,
      "QueryTemporalRange": {
         "Max": {
            "TableArn": "string",
            "Value": number
         }
      },
      "UnloadPartitionCount": number,
      "UnloadWrittenBytes": number,
      "UnloadWrittenRows": number
   },
   "QueryStatus": {
      "CumulativeBytesMetered": number,
      "CumulativeBytesScanned": number,
      "ProgressPercentage": number
   },
   "Rows": [
      {
         "Data": [
            {
               "ArrayValue": [
                  "Datum"
               ],
               "NullValue": boolean,
```

```
                "RowValue": "Row",
                "ScalarValue": "string",
                "TimeSeriesValue": [
                   {
                      "Time": "string",
                      "Value": "Datum"
                   }
                ]
             }
          ]
       }
    ]
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Data berikut dikembalikan dalam JSON format oleh layanan.

ColumnInfo

Tipe data kolom dari set hasil yang dikembalikan.

Tipe: Array objek ColumnInfo

NextToken

Token pagination yang dapat digunakan lagi pada `Query` panggilan untuk mendapatkan set hasil berikutnya.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

QueryId

ID unik untuk kueri yang diberikan.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `[a-zA-Z0-9]+`

QueryInsightsResponse

Merangkum `QueryInsights` berisi wawasan dan metrik yang terkait dengan kueri yang Anda jalankan.

Tipe: Objek QueryInsightsResponse

QueryStatus

Informasi tentang status kueri, termasuk kemajuan dan byte yang dipindai.

Tipe: Objek QueryStatus

Rows

Hasil set baris dikembalikan oleh query.

Tipe: Array objek Row

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

ConflictException

Tidak dapat melakukan polling hasil untuk kueri yang dibatalkan.

HTTPKode Status: 400

InternalServerException

Layanan tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 400

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

QueryExecutionException

Timestream tidak dapat menjalankan kueri dengan sukses.

HTTPKode Status: 400

ThrottlingException

Permintaan ditolak karena throttling permintaan.

HTTPKode Status: 400

ValidationException

Permintaan tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKpara Ruby V3

TagResource
Layanan: Amazon Timestream Query

Kaitkan satu set tag dengan sumber daya Timestream. Anda kemudian dapat mengaktifkan tag yang ditentukan pengguna ini sehingga muncul di konsol Billing and Cost Management untuk melacak alokasi biaya.

Sintaksis Permintaan

```
{
   "ResourceARN": "string",
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

ResourceARN

Mengidentifikasi sumber daya Timestream yang tag harus ditambahkan. Nilai ini adalah Nama Sumber Daya Amazon (ARN).

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Ya

Tags

Tag yang akan ditetapkan ke sumber daya Timestream.

Tipe: Array objek Tag

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 200 item.

Wajib: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan HTTP tubuh kosong.

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

InvalidEndpointException

Titik akhir yang diminta tidak valid.

HTTPKode Status: 400

ResourceNotFoundException

Sumber daya yang diminta tidak dapat ditemukan.

HTTPKode Status: 400

ServiceQuotaExceededException

Anda telah melampaui kuota layanan.

HTTPKode Status: 400

ThrottlingException

Permintaan ditolak karena throttling permintaan.

HTTPKode Status: 400

ValidationException

Permintaan tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKuntuk Ruby V3

UntagResource
Layanan: Amazon Timestream Query

Menghapus asosiasi tag dari sumber kueri Timestream.

Sintaksis Permintaan

```
{
   "ResourceARN": "string",
   "TagKeys": [ "string" ]
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

ResourceARN

Sumber daya Timestream tempat tag akan dihapus. Nilai ini adalah Nama Sumber Daya Amazon (ARN).

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Ya

TagKeys

Daftar kunci tag. Tag yang ada dari sumber daya yang kuncinya adalah anggota daftar ini akan dihapus dari sumber daya Timestream.

Tipe: Array string.

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 200 item.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Wajib: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan HTTP tubuh kosong.

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

InvalidEndpointException

Titik akhir yang diminta tidak valid.

HTTPKode Status: 400

ResourceNotFoundException

Sumber daya yang diminta tidak dapat ditemukan.

HTTPKode Status: 400

ThrottlingException

Permintaan ditolak karena throttling permintaan.

HTTPKode Status: 400

ValidationException

Permintaan tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2

- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKuntuk Ruby V3

UpdateAccountSettings
Layanan: Amazon Timestream Query

Transisi akun Anda untuk digunakan TCUs untuk harga kueri dan memodifikasi unit komputasi kueri maksimum yang telah Anda konfigurasi. Jika Anda mengurangi nilai `MaxQueryTCU` ke konfigurasi yang diinginkan, nilai baru dapat memakan waktu hingga 24 jam untuk menjadi efektif.

> **Note**
>
> Setelah mentransisikan akun untuk digunakan TCUs untuk harga kueri, Anda tidak dapat beralih menggunakan byte yang dipindai untuk harga kueri.

Sintaksis Permintaan

```
{
   "MaxQueryTCU": number,
   "QueryPricingModel": "string"
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

MaxQueryTCU

Jumlah maksimum unit komputasi yang akan digunakan layanan kapan saja untuk melayani pertanyaan Anda. Untuk menjalankan kueri, Anda harus menetapkan kapasitas minimum 4TCU. Anda dapat mengatur jumlah maksimum TCU dalam kelipatan 4, misalnya, 4, 8, 16, 32, dan seterusnya.

Nilai maksimum yang didukung `MaxQueryTCU` adalah 1000. Untuk meminta peningkatan batas lunak ini, hubungi AWS Support. Untuk informasi tentang kuota default maxQueryTCU, lihat Kuota default.

Tipe: Integer

Wajib: Tidak

QueryPricingModel

Model harga untuk kueri di akun.

> **Note**
>
> `QueryPricingModel`Parameter ini digunakan oleh beberapa operasi Timestream; Namun, `UpdateAccountSettings` API operasi tidak mengenali nilai apa pun selain`COMPUTE_UNITS`.

Tipe: String

Nilai yang Valid: `BYTES_SCANNED | COMPUTE_UNITS`

Wajib: Tidak

Sintaksis Respons

```
{
   "MaxQueryTCU": number,
   "QueryPricingModel": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Data berikut dikembalikan dalam JSON format oleh layanan.

MaxQueryTCU

Jumlah maksimum unit komputasi yang dikonfigurasi yang akan digunakan layanan kapan saja untuk melayani kueri Anda.

Jenis: Integer

QueryPricingModel

Model harga untuk sebuah akun.

Tipe: String

Nilai yang Valid: `BYTES_SCANNED | COMPUTE_UNITS`

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

InternalServerException

Layanan tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 400

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

ThrottlingException

Permintaan ditolak karena throttling permintaan.

HTTPKode Status: 400

ValidationException

Permintaan tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDKuntuk. NET
- AWS SDKuntuk C ++

- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKpara Ruby V3

UpdateScheduledQuery
Layanan: Amazon Timestream Query

Perbarui kueri terjadwal.

Sintaksis Permintaan

```
{
   "ScheduledQueryArn": "string",
   "State": "string"
}
```

Parameter Permintaan

Untuk informasi tentang parameter yang umum untuk semua tindakan, lihat Parameter Umum.

Permintaan menerima data berikut dalam JSON format.

ScheduledQueryArn

ARNdari kueri yang dijadwalkan.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Ya

State

Status kueri yang dijadwalkan.

Tipe: String

Nilai yang Valid: `ENABLED | DISABLED`

Wajib: Ya

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200 dengan HTTP tubuh kosong.

Kesalahan

Untuk informasi tentang kesalahan yang umum untuk semua tindakan, lihat Kesalahan Umum.

AccessDeniedException

Anda tidak berwenang untuk melakukan tindakan ini.

HTTPKode Status: 400

InternalServerException

Layanan tidak dapat sepenuhnya memproses permintaan ini karena kesalahan server internal.

HTTPKode Status: 400

InvalidEndpointException

Endpoint yang diminta tidak valid.

HTTPKode Status: 400

ResourceNotFoundException

Sumber daya yang diminta tidak dapat ditemukan.

HTTPKode Status: 400

ThrottlingException

Permintaan ditolak karena throttling permintaan.

HTTPKode Status: 400

ValidationException

Permintaan tidak valid atau cacat.

HTTPKode Status: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS Antarmuka Baris Perintah

- AWS SDKuntuk. NET
- AWS SDKuntuk C ++
- AWS SDKuntuk Go v2
- AWS SDKuntuk Java V2
- AWS SDKuntuk JavaScript V3
- AWS SDKuntuk PHP V3
- AWS SDKuntuk Python
- AWS SDKpara Ruby V3

Tipe Data

Tipe data berikut ini didukung oleh Amazon Timestream Write:

- BatchLoadProgressReport
- BatchLoadTask
- BatchLoadTaskDescription
- CsvConfiguration
- Database
- DataModel
- DataModelConfiguration
- DataModelS3Configuration
- DataSourceConfiguration
- DataSourceS3Configuration
- Dimension
- DimensionMapping
- Endpoint
- MagneticStoreRejectedDataLocation
- MagneticStoreWriteProperties
- MeasureValue
- MixedMeasureMapping
- MultiMeasureAttributeMapping
- MultiMeasureMappings

- PartitionKey
- Record
- RecordsIngested
- RejectedRecord
- ReportConfiguration
- ReportS3Configuration
- RetentionProperties
- S3Configuration
- Schema
- Table
- Tag

Tipe data berikut didukung oleh Amazon Timestream Query:

- ColumnInfo
- Datum
- DimensionMapping
- Endpoint
- ErrorReportConfiguration
- ErrorReportLocation
- ExecutionStats
- MixedMeasureMapping
- MultiMeasureAttributeMapping
- MultiMeasureMappings
- NotificationConfiguration
- ParameterMapping
- QueryInsights
- QueryInsightsResponse
- QuerySpatialCoverage
- QuerySpatialCoverageMax
- QueryStatus

- QueryTemporalRange
- QueryTemporalRangeMax
- Row
- S3Configuration
- S3ReportLocation
- ScheduleConfiguration
- ScheduledQuery
- ScheduledQueryDescription
- ScheduledQueryInsights
- ScheduledQueryInsightsResponse
- ScheduledQueryRunSummary
- SelectColumn
- SnsConfiguration
- Tag
- TargetConfiguration
- TargetDestination
- TimeSeriesDataPoint
- TimestreamConfiguration
- TimestreamDestination
- Type

Amazon Timestream Menulis

Tipe data berikut ini didukung oleh Amazon Timestream Write:

- BatchLoadProgressReport
- BatchLoadTask
- BatchLoadTaskDescription
- CsvConfiguration
- Database
- DataModel
- DataModelConfiguration

- DataModelS3Configuration
- DataSourceConfiguration
- DataSourceS3Configuration
- Dimension
- DimensionMapping
- Endpoint
- MagneticStoreRejectedDataLocation
- MagneticStoreWriteProperties
- MeasureValue
- MixedMeasureMapping
- MultiMeasureAttributeMapping
- MultiMeasureMappings
- PartitionKey
- Record
- RecordsIngested
- RejectedRecord
- ReportConfiguration
- ReportS3Configuration
- RetentionProperties
- S3Configuration
- Schema
- Table
- Tag

BatchLoadProgressReport

Layanan: Amazon Timestream Write

Detail tentang kemajuan tugas pemuatan batch.

Daftar Isi

BytesMetered

Tipe: Panjang

Wajib: Tidak

FileFailures

Tipe: Panjang

Wajib: Tidak

ParseFailures

Tipe: Panjang

Wajib: Tidak

RecordIngestionFailures

Tipe: Panjang

Wajib: Tidak

RecordsIngested

Tipe: Panjang

Wajib: Tidak

RecordsProcessed

Tipe: Panjang

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

BatchLoadTask

Layanan: Amazon Timestream Write

Detail tentang tugas pemuatan batch.

Daftar Isi

CreationTime

Waktu ketika tugas pemuatan batch Timestream dibuat.

Tipe: Timestamp

Wajib: Tidak

DatabaseName

Nama database untuk database tempat tugas pemuatan batch memuat data.

Tipe: String

Wajib: Tidak

LastUpdatedTime

Waktu ketika tugas pemuatan batch Timestream terakhir diperbarui.

Tipe: Timestamp

Wajib: Tidak

ResumableUntil

Tipe: Timestamp

Wajib: Tidak

TableName

Nama tabel untuk tabel tempat tugas pemuatan batch memuat data.

Tipe: String

Wajib: Tidak

TaskId

ID tugas pemuatan batch.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum sebesar 32.

Pola: `[A-Z0-9]+`

Wajib: Tidak

TaskStatus

Status tugas pemuatan batch.

Tipe: String

Nilai yang Valid: `CREATED | IN_PROGRESS | FAILED | SUCCEEDED | PROGRESS_STOPPED | PENDING_RESUME`

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKuntuk Ruby V3

BatchLoadTaskDescription

Layanan: Amazon Timestream Write

Detail tentang tugas pemuatan batch.

Daftar Isi

CreationTime

Waktu ketika tugas pemuatan batch Timestream dibuat.

Tipe: Timestamp

Wajib: Tidak

DataModelConfiguration

Konfigurasi model data untuk tugas pemuatan batch. Ini berisi rincian tentang di mana model data untuk tugas pemuatan batch disimpan.

Tipe: Objek DataModelConfiguration

Wajib: Tidak

DataSourceConfiguration

Detail konfigurasi tentang sumber data untuk tugas pemuatan batch.

Tipe: Objek DataSourceConfiguration

Wajib: Tidak

ErrorMessage

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Tidak

LastUpdatedTime

Waktu ketika tugas pemuatan batch Timestream terakhir diperbarui.

Tipe: Timestamp

Wajib: Tidak

ProgressReport

Tipe: Objek BatchLoadProgressReport

Wajib: Tidak

RecordVersion

Tipe: Panjang

Wajib: Tidak

ReportConfiguration

Laporkan konfigurasi untuk tugas pemuatan batch. Ini berisi rincian tentang di mana laporan kesalahan disimpan.

Tipe: Objek ReportConfiguration

Wajib: Tidak

ResumableUntil

Tipe: Timestamp

Wajib: Tidak

TargetDatabaseName

Tipe: String

Wajib: Tidak

TargetTableName

Tipe: String

Wajib: Tidak

TaskId

ID tugas pemuatan batch.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum sebesar 32.

Pola: `[A-Z0-9]+`

Wajib: Tidak

TaskStatus

Status tugas pemuatan batch.

Tipe: String

Nilai yang Valid: `CREATED | IN_PROGRESS | FAILED | SUCCEEDED | PROGRESS_STOPPED | PENDING_RESUME`

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

CsvConfiguration
Layanan: Amazon Timestream Write

Format data yang dibatasi di mana pemisah kolom dapat berupa koma dan pemisah catatan adalah karakter baris baru.

Daftar Isi

ColumnSeparator

Pemisah kolom dapat berupa koma (','), pipa ('|), titik koma (';'), tab ('/t'), atau ruang kosong (").

Tipe: String

Kendala Panjang: Panjang tetap 1.

Wajib: Tidak

EscapeChar

Karakter melarikan diri bisa menjadi salah satu

Tipe: String

Kendala Panjang: Panjang tetap 1.

Wajib: Tidak

NullValue

Bisa berupa ruang kosong (").

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 256.

Wajib: Tidak

QuoteChar

Bisa berupa kutipan tunggal (') atau kutipan ganda (“).

Tipe: String

Kendala Panjang: Panjang tetap 1.

Wajib: Tidak

TrimWhiteSpace

Menentukan untuk memangkas ruang putih terkemuka dan membuntuti.

Tipe: Boolean

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKuntuk Ruby V3

Database

Layanan: Amazon Timestream Write

Wadah tingkat atas untuk sebuah meja. Database dan tabel adalah konsep manajemen dasar di Amazon Timestream. Semua tabel dalam database dienkripsi dengan kunci yang sama AWS KMS .

Daftar Isi

Arn

Nama Sumber Daya Amazon yang secara unik mengidentifikasi database ini.

Tipe: String

Wajib: Tidak

CreationTime

Waktu ketika database dibuat, dihitung dari waktu epoch Unix.

Tipe: Timestamp

Wajib: Tidak

DatabaseName

Nama basis data Timestream.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 256.

Wajib: Tidak

KmsKeyId

Pengidentifikasi AWS KMS kunci yang digunakan untuk mengenkripsi data yang disimpan dalam database.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Tidak

LastUpdatedTime

Terakhir kali database ini diperbarui.

Tipe: Timestamp

Wajib: Tidak

TableCount

Jumlah total tabel yang ditemukan dalam database Timestream.

Tipe: Panjang

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

DataModel

Layanan: Amazon Timestream Write

Model data untuk tugas pemuatan batch.

Daftar Isi

DimensionMappings

Sumber untuk menargetkan pemetaan untuk dimensi.

Tipe: Array objek DimensionMapping

Anggota Array: Jumlah minimum 1 item.

Wajib: Ya

MeasureNameColumn

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 256.

Wajib: Tidak

MixedMeasureMappings

Sumber untuk menargetkan pemetaan untuk tindakan.

Tipe: Array objek MixedMeasureMapping

Anggota Array: Jumlah minimum 1 item.

Wajib: Tidak

MultiMeasureMappings

Sumber untuk menargetkan pemetaan untuk catatan multi-ukuran.

Tipe: Objek MultiMeasureMappings

Wajib: Tidak

TimeColumn

Kolom sumber untuk dipetakan ke waktu.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 256.

Wajib: Tidak

TimeUnit

Perincian unit stempel waktu. Ini menunjukkan apakah nilai waktu dalam detik, milidetik, nanodetik, atau nilai lain yang didukung. Default-nya adalah `MILLISECONDS`.

Tipe: String

Nilai yang Valid: `MILLISECONDS | SECONDS | MICROSECONDS | NANOSECONDS`

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

DataModelConfiguration
Layanan: Amazon Timestream Write

Daftar Isi

DataModel

Tipe: Objek DataModel

Wajib: Tidak

DataModelS3Configuration

Tipe: Objek DataModelS3Configuration

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

DataModelS3Configuration
Layanan: Amazon Timestream Write

Daftar Isi

BucketName

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 63.

Pola: `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Wajib: Tidak

ObjectKey

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 1024.

Pola: `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

DataSourceConfiguration
Layanan: Amazon Timestream Write

Mendefinisikan detail konfigurasi tentang sumber data.

Daftar Isi

DataFormat

Ini saat iniCSV.

Tipe: String

Nilai yang Valid: CSV

Wajib: Ya

DataSourceS3Configuration

Konfigurasi lokasi S3 untuk file yang berisi data untuk dimuat.

Tipe: Objek DataSourceS3Configuration

Wajib: Ya

CsvConfiguration

Format data yang dibatasi di mana pemisah kolom dapat berupa koma dan pemisah catatan adalah karakter baris baru.

Tipe: Objek CsvConfiguration

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

DataSourceS3Configuration
Layanan: Amazon Timestream Write

Daftar Isi

BucketName

Nama bucket bucket S3 pelanggan.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 63.

Pola: `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Wajib: Ya

ObjectKeyPrefix

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 1024.

Pola: `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

Dimension
Layanan: Amazon Timestream Write

Merupakan atribut metadata dari deret waktu. Misalnya, nama dan Availability Zone dari sebuah EC2 instance atau nama produsen turbin angin adalah dimensi.

Daftar Isi

Name

Dimensi mewakili atribut metadata dari deret waktu. Misalnya, nama dan Availability Zone dari sebuah EC2 instance atau nama produsen turbin angin adalah dimensi.

Untuk batasan pada nama dimensi, lihat Kendala Penamaan.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 60.

Wajib: Ya

Value

Nilai dari dimensi.

Tipe: String

Diperlukan: Ya

DimensionValueType

Tipe data dimensi untuk titik data deret waktu.

Tipe: String

Nilai yang Valid: `VARCHAR`

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

DimensionMapping
Layanan: Amazon Timestream Write

Daftar Isi

DestinationColumn

Tipe: String

Panjang Batasan: Panjang minimum 1.

Wajib: Tidak

SourceColumn

Tipe: String

Panjang Batasan: Panjang minimum 1.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

Endpoint
Layanan: Amazon Timestream Write

Merupakan titik akhir yang tersedia untuk melakukan API panggilan, serta TTL untuk titik akhir itu.

Daftar Isi

Address

Alamat titik akhir.

Tipe: String

Diperlukan: Ya

CachePeriodInMinutes

TTLUntuk titik akhir, dalam hitungan menit.

Tipe: Long

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

MagneticStoreRejectedDataLocation
Layanan: Amazon Timestream Write

Lokasi untuk menulis laporan kesalahan untuk catatan ditolak, secara asinkron, selama penyimpanan magnetik menulis.

Daftar Isi

S3Configuration

Konfigurasi lokasi S3 untuk menulis laporan kesalahan untuk catatan yang ditolak, secara asinkron, selama penulisan penyimpanan magnetik.

Tipe: Objek S3Configuration

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKuntuk Ruby V3

MagneticStoreWriteProperties
Layanan: Amazon Timestream Write

Kumpulan properti di atas meja untuk mengkonfigurasi penyimpanan magnetik menulis.

Daftar Isi

EnableMagneticStoreWrites

Bendera untuk mengaktifkan penulisan penyimpanan magnetik.

Jenis: Boolean

Wajib: Ya

MagneticStoreRejectedDataLocation

Lokasi untuk menulis laporan kesalahan untuk catatan ditolak secara asinkron selama penulisan penyimpanan magnetik.

Tipe: Objek MagneticStoreRejectedDataLocation

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

MeasureValue
Layanan: Amazon Timestream Write

Merupakan atribut data dari deret waktu. Misalnya, CPU penggunaan EC2 instance atau turbin angin adalah ukuran. RPM MeasureValue memiliki nama dan nilai.

MeasureValue hanya diperbolehkan untuk tipe`MULTI`. Menggunakan `MULTI` tipe, Anda dapat meneruskan beberapa atribut data yang terkait dengan deret waktu yang sama dalam satu catatan

Daftar Isi

Name

Nama dari MeasureValue.

Untuk batasan pada MeasureValue nama, lihat Kendala Penamaan di Panduan Pengembang Amazon Timestream.

Tipe: String

Batasan Panjang: Panjang minimum 1.

Wajib: Ya

Type

Berisi tipe data MeasureValue untuk titik data deret waktu.

Tipe: String

Nilai yang Valid: `DOUBLE | BIGINT | VARCHAR | BOOLEAN | TIMESTAMP | MULTI`

Wajib: Ya

Value

Nilai untuk MeasureValue. Untuk selengkapnya, lihat Tipe data.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKuntuk Ruby V3

MixedMeasureMapping
Layanan: Amazon Timestream Write

Daftar Isi

MeasureValueType

Tipe: String

Nilai yang Valid: `DOUBLE | BIGINT | VARCHAR | BOOLEAN | TIMESTAMP | MULTI`

Wajib: Ya

MeasureName

Tipe: String

Panjang Batasan: Panjang minimum 1.

Wajib: Tidak

MultiMeasureAttributeMappings

Tipe: Array objek MultiMeasureAttributeMapping

Anggota Array: Jumlah minimum 1 item.

Wajib: Tidak

SourceColumn

Tipe: String

Panjang Batasan: Panjang minimum 1.

Wajib: Tidak

TargetMeasureName

Tipe: String

Panjang Batasan: Panjang minimum 1.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

MultiMeasureAttributeMapping
Layanan: Amazon Timestream Write

Daftar Isi

SourceColumn

Tipe: String

Batasan Panjang: Panjang minimum 1.

Wajib: Ya

MeasureValueType

Tipe: String

Nilai yang Valid: `DOUBLE | BIGINT | BOOLEAN | VARCHAR | TIMESTAMP`

Wajib: Tidak

TargetMultiMeasureAttributeName

Tipe: String

Panjang Batasan: Panjang minimum 1.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

MultiMeasureMappings
Layanan: Amazon Timestream Write

Daftar Isi

MultiMeasureAttributeMappings

Tipe: Array objek MultiMeasureAttributeMapping

Anggota Array: Jumlah minimum 1 item.

Wajib: Ya

TargetMultiMeasureName

Tipe: String

Panjang Batasan: Panjang minimum 1.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

PartitionKey
Layanan: Amazon Timestream Write

Atribut yang digunakan dalam partisi data dalam tabel. Kunci dimensi mempartisi data menggunakan nilai dimensi yang ditentukan oleh nama dimensi sebagai kunci partisi, sementara kunci ukuran mempartisi data menggunakan nama ukuran (nilai kolom 'ukur_nama').

Daftar Isi

Type

Jenis kunci partisi. Pilihannya adalah DIMENSION (kunci dimensi) dan MEASURE (tombol ukur).

Tipe: String

Nilai yang Valid: `DIMENSION | MEASURE`

Wajib: Ya

EnforcementInRecord

Tingkat penegakan untuk spesifikasi kunci dimensi dalam catatan yang dicerna. Opsi adalah REQUIRED (kunci dimensi harus ditentukan) dan OPTIONAL (kunci dimensi tidak harus ditentukan).

Tipe: String

Nilai yang Valid: `REQUIRED | OPTIONAL`

Wajib: Tidak

Name

Nama atribut yang digunakan untuk kunci dimensi.

Tipe: String

Panjang Batasan: Panjang minimum 1.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

Record

Layanan: Amazon Timestream Write

Merupakan titik data deret waktu yang ditulis ke Timestream. Setiap catatan berisi berbagai dimensi. Dimensi mewakili atribut metadata dari titik data deret waktu, seperti nama instance atau Availability Zone dari sebuah instance. EC2 Catatan juga berisi nama ukuran, yang merupakan nama ukuran yang dikumpulkan (misalnya, CPU pemanfaatan EC2 instance). Selain itu, catatan berisi nilai ukuran dan tipe nilai, yang merupakan tipe data dari nilai ukuran. Juga, catatan berisi stempel waktu kapan ukuran dikumpulkan dan unit stempel waktu, yang mewakili granularitas stempel waktu.

Catatan memiliki `Version` bidang, yang merupakan 64-bit `long` yang dapat Anda gunakan untuk memperbarui titik data. Menulis catatan duplikat dengan dimensi, stempel waktu, dan nama ukuran yang sama tetapi nilai ukuran yang berbeda hanya akan berhasil jika `Version` atribut catatan dalam permintaan tulis lebih tinggi dari catatan yang ada. Timestream default ke `Version` of `1` untuk catatan tanpa bidang. `Version`

Daftar Isi

Dimensions

Berisi daftar dimensi untuk titik data deret waktu.

Tipe: Array objek Dimension

Anggota Array: Jumlah maksimum 128 item.

Wajib: Tidak

MeasureName

Ukuran mewakili atribut data dari deret waktu. Misalnya, CPU penggunaan EC2 instance atau turbin angin adalah ukuran. RPM

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 256.

Wajib: Tidak

MeasureValue

Berisi nilai ukuran untuk titik data deret waktu.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Tidak

MeasureValues

Berisi daftar MeasureValue untuk titik data deret waktu.

Ini hanya diperbolehkan untuk tipe`MULTI`. Untuk nilai skalar, gunakan `MeasureValue` atribut catatan secara langsung.

Tipe: Array objek MeasureValue

Wajib: Tidak

MeasureValueType

Berisi tipe data dari nilai ukuran untuk titik data deret waktu. Jenis default adalah`DOUBLE`. Untuk informasi selengkapnya, lihat Tipe data.

Tipe: String

Nilai yang Valid: `DOUBLE | BIGINT | VARCHAR | BOOLEAN | TIMESTAMP | MULTI`

Wajib: Tidak

Time

Berisi waktu di mana nilai ukuran untuk titik data dikumpulkan. Nilai waktu ditambah unit menyediakan waktu yang berlalu sejak zaman. Misalnya, jika nilai waktu `12345` dan satuannya`ms`, maka `12345 ms` telah berlalu sejak zaman.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 256.

Wajib: Tidak

TimeUnit

Perincian unit stempel waktu. Ini menunjukkan apakah nilai waktu dalam detik, milidetik, nanodetik, atau nilai lain yang didukung. Default-nya adalah `MILLISECONDS`.

Tipe: String

Nilai yang Valid: `MILLISECONDS | SECONDS | MICROSECONDS | NANOSECONDS`

Wajib: Tidak

Version

Atribut 64-bit digunakan untuk pembaruan rekaman. Menulis permintaan untuk data duplikat dengan nomor versi yang lebih tinggi akan memperbarui nilai ukuran dan versi yang ada. Dalam kasus di mana nilai ukurannya sama, masih `Version` akan diperbarui. Nilai default-nya adalah 1.

> **Note**
>
> `Version`harus 1 atau lebih besar, atau Anda akan menerima `ValidationException` kesalahan.

Tipe: Panjang

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

RecordsIngested
Layanan: Amazon Timestream Write

Informasi tentang catatan yang dicerna oleh permintaan ini.

Daftar Isi

MagneticStore

Hitungan catatan yang tertelan ke dalam toko magnetik.

Tipe: Integer

Wajib: Tidak

MemoryStore

Hitungan catatan yang dicerna ke dalam penyimpanan memori.

Tipe: Integer

Wajib: Tidak

Total

Jumlah total catatan yang berhasil dicerna.

Tipe: Integer

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKuntuk Ruby V3

RejectedRecord
Layanan: Amazon Timestream Write

Merupakan catatan yang tidak berhasil dimasukkan ke Timestream karena masalah validasi data yang harus diselesaikan sebelum memasukkan kembali data deret waktu ke dalam sistem.

Daftar Isi

ExistingVersion

Versi rekaman yang ada. Nilai ini diisi dalam skenario di mana catatan identik ada dengan versi yang lebih tinggi dari versi dalam permintaan tulis.

Tipe: Panjang

Wajib: Tidak

Reason

Alasan mengapa catatan tidak berhasil dimasukkan ke Timestream. Kemungkinan penyebab kegagalan meliputi:

- Rekaman dengan data duplikat di mana ada beberapa catatan dengan dimensi, stempel waktu, dan nama ukuran yang sama tetapi:
  - Nilai ukur berbeda
  - Versi tidak hadir dalam permintaan, atau nilai versi dalam catatan baru sama dengan atau lebih rendah dari nilai yang ada

  Jika Timestream menolak data untuk kasus ini, `ExistingVersion` bidang dalam `RejectedRecords` respons akan menunjukkan versi rekaman saat ini. Untuk memaksa pembaruan, Anda dapat mengirim ulang permintaan dengan versi untuk catatan yang ditetapkan ke nilai yang lebih besar dari`ExistingVersion`.
- Catatan dengan stempel waktu yang berada di luar durasi retensi penyimpanan memori.

> **Note**
>
> Ketika jendela retensi diperbarui, Anda akan menerima `RejectedRecords` pengecualian jika Anda segera mencoba untuk menelan data dalam jendela baru. Untuk menghindari `RejectedRecords` pengecualian, tunggu hingga durasi jendela baru untuk menelan data baru. Untuk informasi lebih lanjut, lihat Praktik Terbaik untuk

Mengkonfigurasi Timestream dan penjelasan tentang cara kerja penyimpanan di Timestream.

- Rekaman dengan dimensi atau ukuran yang melebihi batas yang ditentukan Timestream.

Untuk informasi selengkapnya, lihat Manajemen Akses di Panduan Pengembang Timestream.

Tipe: String

Wajib: Tidak

RecordIndex

Indeks catatan dalam permintaan input untuk WriteRecords. Indeks dimulai dengan 0.

Tipe: Integer

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

ReportConfiguration
Layanan: Amazon Timestream Write

Laporkan konfigurasi untuk tugas pemuatan batch. Ini berisi rincian tentang di mana laporan kesalahan disimpan.

Daftar Isi

ReportS3Configuration

Konfigurasi lokasi S3 untuk menulis laporan kesalahan dan peristiwa untuk pemuatan batch.

Tipe: Objek ReportS3Configuration

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

ReportS3Configuration
Layanan: Amazon Timestream Write

Daftar Isi

BucketName

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 63.

Pola: `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Wajib: Ya

EncryptionOption

Tipe: String

Nilai yang Valid: `SSE_S3 | SSE_KMS`

Wajib: Tidak

KmsKeyId

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Tidak

ObjectKeyPrefix

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 928.

Pola: `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

RetentionProperties
Layanan: Amazon Timestream Write

Properti retensi berisi durasi di mana data deret waktu Anda harus disimpan di penyimpanan magnetik dan penyimpanan memori.

Daftar Isi

MagneticStoreRetentionPeriodInDays

Durasi data harus disimpan di penyimpanan magnetik.

Tipe: Long

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 73000.

Wajib: Ya

MemoryStoreRetentionPeriodInHours

Durasi data harus disimpan di penyimpanan memori.

Tipe: Long

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 8766.

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKuntuk Ruby V3

S3Configuration

Layanan: Amazon Timestream Write

Konfigurasi yang menentukan lokasi S3.

Daftar Isi

BucketName

Nama bucket bucket S3 pelanggan.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 63.

Pola: `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Wajib: Tidak

EncryptionOption

Opsi enkripsi untuk lokasi S3 pelanggan. Opsi adalah enkripsi sisi server S3 dengan kunci terkelola S3 atau kunci terkelola. AWS

Tipe: String

Nilai yang Valid: `SSE_S3 | SSE_KMS`

Wajib: Tidak

KmsKeyId

ID AWS KMS kunci untuk lokasi S3 pelanggan saat mengenkripsi dengan kunci terkelola AWS .

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Tidak

ObjectKeyPrefix

Pratinjau kunci objek untuk lokasi S3 pelanggan.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 928.

Pola: `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKuntuk Ruby V3

Schema

Layanan: Amazon Timestream Write

Skema menentukan model data yang diharapkan dari tabel.

Daftar Isi

CompositePartitionKey

Daftar kunci partisi yang tidak kosong yang mendefinisikan atribut yang digunakan untuk mempartisi data tabel. Urutan daftar menentukan hierarki partisi. Nama dan jenis setiap kunci partisi serta urutan kunci partisi tidak dapat diubah setelah tabel dibuat. Namun, tingkat penegakan setiap kunci partisi dapat diubah.

Tipe: Array objek PartitionKey

Anggota Array: Jumlah minimum 1 item.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

Table

Layanan: Amazon Timestream Write

Merupakan tabel database di Timestream. Tabel berisi satu atau lebih deret waktu terkait. Anda dapat memodifikasi durasi retensi penyimpanan memori dan penyimpanan magnetik untuk sebuah tabel.

Daftar Isi

Arn

Nama Sumber Daya Amazon yang secara unik mengidentifikasi tabel ini.

Tipe: String

Wajib: Tidak

CreationTime

Waktu ketika tabel Timestream dibuat.

Tipe: Timestamp

Wajib: Tidak

DatabaseName

Nama basis data Timestream yang berisi tabel ini.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 256.

Wajib: Tidak

LastUpdatedTime

Waktu ketika tabel Timestream terakhir diperbarui.

Tipe: Timestamp

Wajib: Tidak

MagneticStoreWriteProperties

Berisi properti untuk diatur di atas meja saat mengaktifkan penulisan penyimpanan magnetik.

Tipe: Objek MagneticStoreWriteProperties

Wajib: Tidak

RetentionProperties

Durasi retensi untuk penyimpanan memori dan penyimpanan magnetis.

Tipe: Objek RetentionProperties

Wajib: Tidak

Schema

Skema tabel.

Tipe: Objek Schema

Wajib: Tidak

TableName

Nama tabel Timestream.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 256.

Wajib: Tidak

TableStatus

Keadaan tabel saat ini:

- `DELETING`- Tabel sedang dihapus.
- `ACTIVE`- Meja siap digunakan.

Tipe: String

Nilai yang Valid: `ACTIVE | DELETING | RESTORING`

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

Tag
Layanan: Amazon Timestream Write

Tag adalah label yang Anda tetapkan ke and/or table. Each tag consists of a key and an optional value, both of which you define. With tags, you can categorize databases and/or tabel database Timestream, misalnya, berdasarkan tujuan, pemilik, atau lingkungan.

Daftar Isi

Key

Kunci tag. Kunci tag peka huruf besar dan kecil.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Wajib: Ya

Value

Nilai tag. Nilai tag peka huruf besar/kecil dan bisa null.

Tipe: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 256.

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

Kueri Amazon Timestream

Tipe data berikut ini didukung oleh Amazon Timestream Query:

- ColumnInfo
- Datum
- DimensionMapping
- Endpoint
- ErrorReportConfiguration
- ErrorReportLocation
- ExecutionStats
- MixedMeasureMapping
- MultiMeasureAttributeMapping
- MultiMeasureMappings
- NotificationConfiguration
- ParameterMapping
- QueryInsights
- QueryInsightsResponse
- QuerySpatialCoverage
- QuerySpatialCoverageMax
- QueryStatus
- QueryTemporalRange
- QueryTemporalRangeMax
- Row
- S3Configuration
- S3ReportLocation
- ScheduleConfiguration
- ScheduledQuery
- ScheduledQueryDescription
- ScheduledQueryInsights
- ScheduledQueryInsightsResponse
- ScheduledQueryRunSummary
- SelectColumn
- SnsConfiguration

- Tag
- TargetConfiguration
- TargetDestination
- TimeSeriesDataPoint
- TimestreamConfiguration
- TimestreamDestination
- Type

ColumnInfo
Layanan: Amazon Timestream Query

Berisi metadata untuk hasil kueri seperti nama kolom, tipe data, dan atribut lainnya.

Daftar Isi

Type

Tipe data dari kolom set hasil. Tipe data dapat berupa skalar atau kompleks. Jenis data skalar adalah bilangan bulat, string, ganda, Boolean, dan lainnya. Tipe data yang kompleks adalah tipe seperti array, baris, dan lainnya.

Tipe: Objek Type

Wajib: Ya

Name

Nama kolom set hasil. Nama set hasil tersedia untuk kolom dari semua tipe data kecuali untuk array.

Tipe: String

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

Datum

Layanan: Amazon Timestream Query

Datum merupakan titik data tunggal dalam hasil query.

Daftar Isi

ArrayValue

Menunjukkan jika titik data adalah array.

Tipe: Array objek Datum

Wajib: Tidak

NullValue

Menunjukkan jika titik data adalah nol.

Tipe: Boolean

Wajib: Tidak

RowValue

Menunjukkan jika titik data adalah baris.

Tipe: Objek Row

Wajib: Tidak

ScalarValue

Menunjukkan jika titik data adalah nilai skalar seperti integer, string, double, atau Boolean.

Tipe: String

Wajib: Tidak

TimeSeriesValue

Menunjukkan jika titik data adalah tipe data timeseries.

Tipe: Array objek TimeSeriesDataPoint

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

DimensionMapping
Layanan: Amazon Timestream Query

Jenis ini digunakan untuk memetakan kolom dari hasil kueri ke dimensi dalam tabel tujuan.

Daftar Isi

DimensionValueType

Ketik untuk dimensi.

Tipe: String

Nilai yang Valid: `VARCHAR`

Wajib: Ya

Name

Nama kolom dari hasil kueri.

Tipe: String

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

Endpoint
Layanan: Amazon Timestream Query

Merupakan titik akhir yang tersedia untuk melakukan API panggilan, serta TTL untuk titik akhir itu.

Daftar Isi

Address

Alamat titik akhir.

Tipe: String

Diperlukan: Ya

CachePeriodInMinutes

TTLUntuk titik akhir, dalam hitungan menit.

Tipe: Long

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

ErrorReportConfiguration
Layanan: Amazon Timestream Query

Konfigurasi diperlukan untuk pelaporan kesalahan.

Daftar Isi

S3Configuration

Konfigurasi S3 untuk laporan kesalahan.

Tipe: Objek S3Configuration

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

ErrorReportLocation
Layanan: Amazon Timestream Query

Ini berisi lokasi laporan kesalahan untuk satu panggilan kueri terjadwal.

Daftar Isi

S3ReportLocation

Lokasi S3 tempat laporan kesalahan ditulis.

Tipe: Objek S3ReportLocation

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

ExecutionStats

Layanan: Amazon Timestream Query

Statistik untuk menjalankan kueri terjadwal tunggal.

Daftar Isi

BytesMetered

Byte diukur untuk menjalankan kueri terjadwal tunggal.

Tipe: Panjang

Wajib: Tidak

CumulativeBytesScanned

Byte dipindai untuk menjalankan kueri terjadwal tunggal.

Tipe: Panjang

Wajib: Tidak

DataWrites

Data menulis diukur untuk catatan yang dicerna dalam satu proses kueri terjadwal.

Tipe: Panjang

Wajib: Tidak

ExecutionTimeInMillis

Total waktu, diukur dalam milidetik, yang diperlukan agar proses kueri terjadwal selesai.

Tipe: Panjang

Wajib: Tidak

QueryResultRows

Jumlah baris yang ada dalam output dari menjalankan kueri sebelum konsumsi ke sumber data tujuan.

Tipe: Panjang

Wajib: Tidak

RecordsIngested

Jumlah rekaman yang dicerna untuk menjalankan kueri terjadwal tunggal.

Tipe: Panjang

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

MixedMeasureMapping
Layanan: Amazon Timestream Query

MixedMeasureMappings adalah pemetaan yang dapat digunakan untuk menelan data ke dalam campuran ukuran sempit dan multi dalam tabel turunan.

Daftar Isi

MeasureValueType

Jenis nilai yang akan dibacasourceColumn. Jika pemetaan adalah untukMULTI, gunakan MeasureValueType. MULTI.

Tipe: String

Nilai yang Valid: `BIGINT | BOOLEAN | DOUBLE | VARCHAR | MULTI`

Wajib: Ya

MeasureName

Mengacu pada nilai ukuran_name di baris hasil. Bidang ini diperlukan jika MeasureNameColumn disediakan.

Tipe: String

Wajib: Tidak

MultiMeasureAttributeMappings

Diperlukan kapan measureValueType MULTI. Pemetaan atribut untuk ukuran MULTI nilai.

Tipe: Array objek MultiMeasureAttributeMapping

Anggota Array: Jumlah minimum 1 item.

Wajib: Tidak

SourceColumn

Bidang ini mengacu pada kolom sumber dari mana nilai pengukuran harus dibaca untuk perwujudan hasil.

Tipe: String

Wajib: Tidak

TargetMeasureName

Nama ukuran target yang akan digunakan. Jika tidak disediakan, nama ukuran target secara default akan menjadi nama ukuran jika disediakan, atau sourceColumn sebaliknya.

Tipe: String

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

MultiMeasureAttributeMapping
Layanan: Amazon Timestream Query

Pemetaan atribut untuk ukuran MULTI nilai.

Daftar Isi

MeasureValueType

Jenis atribut yang akan dibaca dari kolom sumber.

Tipe: String

Nilai yang Valid: `BIGINT | BOOLEAN | DOUBLE | VARCHAR | TIMESTAMP`

Wajib: Ya

SourceColumn

Kolom sumber dari mana nilai atribut akan dibaca.

Tipe: String

Diperlukan: Ya

TargetMultiMeasureAttributeName

Nama kustom yang akan digunakan untuk nama atribut dalam tabel turunan. Jika tidak disediakan, nama kolom sumber akan digunakan.

Tipe: String

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKuntuk Ruby V3

MultiMeasureMappings
Layanan: Amazon Timestream Query

Hanya satu dari MixedMeasureMappings atau MultiMeasureMappings yang akan disediakan. MultiMeasureMappings dapat digunakan untuk menelan data sebagai multi ukuran dalam tabel turunan.

Daftar Isi

MultiMeasureAttributeMappings

Wajib. Pemetaan atribut yang akan digunakan untuk memetakan hasil kueri untuk mencerna data untuk atribut multi-ukuran.

Tipe: Array objek MultiMeasureAttributeMapping

Anggota Array: Jumlah minimum 1 item.

Wajib: Ya

TargetMultiMeasureName

Nama nama multi-ukuran target dalam tabel turunan. Input ini measureNameColumn diperlukan ketika tidak disediakan. Jika MeasureNameColumn disediakan, maka nilai dari kolom itu akan digunakan sebagai nama multi-ukuran.

Tipe: String

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

NotificationConfiguration
Layanan: Amazon Timestream Query

Konfigurasi pemberitahuan untuk kueri terjadwal. Pemberitahuan dikirim oleh Timestream ketika kueri terjadwal dibuat, statusnya diperbarui atau ketika dihapus.

Daftar Isi

SnsConfiguration

Detail tentang SNS konfigurasi.

Tipe: Objek SnsConfiguration

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

ParameterMapping
Layanan: Amazon Timestream Query

Pemetaan untuk parameter bernama.

Daftar Isi

Name

Nama parameter.

Tipe: String

Diperlukan: Ya

Type

Berisi tipe data kolom dalam kumpulan hasil kueri. Tipe data bisa skalar atau kompleks. Tipe data skalar yang didukung adalah bilangan bulat, Boolean, string, ganda, stempel waktu, tanggal, waktu, dan interval. Tipe data kompleks yang didukung adalah array, baris, dan timeseries.

Tipe: Objek Type

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

QueryInsights
Layanan: Amazon Timestream Query

`QueryInsights`adalah fitur penyetelan kinerja yang membantu Anda mengoptimalkan kueri, mengurangi biaya, dan meningkatkan kinerja. Dengan`QueryInsights`, Anda dapat menilai efisiensi pemangkasan kueri Anda dan mengidentifikasi area untuk perbaikan guna meningkatkan kinerja kueri. Dengan`QueryInsights`, Anda juga dapat menganalisis efektivitas kueri Anda dalam hal pemangkasan temporal dan spasial, dan mengidentifikasi peluang untuk meningkatkan kinerja. Secara khusus, Anda dapat mengevaluasi seberapa baik kueri Anda menggunakan strategi pengindeksan berbasis waktu dan partisi berbasis kunci untuk mengoptimalkan pengambilan data. Untuk mengoptimalkan kinerja kueri, penting bagi Anda untuk menyempurnakan parameter temporal dan spasial yang mengatur eksekusi kueri.

Metrik utama yang disediakan oleh `QueryInsights` adalah `QuerySpatialCoverage` dan`QueryTemporalRange`. `QuerySpatialCoverage`menunjukkan seberapa banyak sumbu spasial yang dipindai kueri, dengan nilai yang lebih rendah menjadi lebih efisien. `QueryTemporalRange`menunjukkan rentang waktu yang dipindai, dengan rentang yang lebih sempit menjadi lebih berkinerja.

Manfaat dari QueryInsights

Berikut ini adalah manfaat utama menggunakan`QueryInsights`:

- Mengidentifikasi kueri yang tidak efisien — `QueryInsights` memberikan informasi tentang pemangkasan tabel berbasis waktu dan atribut yang diakses oleh kueri. Informasi ini membantu Anda mengidentifikasi tabel yang diakses secara sub-optimal.
- Mengoptimalkan model data dan partisi Anda — Anda dapat menggunakan `QueryInsights` informasi untuk mengakses dan menyempurnakan model data dan strategi partisi Anda.
- Tuning query — `QueryInsights` menyoroti peluang untuk menggunakan indeks secara lebih efektif.

> **Note**
>
> Jumlah maksimum `Query` API permintaan yang diizinkan untuk Anda buat dengan `QueryInsights` diaktifkan adalah 1 kueri per detik (QPS). Jika Anda melebihi tingkat kueri ini, itu mungkin mengakibatkan pelambatan.

Daftar Isi

Mode

Menyediakan mode berikut untuk mengaktifkan`QueryInsights`:

- `ENABLED_WITH_RATE_CONTROL`— `QueryInsights` Memungkinkan kueri yang sedang diproses. Mode ini juga mencakup mekanisme kontrol laju, yang membatasi `QueryInsights` fitur hingga 1 kueri per detik (QPS).
- `DISABLED`— Menonaktifkan`QueryInsights`.

Tipe: String

Nilai yang Valid: `ENABLED_WITH_RATE_CONTROL | DISABLED`

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKuntuk Ruby V3

QueryInsightsResponse
Layanan: Amazon Timestream Query

Memberikan berbagai wawasan dan metrik yang terkait dengan kueri yang Anda jalankan.

Daftar Isi

OutputBytes

Menunjukkan ukuran hasil kueri yang diatur dalam byte. Anda dapat menggunakan data ini untuk memvalidasi jika kumpulan hasil telah berubah sebagai bagian dari latihan penyetelan kueri.

Tipe: Panjang

Wajib: Tidak

OutputRows

Menunjukkan jumlah total baris yang dikembalikan sebagai bagian dari set hasil kueri. Anda dapat menggunakan data ini untuk memvalidasi jika jumlah baris dalam kumpulan hasil telah berubah sebagai bagian dari latihan penyetelan kueri.

Tipe: Panjang

Wajib: Tidak

QuerySpatialCoverage

Memberikan wawasan tentang cakupan spasial kueri, termasuk tabel dengan pemangkasan spasial sub-optimal (maks). Informasi ini dapat membantu Anda mengidentifikasi area untuk perbaikan dalam strategi partisi Anda untuk meningkatkan pemangkasan spasial.

Tipe: Objek QuerySpatialCoverage

Wajib: Tidak

QueryTableCount

Menunjukkan jumlah tabel dalam kueri.

Tipe: Panjang

Wajib: Tidak

QueryTemporalRange

Memberikan wawasan tentang rentang temporal kueri, termasuk tabel dengan rentang waktu (maks) terbesar. Berikut adalah beberapa opsi potensial untuk mengoptimalkan pemangkasan berbasis waktu:

- Tambahkan predikat waktu yang hilang.
- Hapus fungsi di sekitar predikat waktu.
- Tambahkan predikat waktu ke semua sub-kueri.

Tipe: Objek QueryTemporalRange

Wajib: Tidak

UnloadPartitionCount

Menunjukkan partisi yang dibuat oleh `Unload` operasi.

Tipe: Panjang

Wajib: Tidak

UnloadWrittenBytes

Menunjukkan ukuran, dalam byte, ditulis oleh `Unload` operasi.

Tipe: Panjang

Wajib: Tidak

UnloadWrittenRows

Menunjukkan baris yang ditulis oleh `Unload` kueri.

Tipe: Panjang

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++

- AWS SDKuntuk Java V2
- AWS SDKuntuk Ruby V3

QuerySpatialCoverage
Layanan: Amazon Timestream Query

Memberikan wawasan tentang cakupan spasial kueri, termasuk tabel dengan pemangkasan spasial sub-optimal (maks). Informasi ini dapat membantu Anda mengidentifikasi area untuk perbaikan dalam strategi partisi Anda untuk meningkatkan pemangkasan spasial

Misalnya, Anda dapat melakukan hal berikut dengan `QuerySpatialCoverage` informasi:

- Tambahkan ukuran_name atau gunakan predikat kunci partisi () yang ditentukan pelanggan. CDPK
- Jika Anda sudah melakukan tindakan sebelumnya, hapus fungsi di sekitarnya atau klausa, seperti. `LIKE`

Daftar Isi

Max

Memberikan wawasan tentang cakupan spasial dari kueri yang dieksekusi dan tabel dengan pemangkasan spasial yang paling tidak efisien.

- `Value`— Rasio maksimum cakupan spasial.
- `TableArn`— Nama Sumber Daya Amazon (ARN) dari tabel dengan pemangkasan spasial yang kurang optimal.
- `PartitionKey`— Kunci partisi yang digunakan untuk partisi, yang dapat menjadi default `measure_name` atau file. CDPK

Tipe: Objek QuerySpatialCoverageMax

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

QuerySpatialCoverageMax
Layanan: Amazon Timestream Query

Memberikan wawasan ke dalam tabel dengan rentang spasial paling sub-optimal yang dipindai oleh kueri Anda.

Daftar Isi

PartitionKey

Kunci partisi yang digunakan untuk partisi, yang dapat menjadi default `measure_name` atau kunci partisi yang ditentukan pelanggan.

Tipe: Array string

Wajib: Tidak

TableArn

Nama Sumber Daya Amazon (ARN) dari tabel dengan pemangkasan spasial paling sub-optimal.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Tidak

Value

Rasio maksimum cakupan spasial.

Tipe: Ganda

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKuntuk Ruby V3

QueryStatus

Layanan: Amazon Timestream Query

Informasi tentang status kueri, termasuk kemajuan dan byte yang dipindai.

Daftar Isi

CumulativeBytesMetered

Jumlah data yang dipindai oleh kueri dalam byte yang akan dikenakan biaya. Ini adalah jumlah kumulatif dan mewakili jumlah total data yang akan dikenakan biaya sejak kueri dimulai. Biaya hanya diterapkan sekali dan diterapkan saat kueri selesai berjalan atau saat kueri dibatalkan.

Tipe: Panjang

Wajib: Tidak

CumulativeBytesScanned

Jumlah data yang dipindai oleh kueri dalam byte. Ini adalah jumlah kumulatif dan mewakili jumlah total byte yang dipindai sejak kueri dimulai.

Tipe: Panjang

Wajib: Tidak

ProgressPercentage

Kemajuan kueri, dinyatakan sebagai persentase.

Tipe: Ganda

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

QueryTemporalRange
Layanan: Amazon Timestream Query

Memberikan wawasan tentang rentang temporal kueri, termasuk tabel dengan rentang waktu (maks) terbesar.

Daftar Isi

Max

Merangkum properti berikut yang memberikan wawasan ke dalam tabel kinerja paling sub-optimal pada sumbu temporal:

- `Value`— Durasi maksimum dalam nanodetik antara awal dan akhir kueri.
- `TableArn`— Nama Sumber Daya Amazon (ARN) dari tabel yang ditanyakan dengan rentang waktu terbesar.

Tipe: Objek QueryTemporalRangeMax

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKuntuk Ruby V3

QueryTemporalRangeMax
Layanan: Amazon Timestream Query

Memberikan wawasan ke dalam tabel dengan pemangkasan temporal paling sub-optimal yang dipindai oleh kueri Anda.

Daftar Isi

TableArn

Amazon Resource Name (ARN) dari tabel yang ditanyakan dengan rentang waktu terbesar.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Tidak

Value

Durasi maksimum dalam nanodetik antara awal dan akhir kueri.

Tipe: Panjang

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

Row
Layanan: Amazon Timestream Query

Merupakan satu baris dalam hasil query.

Daftar Isi

Data

Daftar titik data dalam satu baris dari set hasil.

Tipe: Array objek Datum

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

S3Configuration
Layanan: Amazon Timestream Query

Detail tentang lokasi S3 untuk laporan kesalahan yang dihasilkan dari menjalankan kueri.

Daftar Isi

BucketName

Nama bucket S3 di mana laporan kesalahan akan dibuat.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 63.

Pola: `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Wajib: Ya

EncryptionOption

Opsi enkripsi saat istirahat untuk laporan kesalahan. Jika tidak ada opsi enkripsi yang ditentukan, Timestream akan memilih SSE _S3 sebagai default.

Tipe: String

Nilai yang Valid: `SSE_S3 | SSE_KMS`

Wajib: Tidak

ObjectKeyPrefix

Awalan untuk kunci laporan kesalahan. Timestream secara default menambahkan awalan berikut ke jalur laporan kesalahan.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 896.

Pola: `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKuntuk Ruby V3

S3ReportLocation
Layanan: Amazon Timestream Query

Lokasi laporan S3 untuk menjalankan kueri terjadwal.

Daftar Isi

BucketName

Nama bucket S3.

Tipe: String

Batasan Panjang: Panjang minimum 3. Panjang maksimum 63.

Pola: `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Wajib: Tidak

ObjectKey

Kunci S3.

Tipe: String

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

ScheduleConfiguration
Layanan: Amazon Timestream Query

Konfigurasi jadwal kueri.

Daftar Isi

ScheduleExpression

Ekspresi yang menunjukkan kapan harus memicu proses kueri terjadwal. Ini bisa berupa ekspresi cron atau ekspresi laju.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 256.

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

ScheduledQuery
Layanan: Amazon Timestream Query

Kueri Terjadwal

Daftar Isi

Arn

Nama Sumber Daya Amazon.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Ya

Name

Nama kueri terjadwal.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Wajib: Ya

State

Status kueri terjadwal.

Tipe: String

Nilai yang Valid: `ENABLED | DISABLED`

Wajib: Ya

CreationTime

Waktu pembuatan kueri terjadwal.

Tipe: Timestamp

Wajib: Tidak

ErrorReportConfiguration

Konfigurasi untuk pelaporan kesalahan kueri terjadwal.

Tipe: Objek ErrorReportConfiguration

Wajib: Tidak

LastRunStatus

Status proses kueri terjadwal terakhir.

Tipe: String

Nilai yang Valid: `AUTO_TRIGGER_SUCCESS | AUTO_TRIGGER_FAILURE | MANUAL_TRIGGER_SUCCESS | MANUAL_TRIGGER_FAILURE`

Wajib: Tidak

NextInvocationTime

Lain kali kueri terjadwal akan dijalankan.

Tipe: Timestamp

Wajib: Tidak

PreviousInvocationTime

Terakhir kali kueri terjadwal dijalankan.

Tipe: Timestamp

Wajib: Tidak

TargetDestination

Sumber data target di mana hasil kueri terjadwal akhir akan ditulis.

Tipe: Objek TargetDestination

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

ScheduledQueryDescription
Layanan: Amazon Timestream Query

Struktur yang menjelaskan kueri terjadwal.

Daftar Isi

Arn

Kueri terjadwalARN.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Ya

Name

Nama kueri terjadwal.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Wajib: Ya

NotificationConfiguration

Konfigurasi pemberitahuan.

Tipe: Objek NotificationConfiguration

Wajib: Ya

QueryString

Query yang akan dijalankan.

Tipe: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 262144.

Wajib: Ya

ScheduleConfiguration

Jadwalkan konfigurasi.

Tipe: Objek ScheduleConfiguration

Wajib: Ya

State

Status kueri yang dijadwalkan.

Tipe: String

Nilai yang Valid: `ENABLED | DISABLED`

Wajib: Ya

CreationTime

Waktu pembuatan kueri terjadwal.

Tipe: Timestamp

Wajib: Tidak

ErrorReportConfiguration

Konfigurasi pelaporan kesalahan untuk kueri terjadwal.

Tipe: Objek ErrorReportConfiguration

Wajib: Tidak

KmsKeyId

KMSKunci yang disediakan pelanggan digunakan untuk mengenkripsi sumber daya kueri terjadwal.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Tidak

LastRunSummary

Ringkasan runtime untuk menjalankan kueri terjadwal terakhir.

Tipe: Objek ScheduledQueryRunSummary

Wajib: Tidak

NextInvocationTime

Lain kali kueri terjadwal dijadwalkan untuk dijalankan.

Tipe: Timestamp

Wajib: Tidak

PreviousInvocationTime

Terakhir kali kueri dijalankan.

Tipe: Timestamp

Wajib: Tidak

RecentlyFailedRuns

Ringkasan runtime untuk lima kueri terjadwal terakhir yang gagal dijalankan.

Tipe: Array objek ScheduledQueryRunSummary

Wajib: Tidak

ScheduledQueryExecutionRoleArn

IAMperan yang digunakan Timestream untuk menjalankan kueri jadwal.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Tidak

TargetConfiguration

Konfigurasi toko target kueri terjadwal.

Tipe: Objek TargetConfiguration

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKuntuk Ruby V3

ScheduledQueryInsights
Layanan: Amazon Timestream Query

Merangkum pengaturan untuk mengaktifkan `QueryInsights` pada file. `ExecuteScheduledQueryRequest`

Daftar Isi

Mode

Menyediakan mode berikut untuk mengaktifkan`ScheduledQueryInsights`:

- `ENABLED_WITH_RATE_CONTROL`— `ScheduledQueryInsights` Memungkinkan kueri yang sedang diproses. Mode ini juga mencakup mekanisme kontrol laju, yang membatasi `QueryInsights` fitur hingga 1 kueri per detik (QPS).
- `DISABLED`— Menonaktifkan`ScheduledQueryInsights`.

Tipe: String

Nilai yang Valid: `ENABLED_WITH_RATE_CONTROL | DISABLED`

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKuntuk Ruby V3

ScheduledQueryInsightsResponse
Layanan: Amazon Timestream Query

Memberikan berbagai wawasan dan metrik yang terkait dengan `ExecuteScheduledQueryRequest` yang dieksekusi.

Daftar Isi

OutputBytes

Menunjukkan ukuran hasil kueri yang disetel dalam byte. Anda dapat menggunakan data ini untuk memvalidasi jika kumpulan hasil telah berubah sebagai bagian dari latihan penyetelan kueri.

Tipe: Panjang

Wajib: Tidak

OutputRows

Menunjukkan jumlah total baris yang dikembalikan sebagai bagian dari set hasil kueri. Anda dapat menggunakan data ini untuk memvalidasi jika jumlah baris dalam kumpulan hasil telah berubah sebagai bagian dari latihan penyetelan kueri.

Tipe: Panjang

Wajib: Tidak

QuerySpatialCoverage

Memberikan wawasan tentang cakupan spasial kueri, termasuk tabel dengan pemangkasan spasial sub-optimal (maks). Informasi ini dapat membantu Anda mengidentifikasi area untuk perbaikan dalam strategi partisi Anda untuk meningkatkan pemangkasan spasial.

Tipe: Objek QuerySpatialCoverage

Wajib: Tidak

QueryTableCount

Menunjukkan jumlah tabel dalam kueri.

Tipe: Panjang

Wajib: Tidak

QueryTemporalRange

Memberikan wawasan tentang rentang temporal kueri, termasuk tabel dengan rentang waktu (maks) terbesar. Berikut adalah beberapa opsi potensial untuk mengoptimalkan pemangkasan berbasis waktu:

- Tambahkan predikat waktu yang hilang.
- Hapus fungsi di sekitar predikat waktu.
- Tambahkan predikat waktu ke semua sub-kueri.

Tipe: Objek QueryTemporalRange

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

ScheduledQueryRunSummary
Layanan: Amazon Timestream Query

Jalankan ringkasan untuk kueri terjadwal

Daftar Isi

ErrorReportLocation

Lokasi S3 untuk laporan kesalahan.

Tipe: Objek ErrorReportLocation

Wajib: Tidak

ExecutionStats

Statistik runtime untuk jangka waktu yang dijadwalkan.

Tipe: Objek ExecutionStats

Wajib: Tidak

FailureReason

Pesan galat untuk kueri terjadwal jika terjadi kegagalan. Anda mungkin harus melihat laporan kesalahan untuk mendapatkan alasan kesalahan yang lebih rinci.

Tipe: String

Wajib: Tidak

InvocationTime

InvocationTime untuk lari ini. Ini adalah waktu di mana kueri dijadwalkan untuk dijalankan. Parameter `@scheduled_runtime` dapat digunakan dalam query untuk mendapatkan nilai.

Tipe: Timestamp

Wajib: Tidak

QueryInsightsResponse

Memberikan berbagai wawasan dan metrik yang terkait dengan ringkasan run dari kueri terjadwal.

Tipe: Objek ScheduledQueryInsightsResponse

Wajib: Tidak

RunStatus

Status menjalankan kueri terjadwal.

Tipe: String

Nilai yang Valid: `AUTO_TRIGGER_SUCCESS | AUTO_TRIGGER_FAILURE | MANUAL_TRIGGER_SUCCESS | MANUAL_TRIGGER_FAILURE`

Wajib: Tidak

TriggerTime

Waktu aktual ketika kueri dijalankan.

Tipe: Timestamp

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

SelectColumn
Layanan: Amazon Timestream Query

Rincian kolom yang dikembalikan oleh kueri.

Daftar Isi

Aliased

Benar, jika nama kolom alias oleh kueri. Salah sebaliknya.

Tipe: Boolean

Wajib: Tidak

DatabaseName

Database yang memiliki kolom ini.

Tipe: String

Wajib: Tidak

Name

Nama kolomnya.

Tipe: String

Wajib: Tidak

TableName

Tabel dalam database yang memiliki kolom ini.

Tipe: String

Wajib: Tidak

Type

Berisi tipe data kolom dalam kumpulan hasil kueri. Tipe data bisa skalar atau kompleks. Tipe data skalar yang didukung adalah bilangan bulat, Boolean, string, ganda, stempel waktu, tanggal, waktu, dan interval. Tipe data kompleks yang didukung adalah array, baris, dan timeseries.

Tipe: Objek Type

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

SnsConfiguration
Layanan: Amazon Timestream Query

Detail tentang SNS itu diperlukan untuk mengirim pemberitahuan.

Daftar Isi

TopicArn

SNStopik ARN yang pemberitahuan status kueri terjadwal akan dikirim ke.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

Tag

Layanan: Amazon Timestream Query

Tag adalah label yang Anda tetapkan ke and/or table. Each tag consists of a key and an optional value, both of which you define. Tags enable you to categorize databases and/or tabel database Timestream, misalnya, berdasarkan tujuan, pemilik, atau lingkungan.

Daftar Isi

Key

Kunci tag. Kunci tag peka huruf besar dan kecil.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Wajib: Ya

Value

Nilai tag. Nilai tag peka huruf besar/kecil dan bisa null.

Tipe: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 256.

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

TargetConfiguration
Layanan: Amazon Timestream Query

Konfigurasi yang digunakan untuk menulis output dari query.

Daftar Isi

TimestreamConfiguration

Konfigurasi yang diperlukan untuk menulis data ke dalam database Timestream dan tabel.

Tipe: Objek TimestreamConfiguration

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

TargetDestination
Layanan: Amazon Timestream Query

Detail tujuan untuk menulis data untuk sumber data target. Sumber data yang didukung saat ini adalah Timestream.

Daftar Isi

TimestreamDestination

Detail tujuan hasil kueri untuk sumber data Timestream.

Tipe: Objek TimestreamDestination

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

TimeSeriesDataPoint
Layanan: Amazon Timestream Query

Tipe data timeseries mewakili nilai ukuran dari waktu ke waktu. Deret waktu adalah array baris stempel waktu dan mengukur nilai, dengan baris diurutkan dalam urutan waktu menaik. A TimeSeriesDataPoint adalah titik data tunggal dalam deret waktu. Ini mewakili tupel (waktu, nilai ukuran) dalam deret waktu.

Daftar Isi

Time

Stempel waktu saat nilai ukuran dikumpulkan.

Tipe: String

Diperlukan: Ya

Value

Nilai ukuran untuk titik data.

Tipe: Objek Datum

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

TimestreamConfiguration
Layanan: Amazon Timestream Query

Konfigurasi untuk menulis data ke database Timestream dan tabel. Konfigurasi ini memungkinkan pengguna untuk memetakan kolom pilih hasil kueri ke kolom tabel tujuan.

Daftar Isi

DatabaseName

Nama database Timestream dimana hasil query akan ditulis.

Tipe: String

Diperlukan: Ya

DimensionMappings

Hal ini untuk memungkinkan pemetaan kolom (s) dari hasil query ke dimensi dalam tabel tujuan.

Tipe: Array objek DimensionMapping

Wajib: Ya

TableName

Nama tabel Timestream tempat hasil kueri akan ditulis. Tabel harus berada dalam database yang sama yang disediakan dalam konfigurasi Timestream.

Tipe: String

Diperlukan: Ya

TimeColumn

Kolom dari hasil query yang harus digunakan sebagai kolom waktu dalam tabel tujuan. Jenis kolom untuk ini seharusnyaTIMESTAMP.

Tipe: String

Diperlukan: Ya

MeasureNameColumn

Nama kolom ukuran.

Tipe: String

Wajib: Tidak

MixedMeasureMappings

Menentukan cara memetakan langkah-langkah untuk catatan multi-ukuran.

Tipe: Array objek MixedMeasureMapping

Anggota Array: Jumlah minimum 1 item.

Wajib: Tidak

MultiMeasureMappings

Pemetaan multi-ukuran.

Tipe: Objek MultiMeasureMappings

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

TimestreamDestination
Layanan: Amazon Timestream Query

Tujuan untuk kueri terjadwal.

Daftar Isi

DatabaseName

Nama database Timestream.

Tipe: String

Wajib: Tidak

TableName

Nama tabel Timestream.

Tipe: String

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

Type
Layanan: Amazon Timestream Query

Berisi tipe data kolom dalam kumpulan hasil kueri. Tipe data bisa skalar atau kompleks. Tipe data skalar yang didukung adalah bilangan bulat, Boolean, string, ganda, stempel waktu, tanggal, waktu, dan interval. Tipe data kompleks yang didukung adalah array, baris, dan timeseries.

Daftar Isi

ArrayColumnInfo

Menunjukkan jika kolom adalah array.

Tipe: Objek ColumnInfo

Wajib: Tidak

RowColumnInfo

Menunjukkan jika kolom adalah baris.

Tipe: Array objek ColumnInfo

Wajib: Tidak

ScalarType

Menunjukkan jika kolom adalah tipe string, integer, Boolean, ganda, stempel waktu, tanggal, waktu. Untuk informasi selengkapnya, lihat Tipe data yang didukung.

Tipe: String

Nilai yang Valid: `VARCHAR | BOOLEAN | BIGINT | DOUBLE | TIMESTAMP | DATE | TIME | INTERVAL_DAY_TO_SECOND | INTERVAL_YEAR_TO_MONTH | UNKNOWN | INTEGER`

Wajib: Tidak

TimeSeriesMeasureValueColumnInfo

Menunjukkan jika kolom adalah tipe data timeseries.

Tipe: Objek ColumnInfo

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan ini API di salah satu bahasa khusus AWS SDKs, lihat berikut ini:

- AWS SDKuntuk C ++
- AWS SDKuntuk Java V2
- AWS SDKpara Ruby V3

Kesalahan Umum

Bagian ini mencantumkan kesalahan yang umum terjadi pada API tindakan semua AWS layanan. Untuk kesalahan khusus untuk API tindakan untuk layanan ini, lihat topik untuk API tindakan tersebut.

AccessDeniedException

Anda tidak memiliki akses yang memadai untuk melakukan tindakan ini.

HTTPKode Status: 400

IncompleteSignature

Tanda tangan permintaan tidak sesuai dengan AWS standar.

HTTPKode Status: 400

InternalFailure

Pemrosesan permintaan telah gagal karena kesalahan yang tidak diketahui, pengecualian atau kegagalan.

HTTPKode Status: 500

InvalidAction

Tindakan atau operasi yang diminta tidak valid. Verifikasi bahwa tindakan diketik dengan benar.

HTTPKode Status: 400

InvalidClientTokenId

Sertifikat X.509 atau ID kunci AWS akses yang disediakan tidak ada dalam catatan kami.

HTTPKode Status: 403

NotAuthorized

Anda tidak memiliki izin untuk melakukan tindakan ini.

HTTPKode Status: 400

OptInRequired

ID kunci AWS akses memerlukan langganan untuk layanan ini.

HTTPKode Status: 403

RequestExpired

Permintaan mencapai layanan lebih dari 15 menit setelah cap tanggal pada permintaan atau lebih dari 15 menit setelah tanggal kedaluwarsa permintaan (seperti untuk pra-ditandatanganiURLs), atau cap tanggal pada permintaan lebih dari 15 menit di masa depan.

HTTPKode Status: 400

ServiceUnavailable

Permintaan telah gagal karena kegagalan sementara server.

HTTPKode Status: 503

ThrottlingException

Permintaan ditolak karena throttling permintaan.

HTTPKode Status: 400

ValidationError

Input gagal memenuhi kendala yang ditentukan oleh layanan. AWS

HTTPKode Status: 400

Parameter Umum

Daftar berikut berisi parameter yang digunakan semua tindakan untuk menandatangani permintaan Tanda Tangan Versi 4 dengan string kueri. Setiap parameter khusus tindakan tercantum dalam

topik untuk tindakan tersebut. Untuk informasi selengkapnya tentang Tanda Tangan Versi 4, lihat Menandatangani AWS API permintaan di Panduan IAM Pengguna.

Action

Tindakan yang harus dilakukan.

Tipe: string

Diperlukan: Ya

Version

APIVersi yang permintaan ditulis untuk, dinyatakan dalam format YYYY-MM-DD.

Jenis: string

Diperlukan: Ya

X-Amz-Algorithm

Algoritme hash yang Anda gunakan untuk membuat tanda tangan permintaan.

Kondisi: Tentukan parameter ini saat Anda menyertakan informasi otentikasi dalam string kueri, bukan di header HTTP otorisasi.

Jenis: string

Nilai yang Valid: `AWS4-HMAC-SHA256`

Diperlukan: Kondisional

X-Amz-Credential

Nilai lingkup kredensial, yang merupakan string yang menyertakan access key Anda, tanggal, wilayah yang Anda targetkan, layanan yang Anda minta, dan string penghentian ("aws4_request"). Nilai dinyatakan dalam format berikut: access_key//region YYYYMMDD/service /aws4_request.

Untuk informasi selengkapnya, lihat Membuat AWS API permintaan yang ditandatangani di Panduan IAM Pengguna.

Kondisi: Tentukan parameter ini saat Anda menyertakan informasi otentikasi dalam string kueri, bukan di header HTTP otorisasi.

Jenis: string

Wajib: Bersyarat

X-Amz-Date

Tanggal yang digunakan untuk membuat tanda tangan. Formatnya harus ISO 8601 format dasar (YYYYMMDD'T' 'ZHHMMSS'). Misalnya, waktu tanggal berikut adalah X-Amz-Date nilai yang valid:`20120325T120000Z`.

Kondisi: X-Amz-Date bersifat opsional untuk semua permintaan; dapat digunakan untuk mengganti tanggal yang digunakan untuk menandatangani permintaan. Jika header Tanggal ditentukan dalam format dasar ISO 8601, tidak X-Amz-Date diperlukan. Ketika X-Amz-Date digunakan, itu selalu mengesampingkan nilai header Tanggal. Untuk informasi selengkapnya, lihat Elemen tanda tangan AWS API permintaan di Panduan IAM Pengguna.

Jenis: string

Wajib: Bersyarat

X-Amz-Security-Token

Token keamanan sementara yang diperoleh melalui panggilan ke AWS Security Token Service (AWS STS). Untuk daftar layanan yang mendukung kredensyal keamanan sementara AWS STS, lihat layanan Layanan AWS yang berfungsi IAM di IAMPanduan Pengguna.

Kondisi: Jika Anda menggunakan kredensyal keamanan sementara dari AWS STS, Anda harus menyertakan token keamanan.

Jenis: string

Wajib: Bersyarat

X-Amz-Signature

Menentukan tanda tangan yang dikodekan oleh hex yang dihitung dari string to sign dan kunci penandatanganan turunan.

Kondisi: Tentukan parameter ini saat Anda menyertakan informasi otentikasi dalam string kueri, bukan di header HTTP otorisasi.

Jenis: string

Wajib: Bersyarat

X-Amz-SignedHeaders

Menentukan semua HTTP header yang disertakan sebagai bagian dari permintaan kanonik. Untuk informasi selengkapnya tentang menentukan header yang ditandatangani, lihat Membuat AWS API permintaan yang ditandatangani di IAMPanduan Pengguna.

Kondisi: Tentukan parameter ini saat Anda menyertakan informasi otentikasi dalam string kueri, bukan di header HTTP otorisasi.

Jenis: string

Diperlukan: Kondisional

Riwayat dokumen

Perubahan	Deskripsi	Tanggal
Pembaruan khusus dokumentasi	Memperbarui topik Kuota untuk memisahkan kuota default dan batas sistem.	Oktober 22, 2024
Amazon Timestream sekarang mendukung wawasan kueri	Timestream sekarang menyertakan dukungan untuk fitur wawasan kueri yang membantu Anda mengoptim alkan kueri, meningkatkan kinerjanya, dan mengurangi biaya.	Oktober 22, 2024
Amazon Timestream untuk InfluxDB memperbarui ke kebijakan yang ada.	Amazon Timestream untuk InfluxDB telah menambahk an `ec2:DescribeRouteT ables` tindakan ke kebijakan `AmazonTimestreamIn fluxDBFullAccess` terkelola yang ada untuk menjelaskan tabel rute Anda	Oktober 8, 2024

AmazonTimestreamReadOnlyAccess — Perbarui ke kebijakan yang ada	Timestream for LiveAnalytics telah menambahkan DescribeAccountSettings izin ke kebijakan AmazonTimestreamReadOnlyAccess terkelola untuk menjelaskan Akun AWS setelan.	Juni 3, 2024
Amazon Timestream untuk LiveAnalytics saat ini mendukung Unit Komputasi Timestream () TCUs	Amazon Timestream untuk LiveAnalytics saat ini menyertakan dukungan untuk Timestream Compute Units (TCUs) untuk mengukur kapasitas komputasi yang dialokasikan untuk kebutuhan kueri Anda.	April 29, 2024

Kebijakan baru ditambahkan	Amazon Timestream untuk InfluxDB menambahkan dua kebijakan baru: Satu yang memungkinkan layanan untuk mengelola antarmuka jaringan dan grup keamanan di akun Anda. Untuk informasi lebih lanjut, lihat AmazonTimestreamInfluxDBServiceRolePolicy. Lain yang menyediakan akses administratif penuh untuk membuat, memperbarui, menghapus, dan mencantumkan instans Amazon TimeStream InfluxDB serta membuat dan membuat daftar grup parameter. Untuk informasi lebih lanjut, lihat AmazonTimestreamInfluxDBFullAccess.	Maret 14, 2024
Amazon Timestream untuk InfluxDB sekarang tersedia secara umum.	Dokumentasi ini mencakup rilis awal Amazon Timestream untuk InfluxDB.	Maret 14, 2024

Amazon Timestream untuk acara LiveAnalytics Kueri tersedia di AWS CloudTrail	Amazon Timestream untuk LiveAnalytics saat ini menerbitkan peristiwa API data Kueri ke. AWS CloudTrai l Pelanggan dapat mengaudit semua API permintaan Kueri yang dibuat di AWS akun mereka, dan melihat informasi seperti IAM Pengguna/ Peran mana yang membuat permintaan, kapan permintaa n dibuat, database dan tabel mana yang ditanyakan, dan ID Kueri permintaan.	12 September 2023
Amazon Timestream untuk LiveAnalytics UNLOAD	Amazon Timestream untuk LiveAnalytics saat ini mendukung UNLOAD untuk mengekspor hasil kueri ke S3.	12 Mei 2023
Amazon Timestream untuk LiveAnalytics pembaruan ke kebijakan yang ada.	Izin pemuatan batch ditambahkan ke kebijakan terkelola.	Februari 24, 2023
Amazon Timestream untuk pemuatan LiveAnalytics batch.	Amazon Timestream untuk LiveAnalytics saat ini mendukung fungsionalitas pemuatan batch.	Februari 24, 2023
Amazon Timestream untuk LiveAnalytics saat ini mendukung. AWS Backup	Amazon Timestream untuk LiveAnalytics saat ini mendukung. AWS Backup	14 Desember 2022

Amazon Timestream untuk LiveAnalytics pembaruan kebijakan terkelola AWS	Informasi baru tentang kebijakan AWS terkelola dan Amazon Timestream untuk LiveAnalytics, termasuk pembaruan kebijakan terkelola yang ada.	29 November 2021
Amazon Timestream untuk LiveAnalytics mendukung kueri terjadwal	Amazon Timestream untuk LiveAnalytics saat ini mendukung menjalankan kueri atas nama Anda, berdasarkan jadwal.	29 November 2021
Amazon Timestream untuk LiveAnalytics mendukung penyimpanan magnetik.	Amazon Timestream untuk LiveAnalytics saat ini mendukung penggunaan penyimpanan magnetik untuk penulisan tabel Anda.	29 November 2021
Amazon Timestream untuk catatan LiveAnalytics multi-ukuran.	Amazon Timestream untuk LiveAnalytics saat ini mendukung format yang lebih ringkas untuk menyimpan data deret waktu Anda.	29 November 2021
Amazon Timestream untuk LiveAnalytics pembaruan kebijakan terkelola AWS	Informasi baru tentang kebijakan AWS terkelola dan Amazon Timestream untuk LiveAnalytics, termasuk pembaruan kebijakan terkelola yang ada.	24 Mei 2021
Amazon Timestream untuk sekarang LiveAnalytics tersedia di wilayah eropa (frankfurt).	Amazon Timestream untuk sekarang LiveAnalytics umumnya tersedia di wilayah Eropa (Frankfurt) (). `eu-central-1`	23 April 2021

Amazon Timestream for LiveAnalytics is sekarang mendukung VPC endpoints ().AWS PrivateLink	Amazon Timestream untuk LiveAnalytics saat ini mendukung penggunaan VPC endpoints ().AWS PrivateLink	23 Maret 2021
Amazon Timestream sekarang mendukung kueri lintas tabel.	Anda dapat menggunak an Amazon Timestream LiveAnalytics untuk menjalank an kueri lintas tabel.	10 Februari 2021
Amazon Timestream untuk LiveAnalytics saat ini mendukung statistik eksekusi kueri yang disempurnakan.	Amazon Timestream untuk LiveAnalytics saat ini mendukung statistik eksekusi kueri yang disempurnakan, seperti jumlah data yang dipindai.	10 Februari 2021
Amazon Timestream untuk LiveAnalytics saat ini mendukung fungsi deret waktu lanjutan.	Anda dapat menggunak an Amazon Timestream LiveAnalytics untuk menjalank an SQL kueri dengan fungsi deret waktu lanjutan, seperti turunan, integral, dan korelasi.	10 Februari 2021
Amazon Timestream untuk saat LiveAnalytics iniHIPAA, ISO, dan PCI sesuai.	Sekarang Anda dapat menggunakan Amazon Timestream LiveAnalytics untuk beban kerja yang memerlukan HIPAAISO, dan PCI infrastruktur yang sesuai.	27 Januari 2021

Amazon Timestream untuk LiveAnalytics saat ini mendukung telegraf dan grafana sumber terbuka.	Anda sekarang dapat menggunakan Telegraf, agen server open-source, berbasis plugin untuk mengumpulkan dan melaporkan metrik, dan Grafana, platform analitik dan pemantauan sumber terbuka untuk database, dengan Amazon Timestream for. LiveAnalytics	25 November 2020
Amazon Timestream untuk sekarang LiveAnalytics tersedia secara umum.	Dokumentasi ini mencakup rilis awal Amazon Timestream untuk. LiveAnalytics	30 September 2020

Apa itu Timestream untuk InfluxDB?

Amazon TimeStream for InfluxDB adalah mesin database seri waktu terkelola yang memudahkan pengembang dan DevOps tim aplikasi untuk menjalankan database InfluxDB untuk aplikasi deret waktu nyata menggunakan sumber terbuka. AWS APIs Dengan Amazon TimeStream untuk InfluxDB, mudah untuk mengatur, mengoperasikan, dan menskalakan beban kerja deret waktu yang dapat menjawab kueri dengan waktu respons kueri milidetik satu digit.

Amazon TimeStream untuk InfluxDB memberi Anda akses ke kemampuan InfluxDB versi open source yang sudah dikenal di cabang 2.x-nya. Ini berarti bahwa kode, aplikasi, dan alat yang sudah Anda gunakan saat ini dengan database sumber terbuka InfluxDB yang ada harus bekerja dengan mulus dengan Amazon Timestream untuk InfluxDB. Amazon TimeStream untuk InfluxDB dapat secara otomatis mencadangkan database Anda dan menjaga perangkat lunak database Anda tetap up to date dengan versi terbaru. Selain itu, Amazon TimeStream untuk InfluxDB memudahkan penggunaan replikasi untuk meningkatkan ketersediaan database, dan meningkatkan daya tahan data. Seperti semua AWS layanan, tidak ada investasi di muka yang diperlukan, dan Anda hanya membayar untuk sumber daya yang Anda gunakan.

Instans DB

Instans DB adalah lingkungan basis data terisolasi yang berjalan di cloud. Ini adalah blok bangunan dasar Amazon Timestream untuk InfluxDB. Instans DB dapat berisi beberapa database yang dibuat pengguna (atau organisasi dan bucket untuk kasus database InfluxDb 2.x), dan dapat diakses menggunakan alat dan aplikasi klien yang sama yang mungkin Anda gunakan untuk mengakses instans InfluxDB mandiri yang dikelola sendiri. Instans DB mudah dibuat dan dimodifikasi dengan alat baris AWS perintah, operasi Amazon TimeStream API InfluxDB, atau. AWS Management Console

> **Note**
>
> Amazon Timestream untuk InfluxDB mendukung akses ke database menggunakan operasi Influx dan API Influx UI. Amazon Timestream untuk InfluxDB tidak mengizinkan akses host langsung.

Anda dapat memiliki hingga 40 Amazon Timestream untuk instans InfluxDB.

Setiap instans DB memiliki nama instans DB. Nama yang disediakan pelanggan ini secara unik mengidentifikasi instans DB saat berinteraksi dengan Amazon Timestream untuk InfluxDB dan perintah. API AWS CLI Nama instans DB harus unik untuk pelanggan di suatu AWS Wilayah.

Nama instans DB merupakan bagian dari DNS nama host yang dialokasikan ke instans Anda oleh Timestream untuk InfluxDB. Misalnya, jika Anda menentukan influxdb1 sebagai nama instans DB, Timestream akan secara otomatis mengalokasikan titik akhir untuk instance Anda. DNS Contoh titik akhir adalah`influxdb1-3ksj4dla5nfjhi.us-east-1.timestream-influxdb.amazonaws.com`, di mana `influxdb1` nama instance Anda.

Dalam contoh titik akhir`influxdb1-3ksj4dla5nfjhi.us-east-1.timestream-influxdb.amazonaws.com`, string `3ksj4dla5nfjhi` adalah pengidentifikasi akun unik yang dihasilkan oleh. AWS Pengidentifikasi `3ksj4dla5nfjhi` dalam contoh tidak berubah untuk akun yang ditentukan di wilayah tertentu. Oleh karena itu, semua instans DB Anda yang dibuat oleh akun ini berbagi pengenal tetap yang sama. Pertimbangkan fitur pengenal tetap berikut:

- Saat ini Timestream untuk InfluxDB tidak mendukung penggantian nama instans DB.
- Jika Anda menghapus dan membuat ulang instans DB dengan pengidentifikasi instans DB yang sama, titik akhirnya akan sama.
- Jika Anda menggunakan akun yang sama untuk membuat instans DB di Wilayah yang berbeda, pengidentifikasi yang dibuat secara internal akan berbeda karena Wilayahnya berbeda, seperti dalam `influxdb2.4a3j5du5ks7md2.us-west-1.timestream-influxdb.amazonaws.com`.

Setiap instans DB hanya mendukung satu Timestream untuk mesin database InfluxDB.

Saat membuat instance DB, InfluxDB mengharuskan nama organisasi ditentukan. Instans DB dapat menampung beberapa organisasi dan beberapa bucket yang terkait dengan setiap organisasi.

Amazon Timestream untuk InfluxDB memungkinkan Anda membuat akun pengguna master dan kata sandi untuk instans DB Anda sebagai bagian dari proses pembuatan. Pengguna master ini memiliki izin untuk membuat organisasi, bucket, dan melakukan operasi baca, tulis, hapus, dan upsert pada data Anda. Anda juga akan dapat mengakses InfluxUI dan mengambil token operator Anda. login pertama Anda. Dari sana Anda akan dapat mengelola semua token akses Anda juga. Anda harus menyetel kata sandi pengguna utama saat membuat instans DB, tetapi Anda dapat mengubahnya kapan saja menggunakan Influx, Influx APICLI, atau InfluxUI.

Kelas instans DB

Kelas instans DB menentukan komputasi dan kapasitas memori instans Amazon `fi UbfkyxDB` Timestream DB. Kelas instans DB yang Anda butuhkan tergantung pada kebutuhan daya dan memori pemrosesan Anda.

Sebuah kelas instans DB terdiri dari jenis dan ukuran kelas instans DB. Misalnya, `db.influx` adalah jenis kelas instans DB yang dioptimalkan untuk memori yang cocok untuk persyaratan memori kinerja tinggi yang terkait dengan menjalankan InfluxDb beban kerja. Dalam tipe kelas `db.influx` instance, `db.influx.2xlarge` adalah kelas instance DB. Ukuran kelas ini adalah 2xlarge.

Untuk informasi selengkapnya tentang harga kelas instans, lihat Amazon TimeStream untuk harga InfluxDB.

Jenis kelas instans DB

Amazon Timestream untuk InfluxDB mendukung kelas instans DB untuk kasus penggunaan berikut yang dioptimalkan untuk kasus penggunaan InfluxDB.

- **`db.influx`**—Kelas instance ini ideal untuk menjalankan beban kerja intensif memori dalam database InfluxDB sumber terbuka

Spesifikasi perangkat keras untuk kelas instans DB

Terminologi berikut menjelaskan spesifikasi perangkat keras untuk kelas instans DB:

- v CPU

  Jumlah unit pemrosesan pusat virtual (CPUs). Virtual CPU adalah unit kapasitas yang dapat Anda gunakan untuk membandingkan kelas instans DB.

- Memori (GiB)

  ItuRAM, dalam gibibytes, dialokasikan ke instance DB. Seringkali ada rasio yang konsisten antara memori dan vCPU. Sebagai contoh, ambil kelas instance db.influx, yang memiliki CPU rasio memori ke v yang mirip dengan kelas instance EC2 r7g.

- Influx-Dioptimalkan

Instans DB menggunakan tumpukan konfigurasi yang dioptimalkan dan menyediakan kapasitas khusus tambahan untuk I/O. Pengoptimalan ini memberikan performa terbaik dengan meminimalkan konflik antara I/O dan lalu lintas lain dari instans Anda.

- Bandwidth jaringan

  Kecepatan jaringan relatif terhadap kelas instans DB lainnya. Dalam tabel berikut, Anda dapat menemukan detail perangkat keras tentang Amazon Timestream untuk kelas instans InfluxDB.

Kelas Instance	v CPU	Memori (GiB)	Jenis Penyimpanan	Bandwidth jaringan (Gbps)
db.influx.medium	1	8	Influx Termasuk IOPS	10
db.influx.large	2	16	Influx Termasuk IOPS	10
db.influx.xlarge	4	32	Influx Termasuk IOPS	10
db.influx.2xlarge	8	64	Influx Termasuk IOPS	10
db.influx.4xlarge	16	128	Influx Termasuk IOPS	10
db.influx.8xlarge	32	256	Influx Termasuk IOPS	12
db.influx.12xlarge	48	384	Influx Termasuk IOPS	20
db.influx.16xlarge	64	512	Influx Termasuk IOPS	25

Penyimpanan instans InfluxDB

Instans DB untuk Amazon Timestream untuk InfluxDB menggunakan volume IOPS Influx Included untuk database dan penyimpanan log.

Dalam beberapa kasus, beban kerja database Anda mungkin tidak dapat mencapai 100 persen dari IOPS yang telah Anda sediakan. Untuk informasi selengkapnya, lihat Faktor yang memengaruhi performa penyimpanan. Untuk informasi selengkapnya tentang Timestream untuk harga penyimpanan InfluxDB, lihat harga Amazon Timestream.

Amazon Timestream untuk jenis penyimpanan InfluxDB

Amazon Timestream untuk InfluxDB menyediakan dukungan untuk satu jenis penyimpanan, Influx Included. IOPS Anda dapat membuat Timestream untuk instans InfluxDB dengan penyimpanan hingga 16 tebibytes (TiB).

Berikut adalah deskripsi singkat dari jenis penyimpanan yang tersedia:

- Influx IO Termasuk penyimpanan: Kinerja penyimpanan adalah kombinasi dari operasi I/O per detik (IOPS) dan seberapa cepat volume penyimpanan dapat melakukan pembacaan dan penulisan (throughput penyimpanan). Pada volume penyimpanan Influx IOPS Included, Amazon Timestream untuk InfluxDB menyediakan 3 tingkatan penyimpanan yang telah dikonfigurasi sebelumnya dengan IOPS optimal dan throughput yang diperlukan untuk berbagai jenis beban kerja.

Ukuran instans InfluxDB

Konfigurasi optimal Timestream untuk instans InfluxDB bergantung pada banyak faktor yang mencakup tingkat konsumsi, ukuran batch, kardinalitas deret waktu, kueri bersamaan, dan jenis kueri. Dalam upaya memberikan rekomendasi ukuran, kami berfokus pada beban kerja yang patut dicontoh dengan karakteristik sebagai berikut:

- Data dikumpulkan dan ditulis oleh armada agen Telegraf yang mengumpulkan Sistem, MemoriCPU, Disk, IO, dan lain-lain dari pusat data.

  Setiap permintaan tulis berisi 5000 baris.
- Jenis kueri yang dijalankan pada sistem dikategorikan sebagai kueri “kompleksitas sedang”. Kategori kueri ini menyajikan karakteristik berikut:
  - Memiliki beberapa fungsi dan satu atau dua ekspresi reguler

- Mungkin juga memiliki kelompok demi klausa atau sampel rentang waktu beberapa minggu.
- Biasanya membutuhkan beberapa ratus milidetik hingga beberapa ribu milidetik untuk dieksekusi.
- CPUmendukung kinerja kueri terutama.

Kelas instans	Jenis Penyimpanan	Menulis (baris per detik)	Membaca (Kueri per detik)
db.influx.large	Influx IO Termasuk 3K	~ 50.000	<10
db.influx.2xlarge	Influx IO Termasuk 3K	~ 150.000	<25
db.influx.4xlarge	Influx IO Termasuk 3K	~ 200.000	~ 25
db.influx.4xlarge	Influx IO Termasuk 12K	~ 250.000	~35
db.influx.8xlarge	Influx IO Termasuk 12K	~ 500.000	~50
db.influx.12xlarge	Influx IO Termasuk 12K	<750.000	<100

AWS Wilayah dan Zona Ketersediaan

Sumber daya komputasi cloud Amazon di-hosting di beberapa lokasi di seluruh dunia. Lokasi ini terdiri dari AWS Wilayah dan zona Ketersediaan. Setiap AWS wilayah adalah wilayah geografis yang terpisah. Setiap AWS Wilayah memiliki beberapa lokasi terisolasi yang dikenal sebagai zona ketersediaan.

> **Note**
>
> Untuk informasi tentang menemukan zona ketersediaan untuk suatu AWS Wilayah, lihat Wilayah dan Zona di Panduan EC2 Pengguna Amazon.

Amazon TimeStream untuk InfluxDB memungkinkan Anda menempatkan sumber daya, seperti instans DB, dan data di beberapa lokasi.

Amazon beroperasi state-of-the-art, pusat data yang sangat tersedia. Meskipun jarang, kegagalan yang memengaruhi ketersediaan instans DB yang berada di lokasi yang sama dapat terjadi. Jika Anda meng-hosting semua instans DB di satu lokasi yang terpengaruh oleh kegagalan tersebut, tidak satu pun instans DB Anda akan tersedia.

Penting untuk diingat bahwa setiap AWS Wilayah sepenuhnya independen. Setiap Amazon TimeStream untuk aktivitas InfluxDB yang Anda lakukan (misalnya, membuat instance database atau mencantumkan instance database yang tersedia) hanya berjalan di Wilayah default Anda saat ini. AWS Wilayah AWS default dapat diubah di konsol atau dengan mengatur variabel lingkungan `AWS_DEFAULT_REGION`. Atau dapat diganti dengan menggunakan `--region` parameter dengan (). AWS Command Line Interface AWS CLI Untuk informasi selengkapnya, lihat Mengkonfigurasi AWS Command Line Interface, khususnya bagian tentang variabel lingkungan dan opsi baris perintah.

Untuk membuat atau bekerja dengan Amazon Timestream untuk instans DB InfluxDB di AWS Wilayah tertentu, gunakan titik akhir layanan regional yang sesuai.

AWS Ketersediaan wilayah

Untuk informasi selengkapnya tentang AWS Wilayah di mana Amazon TimeStream untuk InfluxDB saat ini tersedia dan titik akhir untuk setiap Wilayah, lihat titik akhir dan kuota Amazon Timestream.

AWS Desain daerah

Setiap AWS Wilayah dirancang untuk diisolasi dari AWS Wilayah lain. Rancangan ini mencapai toleransi kesalahan dan stabilitas sebesar mungkin.

Saat Anda melihat sumber daya, Anda hanya melihat sumber daya yang terkait dengan AWS Wilayah yang Anda tentukan. Ini karena AWS Wilayah terisolasi satu sama lain, dan kami tidak secara otomatis mereplikasi sumber daya di seluruh AWS Wilayah.

AWS Zona Ketersediaan

Saat Anda membuat instans DB, Amazon Timestream untuk InfluxDB memilih satu untuk Anda secara acak berdasarkan konfigurasi subnet Anda. Availability Zone diwakili oleh kode AWS Region diikuti oleh pengidentifikasi huruf (misalnya,`us-east-1a`).

Gunakan EC2 perintah `describe-availability` -zones Amazon sebagai berikut untuk menjelaskan Availability Zone dalam Wilayah tertentu yang diaktifkan untuk akun Anda.

```
aws ec2 describe-availability-zones --region region-name
```

Misalnya, untuk mendeskripsikan Availability Zone di Wilayah AS Timur (Virginia N.) (us-east-1) yang diaktifkan untuk akun Anda, jalankan perintah berikut:

```
aws ec2 describe-availability-zones --region us-east-1
```

Anda tidak dapat memilih Availability Zones untuk instans DB primer dan sekunder dalam penerapan DB multi-AZ. Amazon Timesteram untuk InfluxDB memilihnya untuk Anda secara acak. Untuk informasi selengkapnya tentang penerapan Multi-AZ, lihat.. Mengkonfigurasi dan mengelola penyebaran Multi-AZ

Penagihan Instans DB untuk Amazon Timestream untuk InfluxDB

Amazon Timestream untuk instans InfluxDB ditagih berdasarkan komponen berikut:

- Jam instans DB (per jam) - Berdasarkan kelas instans DB dari instance DB, misalnya, db.influx.large. Harga dicantumkan per jam, tetapi tagihan dihitung turun menjadi detik dan menunjukkan waktu dalam bentuk desimal. Amazon Timestream untuk penggunaan InfluxDB ditagih dalam kenaikan 1 detik, dengan minimal 10 menit. Untuk informasi selengkapnya, lihat kelas instans Kelas instans DB DB.
- Penyimpanan (per GiB per bulan) - Kapasitas penyimpanan yang telah Anda berikan ke instans DB Anda. Untuk informasi selengkapnya, lihat Penyimpanan instans InfluxDB.
- Transfer data (per GB) — Transfer data masuk dan keluar dari instans DB Anda dari atau ke internet dan AWS Wilayah lain.

Untuk informasi harga Amazon Timestream untuk InfluxDB, lihat halaman harga Amazon Timestream untuk InfluxDB.

Menyiapkan Amazon Timestream untuk InfluxDB

Sebelum Anda menggunakan Amazon Timestream untuk InfluxDB untuk pertama kalinya, selesaikan tugas-tugas berikut:

Jika Anda sudah memiliki AWS akun, ketahui Amazon Timestream untuk persyaratan InfluxDB, dan pilih untuk menggunakan default untuk dan IAM Memulai VPC Memulai Timestream untuk InfluxDB Amazon Timestream untuk InfluxDB.

Mendaftar untuk AWS akun

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

Untuk mendaftar AWS akun

- Buka halaman AWS Masuk.
- Pilih Buat akun baru dan ikuti petunjuknya.

> **Note**
>
> Bagian dari prosedur pendaftaran melibatkan tindakan menerima panggilan telepon dan memasukkan kode verifikasi di keypad telepon.

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

AWS mengirimi Anda email konfirmasi setelah proses pendaftaran selesai. Kapan saja, Anda dapat melihat aktivitas akun Anda saat ini dan mengelola akun Anda dengan masuk https://aws.amazon.comke/ dan memilih Akun Saya.

Pengelolaan Pengguna

Buat pengguna administratif

Membuat pengguna administratif

Setelah Anda mendaftar untuk AWS akun, buat pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Amankan pengguna root AWS akun Anda

Masuk ke Konsol AWS Manajemen sebagai pemilik akun dengan memilih pengguna Root dan memasukkan alamat email AWS akun Anda. Di laman berikutnya, masukkan kata sandi. Untuk bantuan saat masuk menggunakan pengguna root, lihat Masuk sebagai pengguna root di Panduan Pengguna AWS Masuk

Aktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda. Untuk petunjuk, lihat Mengaktifkan MFA perangkat virtual untuk pengguna root AWS akun Anda (konsol) di Panduan IAM Pengguna.

Berikan akses terprogram

Pengguna membutuhkan akses terprogram jika mereka ingin berinteraksi dengan AWS luar. AWS Management Console Cara memberikan akses programatis bergantung pada jenis pengguna yang mengakses AWS.

Untuk memberikan akses terprogram kepada pengguna, pilih salah satu opsi berikut:

Pengguna mana yang membutuhkan akses programatis?	Untuk	Oleh
Identitas tenaga kerja (Pengguna dikelola di Pusat IAM Identitas)	Gunakan kredensi sementara untuk menandatangani permintaan terprogram ke AWS CLI,, AWS SDKs atau. AWS APIs	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan.* Untuk, lihat AWS CLI Mengkonfigurasi AWS CLI untuk menggunakan AWS IAM Identity Center di Panduan Pengguna Antarmuka Baris Perintah AWS .* Untuk AWS SDKs, alat, dan AWS APIs, lihat IAMOtentikasi Pusat Identitas di AWSSDKsdan Panduan Referensi Alat.
IAM	Gunakan kredensi sementara untuk menandatangani permintaan terprogram ke AWS CLI,, SDKs dan. APIs	Mengikuti petunjuk dalam Menggunakan kredensil sementara dengan AWS sumber daya di IAMPanduan Pengguna.
IAM	(Tidak disarankan) Gunakan kredensil jangka panjang untuk menandatangani permintaan terprogram ke AWS CLI,, SDKs dan. APIs	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan.Untuk, lihat AWS CLI Mengautentikasi menggunakan IAM kredensil pengguna

Pengguna mana yang membutuhkan akses programatis?	Untuk	Oleh
		di AWS Panduan Pengguna Antarmuka Baris Perintah .Untuk AWS SDKs dan alat, lihat Otentikasi menggunakan kredensi jangka panjang di AWS SDKsdan Panduan Referensi Alat .Untuk AWS APIs, lihat Mengelola kunci akses untuk IAM pengguna di Panduan Pengguna IAM

Menentukan persyaratan

Blok bangunan dasar Amazon Timestream for Influx adalah instans DB. Dalam instance DB, Anda membuat bucket Anda. Instans DB menyediakan alamat jaringan yang disebut titik akhir. Aplikasi Anda menggunakan titik akhir ini untuk terhubung ke instans DB Anda. Anda juga akan mengakses InfluxUI Anda menggunakan titik akhir yang sama dari browser Anda. Saat Anda membuat instans DB, Anda menentukan detail seperti penyimpanan, memori, mesin dan versi database, konfigurasi jaringan, dan keamanan. Anda mengontrol akses jaringan ke instans DB melalui grup keamanan.

Sebelum membuat grup keamanan dan instans DB, Anda harus mengetahui instans DB dan kebutuhan jaringan Anda. Berikut beberapa hal penting yang perlu dipertimbangkan:

- Persyaratan sumber daya — Apa persyaratan memori dan prosesor untuk aplikasi atau layanan Anda? Anda menggunakan pengaturan ini untuk membantu Anda menentukan kelas instans DB apa yang akan digunakan. Untuk spesifikasi tentang kelas instans DB, lihat kelas instans DB.
- VPCdan grup keamanan — Instans DB Anda kemungkinan besar akan berada di cloud pribadi virtual (VPC). Untuk menghubungkan ke instans DB Anda, Anda perlu menyiapkan aturan grup keamanan. Aturan-aturan ini diatur secara berbeda tergantung pada jenis apa yang VPC Anda gunakan dan bagaimana Anda menggunakannya. Misalnya, Anda dapat menggunakan: default VPC atau yang ditentukan penggunaVPC.

  Daftar berikut menjelaskan aturan untuk setiap VPC opsi:

  - Default VPC — Jika AWS akun Anda memiliki default VPC di AWS Wilayah saat ini, yang VPC dikonfigurasi untuk mendukung instans DB. Jika Anda menentukan default VPC saat membuat instans DB, pastikan untuk membuat grup VPC keamanan yang mengotorisasi koneksi dari aplikasi atau layanan ke Amazon Timestream untuk instans DB InfluxDB. Gunakan opsi Grup Keamanan di VPC konsol atau AWS CLI untuk membuat grup VPC keamanan. Untuk informasi selengkapnya, lihat Langkah 3: Membuat grup VPC keamanan.
- Ditentukan pengguna VPC - Jika Anda ingin menentukan yang ditentukan pengguna VPC saat Anda membuat instance DB, perhatikan hal berikut:
  - Pastikan untuk membuat grup VPC keamanan yang mengotorisasi koneksi dari aplikasi atau layanan ke Amazon Timestream untuk instans DB InfluxDB. Gunakan opsi Grup Keamanan di VPC konsol atau AWS CLI untuk membuat grup VPC keamanan. Untuk selengkapnya, lihat Langkah 3: Membuat grup VPC keamanan.
  - VPCHarus memenuhi persyaratan tertentu untuk meng-host instans DB, seperti memiliki setidaknya dua subnet, masing-masing di zona Ketersediaan terpisah. Untuk selengkapnya, lihat Amazon VPC VPCs dan Amazon TimeStream untuk InfluxDB.
- Ketersediaan tinggi - Apakah Anda memerlukan dukungan failover? Di Amazon Timestream untuk InfluxDB, penerapan multi-AZ membuat instans DB primer dan instans DB siaga sekunder di zona Ketersediaan lain untuk dukungan failover. Kami merekomendasikan deployment Multi-AZ untuk beban kerja produksi agar menjaga ketersediaan yang tinggi. Untuk tujuan pengembangan dan pengujian, Anda dapat menggunakan deployment yang bukan Multi-AZ. Untuk informasi selengkapnya, lihat Deployment instans DB Multi-AZ.
- IAMkebijakan — Apakah AWS akun Anda memiliki kebijakan yang memberikan izin yang diperlukan untuk menjalankan Amazon Timestream untuk operasi InfluxDB? Jika Anda terhubung AWS menggunakan IAM kredensyal, IAM akun Anda harus memiliki IAM kebijakan yang memberikan izin yang diperlukan untuk menjalankan Amazon Timestream untuk operasi bidang

kontrol InfluxDB. Untuk informasi selengkapnya, lihat Identity and Access Management untuk Amazon Timestream untuk InfluxDB.

- Buka port — Port TCP /IP apa yang didengarkan database Anda? Firewall di beberapa perusahaan mungkin memblokir koneksi ke port default untuk mesin basis data Anda. Default untuk Timestream untuk InfluxDB adalah 8086.
- AWS Wilayah — AWS Wilayah mana Anda ingin database Anda berada? Memiliki basis data yang dekat dengan aplikasi atau layanan web dapat mengurangi latensi jaringan. Untuk informasi selengkapnya, lihat AWS Wilayah dan Zona Ketersediaan.
- Subsistem disk DB — Apa persyaratan penyimpanan Anda? Amazon Timestream untuk InfluxDB menyediakan tiga konfigurasi untuk itu Jenis penyimpanan IOPS Influx Included::
  - Influx Io Termasuk 3k () IOPS SSD
  - Influx Io Termasuk 12k () IOPS SSD
  - Influx Io Termasuk 25k () IOPS SSD

  Untuk informasi selengkapnya tentang Amazon Timestream untuk penyimpanan InfluxDB, lihat Amazon Timestream untuk penyimpanan instans DB InfluxDB. Setelah memiliki informasi yang Anda perlukan untuk membuat grup keamanan dan instans DB, lanjutkan ke langkah berikutnya.

Berikan akses ke instans DB Anda VPC dengan membuat grup keamanan

VPCgrup keamanan menyediakan akses ke instans DB di file. VPC Grup keamanan tersebut bertindak sebagai firewall untuk instans DB yang terkait, yang mengontrol lalu lintas masuk dan keluar di tingkat instans DB. Instans DB dibuat secara default dengan firewall dan grup keamanan default yang melindungi instans DB.

Sebelum dapat terhubung ke instans DB, Anda harus menambahkan aturan ke grup keamanan yang memungkinkan Anda untuk terhubung. Gunakan informasi jaringan dan konfigurasi untuk membuat aturan yang akan mengizinkan akses ke instans DB Anda.

Misalnya, Anda memiliki aplikasi yang mengakses database pada instans DB Anda di file. VPC Dalam hal ini, Anda harus menambahkan TCP aturan khusus yang menentukan rentang port dan alamat IP yang digunakan aplikasi Anda untuk mengakses database. Jika Anda memiliki aplikasi di EC2 instans Amazon, Anda dapat menggunakan grup keamanan yang Anda siapkan untuk EC2 instans Amazon.

Membuat grup keamanan untuk VPC akses

Untuk membuat grup VPC keamanan, masuk ke AWS Management Console dan pilih VPC.

> **Note**
>
> Pastikan Anda berada di VPC konsol, bukan Amazon Timesteam untuk konsol InfluxDB.

- Di sudut kanan atas AWS Management Console, pilih AWSWilayah tempat Anda ingin membuat grup VPC keamanan dan instans DB. Dalam daftar VPC sumber daya Amazon untuk AWS Wilayah itu, Anda harus melihat setidaknya satu VPC dan beberapa subnet. Jika tidak, Anda tidak memiliki default VPC di AWS Wilayah itu. .
- Pada panel navigasi, pilih Grup Keamanan.
- Pilih Buat grup keamanan.
- Di bagian Detail dasar halaman grup keamanan, masukkan nama grup Keamanan dan Deskripsi. Untuk VPC, pilih yang VPC Anda inginkan untuk membuat instans DB Anda.
- Di bagian Aturan masuk, pilih Tambahkan aturan.
  - Untuk Jenis, pilih Kustom TCP.
  - Untuk Sumber, pilih nama grup Keamanan atau masukkan rentang alamat IP (CIDRnilai) dari tempat Anda mengakses instans DB. Jika Anda memilih IP Saya, pilihan ini akan mengizinkan akses ke instans DB dari alamat IP yang terdeteksi di browser Anda.

  Untuk Sumber, pilih nama grup keamanan atau ketik rentang alamat IP (CIDRnilai) dari tempat Anda mengakses instans DB. Jika Anda memilih IP Saya, pilihan ini akan mengizinkan akses ke instans DB dari alamat IP yang terdeteksi di browser Anda.
- (Opsional) Dalam Aturan keluar, tambahkan aturan untuk lalu lintas keluar. Secara default, semua lalu lintas keluar akan diizinkan.
- Pilih Buat grup keamanan.

Anda dapat menggunakan grup VPC keamanan ini sebagai grup keamanan untuk instans DB Anda saat Anda membuatnya.

> **Note**
>
> Jika Anda menggunakan default VPC, grup subnet default yang mencakup semua subnet dibuat untuk Anda. VPC Ketika Anda membuat instance DB, Anda dapat memilih default eiifccntf VPC dan memilih default untuk DB Subnet Group.

Setelah menyelesaikan persyaratan penyiapan, Anda dapat membuat instans DB menggunakan persyaratan dan grup keamanan Anda. Untuk melakukannya, ikuti petunjuk di Membuat instans DB.

Memulai Timestream untuk InfluxDB

Dalam contoh berikut, Anda dapat mengetahui cara membuat dan menghubungkan ke instans DB menggunakan Amazon Timestream untuk Layanan InfluxDB.

> **Note**
>
> Sebelum dapat membuat atau terhubung ke instans DB, pastikan untuk menyelesaikan tugas dalam Menyiapkan Amazon Timestream untuk InfluxDB.

Topik

- Membuat dan menghubungkan ke Timestream untuk instans InfluxDB
- Membuat Token Operator baru untuk instans InfluxDB Anda

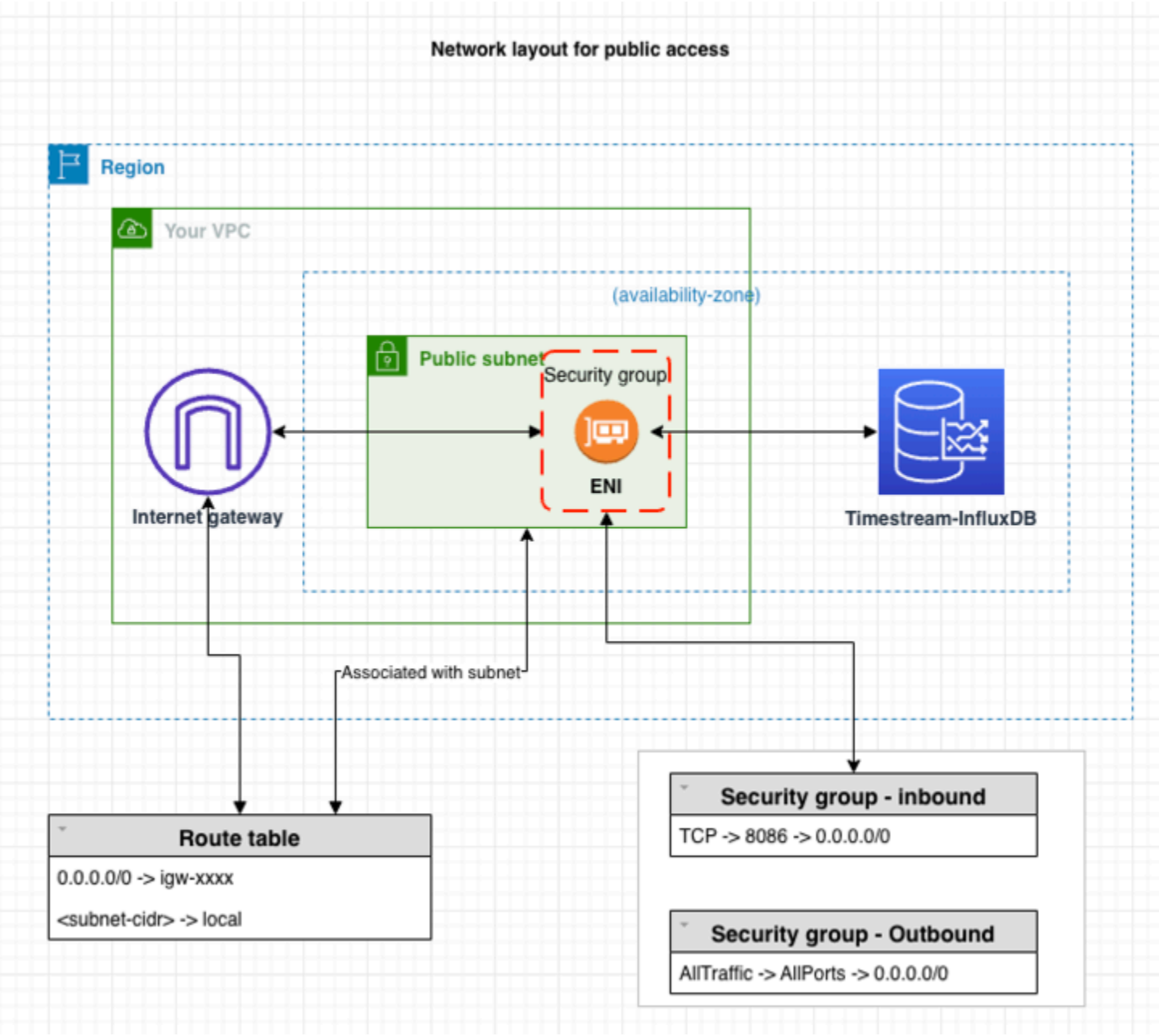
Membuat dan menghubungkan ke Timestream untuk instans InfluxDB

Tutorial ini membuat EC2 instance Amazon dan Amazon Timestream untuk instans DB InfluxDB. Tutorial menunjukkan cara menulis data ke instans DB dari EC2 instance menggunakan klien Telegraf. Sebagai praktik terbaik, tutorial ini membuat instance DB pribadi di cloud pribadi virtual (VPC). Dalam kebanyakan kasus, sumber daya lain dalam hal yang samaVPC, seperti EC2 instance, dapat mengakses instans DB, tetapi sumber daya di luar tidak VPC dapat mengaksesnya.

Setelah Anda menyelesaikan tutorial, ada subnet publik dan pribadi di setiap Availability Zone di AndaVPC. Dalam satu Availability Zone, EC2 instance ada di subnet publik, dan instance DB ada di subnet pribadi.

> **Note**
>
> Tidak ada biaya untuk membuat AWS akun. Namun, dengan menyelesaikan tutorial ini, Anda mungkin dikenakan biaya untuk AWS sumber daya yang Anda gunakan. Anda dapat menghapus sumber daya ini setelah menyelesaikan tutorial jika tidak diperlukan lagi.

Diagram berikut menunjukkan konfigurasi saat aksesibilitas bersifat publik.



> **Warning**
>
> Kami tidak menyarankan menggunakan 0.0.0.0/0 untuk HTTP akses, karena Anda memungkinkan semua alamat IP untuk mengakses instans InfluxDB publik Anda melalui. HTTP Pendekatan ini bahkan tidak dapat diterima untuk waktu yang singkat di lingkungan pengujian. Otorisasi hanya alamat IP tertentu atau rentang alamat untuk mengakses instans InfluxDB Anda menggunakan being HTTP for WebUI atau akses. API

Tutorial ini membuat instance DB yang menjalankan InfluxDB dengan file. AWS Management Console Kami hanya akan fokus pada ukuran instans DB dan pengidentifikasi instans DB. Kami akan menggunakan pengaturan default untuk opsi konfigurasi lainnya. Instans DB yang dibuat oleh contoh ini akan bersifat pribadi.

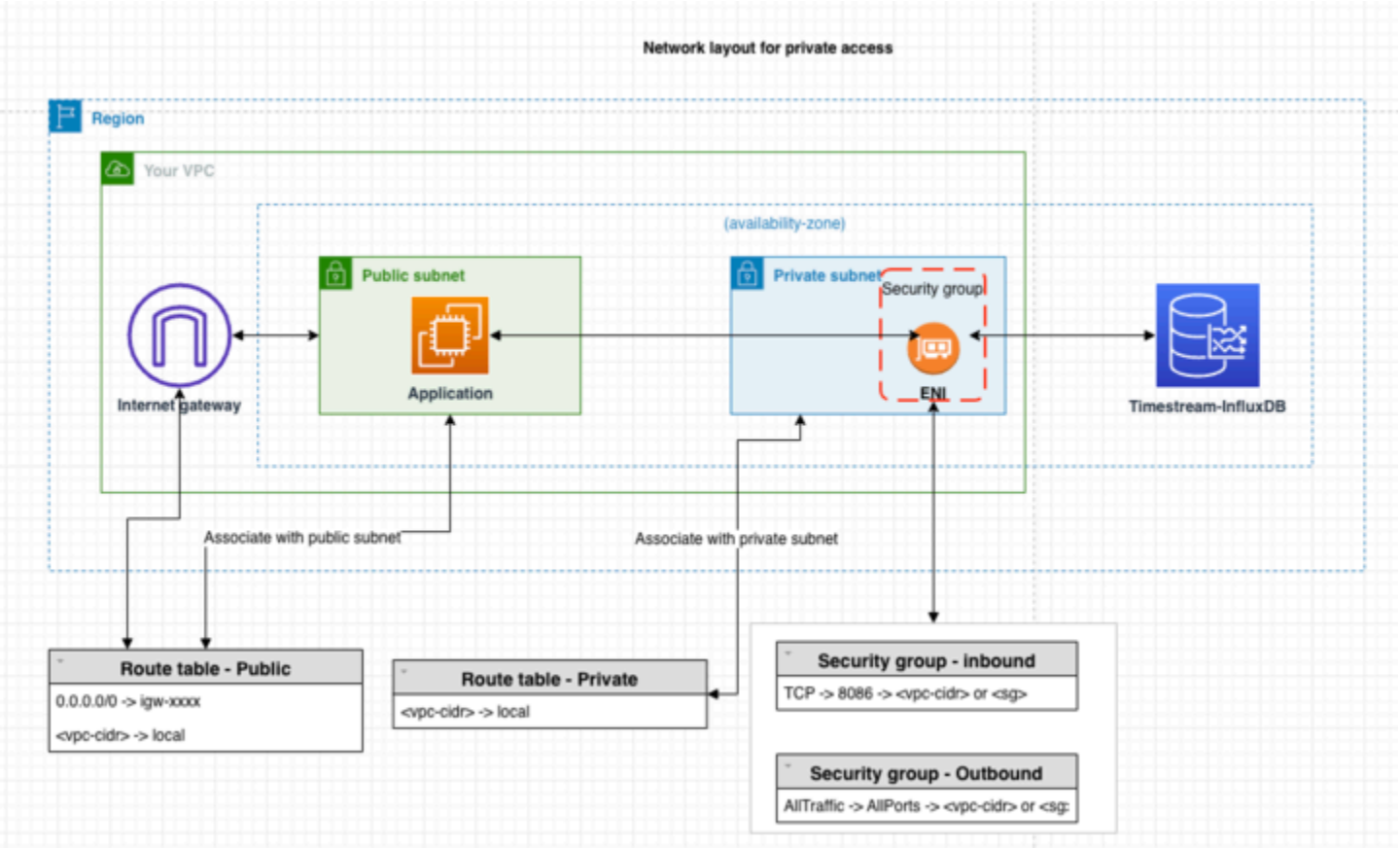
Pengaturan lain yang dapat Anda konfigurasikan termasuk ketersediaan, keamanan, dan pencatatan. Untuk membuat instans DB publik, Anda harus memilih untuk membuat instans Anda "Dapat diakses secara publik" di bagian Konfigurasi Konektivitas. Untuk informasi tentang membuat instans DB, lihat.. Membuat instans DB

Jika instans Anda tidak dapat diakses publik, lakukan hal berikut:

- Buat host pada contoh yang VPC melaluinya Anda dapat melakukan terowongan lalu lintas.
- Siapkan tunneling ssh ke instance. Untuk informasi selengkapnya, lihat penerusan port EC2 instans Amazon dengan Systems Manager AWS
- Agar sertifikat berfungsi, tambahkan baris berikut ke `/etc/hosts` file mesin klien Anda:`127.0.0.1`. Ini adalah DNS alamat contoh Anda.
- Connect ke instans Anda menggunakan nama domain yang sepenuhnya memenuhi syarat, misalnya, https: //< DNS >:8086.

> **Note**
>
> Localhost tidak dapat memvalidasi sertifikat karena localhost bukan bagian dari sertifikat. SAN

Diagram berikut menunjukkan konfigurasi saat aksesibilitas bersifat pribadi:



Prasyarat

Sebelum memulai, selesaikan langkah-langkah di bagian berikut:

- Mendaftar untuk sebuah AWS akun.
- Buat pengguna administratif.

Langkah 1: Buat EC2 instance Amazon

Buat EC2 instance Amazon yang akan Anda gunakan untuk terhubung ke database Anda.

1. Masuk ke AWS Management Console dan buka EC2 konsol Amazon di https://console.aws.amazon.com/ec2/.
2. Di sudut kanan atas AWS Management Console, pilih AWS Wilayah tempat Anda ingin membuat instance. EC2
3. Pilih EC2Dasbor, lalu pilih Launch instance.

4. Saat halaman Launch an instance terbuka, pilih pengaturan berikut pada halaman Luncurkan instance.

   a. Di bawah Nama dan tag, untuk Nama, masukkan ec2-database-connect.

   b. Di bawah Gambar Aplikasi dan OS (Gambar Mesin Amazon), pilih Amazon Linux, lalu pilih Amazon Linux 2023AMI. Biarkan default untuk pilihan lainnya.

   c. Pada Jenis instans, pilih t2.micro.

   d. Pada Pasangan kunci (login), pilih Nama pasangan kunci untuk menggunakan pasangan kunci yang ada. Untuk membuat key pair baru untuk EC2 instance Amazon, pilih Create new key pair dan kemudian gunakan jendela Create key pair untuk membuatnya. Untuk informasi selengkapnya tentang membuat key pair baru, lihat Membuat key pair di Panduan EC2 Pengguna Amazon untuk Instans Linux.

   e. Untuk Izinkan SSH lalu lintas di pengaturan Jaringan, pilih sumber SSH koneksi ke EC2 instance. Anda dapat memilih IP Saya jika alamat IP yang ditampilkan benar untuk SSH koneksi. Jika tidak, Anda dapat menentukan alamat IP yang akan digunakan untuk menyambung ke EC2 instance dalam VPC menggunakan Secure Shell (SSH). Untuk menentukan alamat IP publik Anda, di jendela atau tab browser yang berbeda, Anda dapat menggunakan layanan di https://checkip.amazonaws.com. Contoh alamat IP adalah 192.0.2.1/32. Dalam banyak kasus, Anda mungkin terhubung melalui penyedia layanan internet (ISP) atau dari belakang firewall Anda tanpa alamat IP statis. Jika demikian, tentukan rentang alamat IP yang digunakan oleh komputer klien.

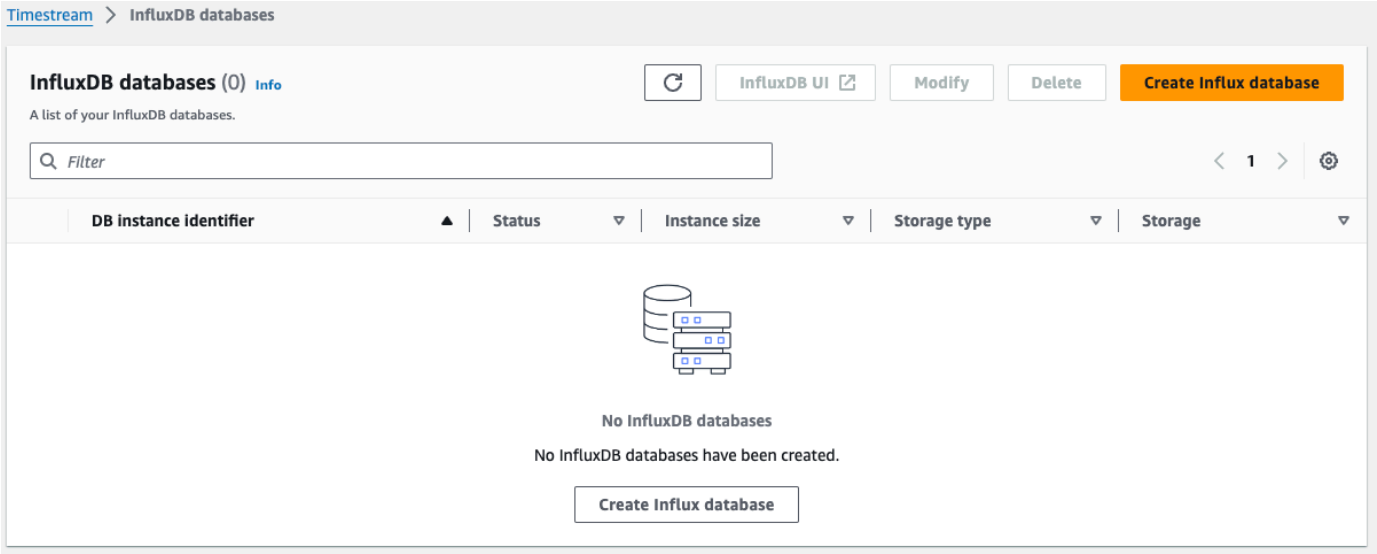
   > ⚠ Warning
   >
   > Kami tidak menyarankan menggunakan 0.0.0.0/0 untuk SSH akses, karena Anda memungkinkan semua alamat IP untuk mengakses instans publik Anda menggunakan. EC2 SSH Pendekatan ini bahkan tidak dapat diterima untuk waktu yang singkat di lingkungan pengujian, hanya mengotorisasi alamat IP tertentu atau rentang alamat untuk mengakses EC2 instance Anda menggunakan. SSH

Langkah 2: Buat instans DB InfluxDB

Blok bangunan dasar Amazon Timestream untuk InfluxDB adalah instans DB. Lingkungan ini adalah tempat Anda menjalankan database InfluxDB Anda.

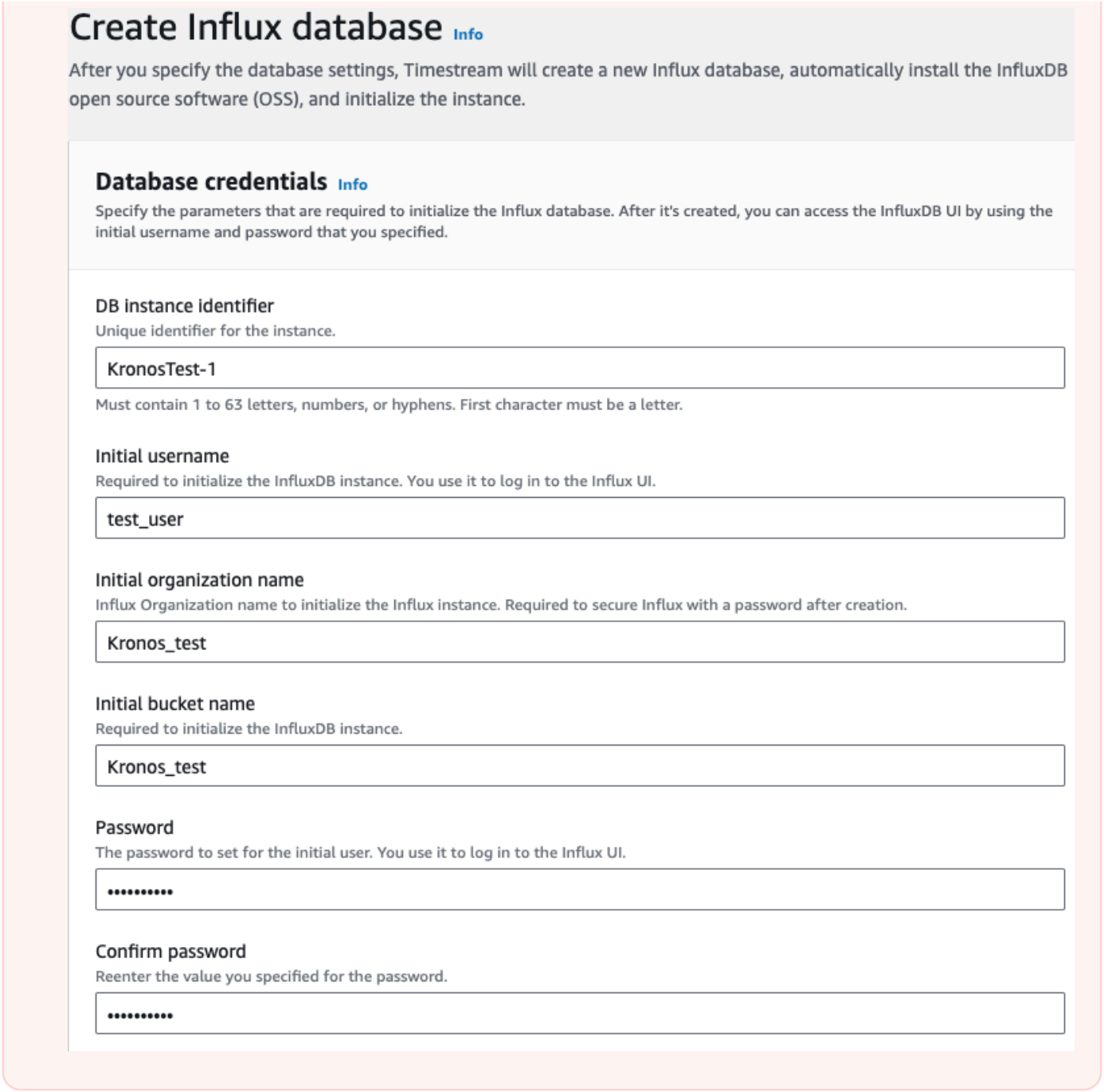
Dalam contoh ini, Anda akan membuat instance DB yang menjalankan mesin database InfluxDB dengan kelas instans db.influx.large DB.

1. Masuk ke AWS Management Console dan buka Amazon Timestream untuk konsol InfluxDB di. https://console.aws.amazon.com/timestream/
2. Di sudut kanan atas Amazon Timestream untuk konsol InfluxDB, pilih AWS Wilayah tempat Anda ingin membuat instans DB.
3. Di panel navigasi, pilih Database InfluxDB.
4. Pilih Buat database Influx.

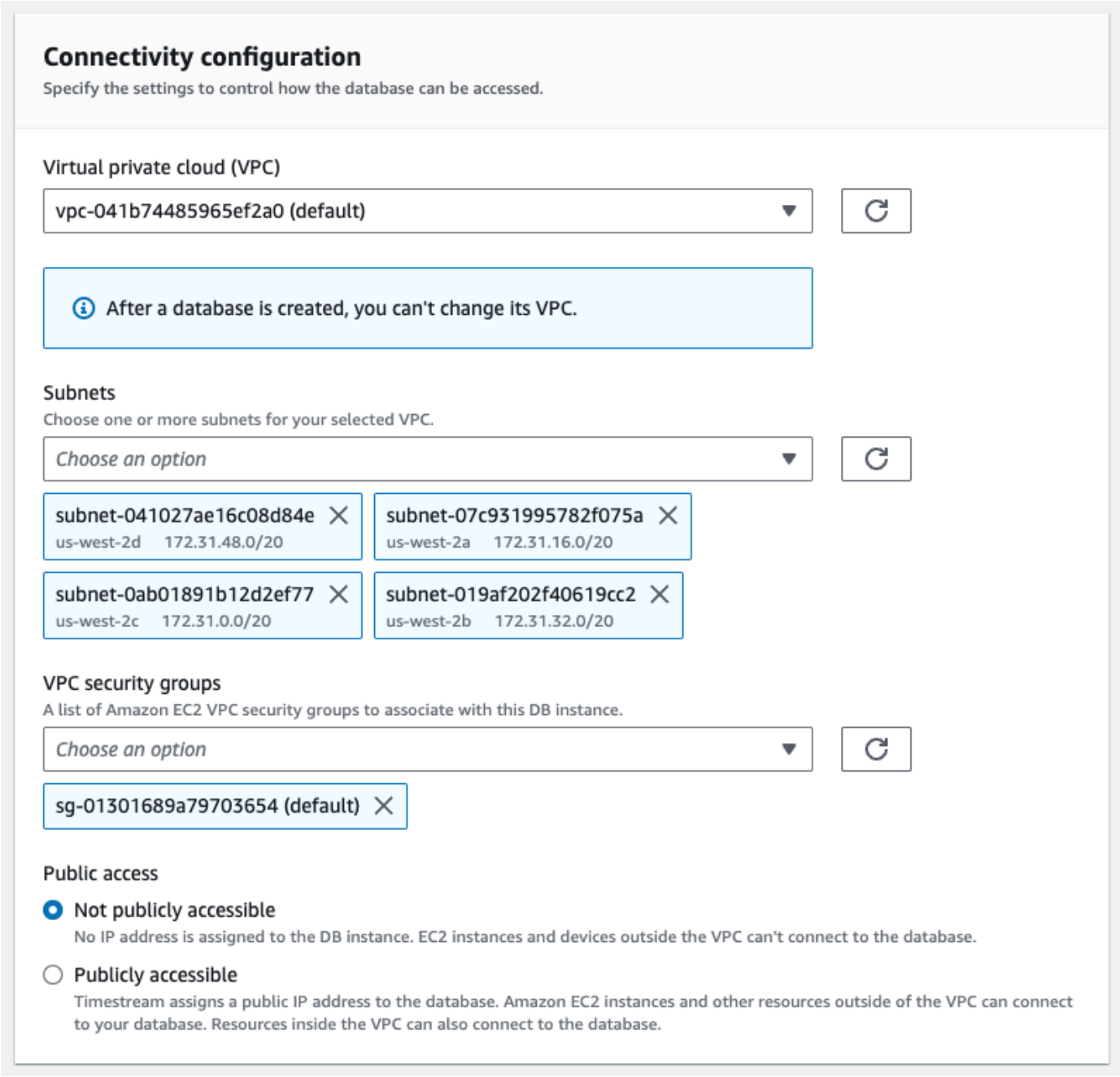


5. Untuk DB Instance Identifier, masukkan KronosTest -1.
6. Berikan parameter konfigurasi dasar InfluxDB: Nama Pengguna, Organisasi, Nama Bucket, dan Kata Sandi.

> **Important**
>
> Anda tidak akan dapat melihat kata sandi pengguna lagi. Anda tidak akan dapat mengakses instans Anda dan mendapatkan token operator tanpa kata sandi Anda. Jika Anda tidak mencatatnya, Anda mungkin harus mengubahnya. Lihat Membuat Token Operator baru untuk instans InfluxDB Anda.
>
> Jika Anda perlu mengubah kata sandi pengguna setelah instans DB tersedia, Anda dapat memodifikasi instans DB untuk melakukannya. Untuk informasi selengkapnya tentang cara mengubah instans DB, lihat Memperbarui instans DB.



7. Untuk Kelas Instance DB, pilih db.influx.large.
8. Untuk Kelas Penyimpanan DB, pilih masuknya IOPS Termasuk 3K.
9. Konfigurasikan log Anda. Untuk informasi selengkapnya, lihat Pengaturan untuk melihat log InfluxDB pada Instans Timestream Influxdb.
10. Di bagian Konfigurasi Konektivitas, pastikan instans InfluxDB Anda berada di subnet yang sama dengan instans yang baru dibuat. EC2



11. Pilih Buat database Influx.
12. Dalam daftar Database, pilih nama instans InfluxDB baru Anda untuk menampilkan detailnya. Instans DB memiliki status Creating sampai siap digunakan.

Anda dapat terhubung ke instans DB saat status berubah menjadi Tersedia. Tergantung pada kelas instans DB dan jumlah penyimpanan, diperlukan waktu hingga 20 menit sebelum instans baru tersedia.

> **Important**
>
> Saat ini, Anda tidak dapat mengubah konfigurasi komputasi (Jenis instans) dan Penyimpanan (Jenis Penyimpanan) dari instance yang ada.

Langkah 3: Kirim data Telegraf ke instans InfluxDB Anda

Anda sekarang dapat mulai mengirim data telemetri ke instans DB InfluxDB Anda menggunakan agen Telegraf. Dalam contoh ini, Anda akan menginstal dan mengonfigurasi agen Telegraf untuk mengirim metrik kinerja ke instans DB InfluxDB Anda.

1. Temukan titik akhir (DNSnama) dan nomor port untuk instans DB Anda.

   a. Masuk ke Konsol AWS Manajemen dan buka konsol Amazon Timestream di. https://console.aws.amazon.com/timestream/

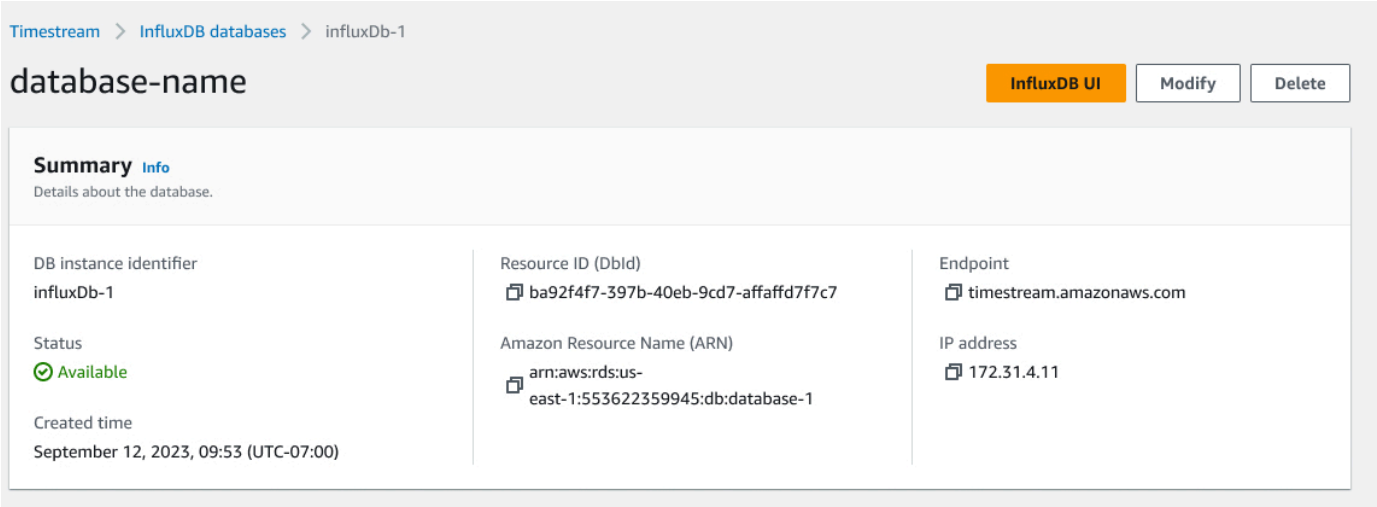
   b. Di sudut kanan atas konsol Amazon Timestream, pilih AWS Region untuk instans DB.

   c. Di panel navigasi, pilih Database InfluxDB.

   d. Pilih nama instans DB InfluxDB untuk menampilkan detailnya.

   e. Pada bagian Ringkasan, salin titik akhir. Perhatikan juga nomor port. Anda memerlukan titik akhir dan nomor port untuk terhubung ke instans DB (nomor port default untuk InfluxDB adalah 8086).

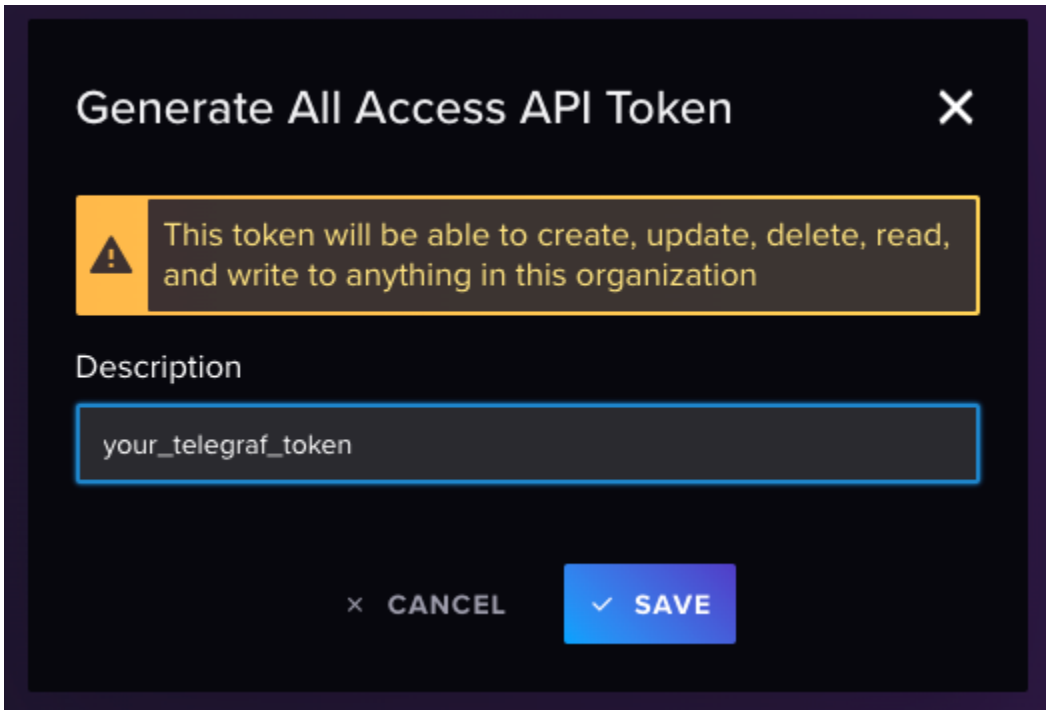
2. Selanjutnya, pilih InfluxDB UI.



3. Ini akan membuka jendela browser baru di mana Anda akan melihat prompt login. Masukkan kredenal yang Anda gunakan sebelumnya untuk membuat instans InfluxDB Db Anda.

4. Di panel navigasi, klik pada Panah dan pilih APIToken.
5. Untuk pengujian ini, buat All Access Token.

> **Note**
>
> Untuk skenario produksi, kami sarankan untuk membuat token dengan akses khusus ke bucket yang diperlukan yang dibuat untuk kebutuhan Telegraf tertentu.



6. Token Anda akan muncul di layar.

> **Important**
>
> Pastikan untuk menyalin dan menyimpan Token karena Anda tidak akan dapat menampilkannya lagi.

7. Hubungkan ke EC2 instans yang Anda buat sebelumnya dengan mengikuti langkah-langkah di Connect to Linux Anda di Panduan EC2 Pengguna Amazon untuk Instans Linux.

   Kami menyarankan Anda terhubung ke EC2 instans Anda menggunakanSSH. Jika utilitas SSH klien diinstal pada Windows, Linux, atau Mac, Anda dapat terhubung ke instance menggunakan format perintah berikut:

```
ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
```

Misalnya, asumsikan bahwa `ec2-database-connect-key-pair.pem` disimpan `/dir1` di Linux, dan publik IPv4 DNS untuk EC2 instance Andaec2-12-345-678-90.compute-1.amazonaws.com. SSHPerintah Anda akan terlihat sebagai berikut:

```
ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
user@ec2-12-345-678-90.compute-1.amazonaws.com
```

8. Dapatkan telegraf versi terbaru yang diinstal pada instans Anda. Untuk melakukan ini, gunakan perintah berikut:

```
cat <<EOF | sudo tee /etc/yum.repos.d/influxdata.repo
[influxdata]
name = InfluxData Repository - Stable
baseurl = https://repos.influxdata.com/stable/\$basearch/main
enabled = 1
gpgcheck = 1
gpgkey = https://repos.influxdata.com/influxdata-archive_compat.key
EOF

sudo yum install telegraf
```

9. Konfigurasikan instance Telegraf Anda.

> **Note**
>
> Jika telegraf.conf tidak ada atau berisi `timestream` bagian, Anda dapat membuatnya dengan:
>
> ```
> telegraf –section-filter agent:inputs:outputs –input-filter cpu:mem –output-
> filter timestream config > telegraf.conf
> ```

a. Edit file konfigurasi yang biasanya terletak di`/etc/telegraf`.

```
sudo nano /etc/telegraf/telegraf.conf
```

b. Konfigurasikan input dasar untukCPU, MEM danDISK.

```
[[inputs.cpu]]
  percpu = true
  totalcpu = true
  collect_cpu_time = false
  report_active = false

[[inputs.mem]]

[[inputs.disk]]
  ignore_fs = ["tmpfs", "devtmpfs", "devfs"]
```

c. Konfigurasikan plug Output untuk mengirim data ke instans DB InfluxDB Anda dan menyimpan perubahan Anda.

```
[[outputs.influxdb_v2]]
   urls = ["https://us-west-2-1.aws.cloud2.influxdata.com"]
   token = "<your_telegraf_token"
   organization = "your_org"
   bucket = "your_bucket"
   timeout = "5s"
```

d. Konfigurasikan target Timestream.

```
# Configuration for sending metrics to Amazon Timestream.
[[outputs.timestream]]

  ## Amazon Region and credentials
  region = "us-east-1"
  access_key = "<AWS key here>"
  secret_key = "<AWS secret key here>"
  database_name = "<timestream database name>" # needs to exist

  ## Specifies if the plugin should describe t start.
  describe_database_on_start = false
  mapping_mode = "multi-table" # allows multible tables for each input metrics

  create_table_if_not_exists = true
  create_table_magnetic_store_retention_period_in_days = 365
  create_table_memory_store_retention_period_in_hours = 24

  use_multi_measure_records = true # Important to use multi-measure records
```

```
  measure_name_for_multi_measure_records = "telegraf_measure"
  max_write_go_routines = 25
```

10. Aktifkan dan mulai layanan Telegraf.

```
$ sudo systemctl enable telegraf
$ sudo systemctl start telegraf
```

Langkah 4: Hapus EC2 instans Amazon dan instans DB InfluxDB

Setelah Anda menjelajahi data yang dihasilkan Telegraf menggunakan instans DB InfluxDB Anda dengan InfluxUI, hapus instans DB InfluxDB Anda EC2 dan InfluxDB Anda sehingga Anda tidak lagi dikenakan biaya untuk itu.

Untuk menghapus EC2 instance:

1. Masuk ke AWS Management Console dan buka EC2 konsol Amazon di https://console.aws.amazon.com/ec2/.
2. Di panel navigasi, pilih Instans.
3. Pilih EC2 instance, pilih Instance state, dan Terminate instance.
4. Pilih Akhiri saat diminta untuk konfirmasi.

Untuk informasi selengkapnya tentang menghapus EC2 instance, lihat Mengakhiri instans Anda di EC2 Panduan Pengguna Amazon.

Untuk menghapus instans DB tanpa snapshot DB akhir:

1. Masuk ke AWS Management Console dan buka Amazon Timestream untuk konsol InfluxDB di. https://console.aws.amazon.com/timestream/
2. Di panel navigasi, pilih Database InfluxDB.
3. Pilih instans DB yang ingin Anda hapus.
4. Untuk Tindakan, pilih Hapus.
5. Lengkapi pengakuan dan pilih Hapus.

(Opsional) Hubungkan ke instans DB Anda menggunakan Amazon Managed Grafana

Anda dapat menggunakan Grafana Terkelola Amazon untuk membuat dasbor dan memantau kinerja EC2 instans menggunakan Amazon Timestream untuk InfluxDB. Grafana yang Dikelola Amazon adalah layanan yang dikelola sepenuhnya untuk Grafana, platform analitik sumber terbuka populer yang memungkinkan Anda melakukan kueri, memvisualisasikan, dan memberi tahu metrik, log, dan jejak Anda.

Membuat Token Operator baru untuk instans InfluxDB Anda

Jika Anda perlu mendapatkan Token Operator untuk instans InfluxDB baru Anda, lakukan langkah-langkah berikut:

1. Untuk mengubah token operator Anda, sebaiknya gunakan InfluxCLI. Untuk instruksi, silakan lihat: Instal dan gunakan arus masuk CLI.
2. Konfigurasikan yang akan Anda CLI gunakan `--username-password` agar dapat membuat operator:

   ```
   influx config create --config-name CONFIG_NAME1  --host-url "https://
   yourinstanceid.eu-central-1.timestream-influxdb.amazonaws.com:8086" --org [YOURORG]
     --username-password [YOURUSERNAME] --active
   ```

3. Buat token operator baru Anda. Anda akan diminta kata sandi Anda untuk mengonfirmasi langkah ini.

   ```
   influx auth create --org [YOURORG] --operator
   ```

> **Important**
>
> Setelah token operator baru dibuat, Anda perlu memperbarui klien apa pun yang saat ini menggunakan yang lama.

Migrasi data dari InfluxDB yang dikelola sendiri ke Timestream untuk InfluxDB

Skrip migrasi Influx adalah skrip Python yang memigrasikan data antara instans OSS InfluxDB, apakah instance tersebut dikelola oleh atau tidak. AWS

InfluxDB adalah database time series. InfluxDB berisi poin, yang berisi sejumlah pasangan kunci-nilai dan stempel waktu. Ketika poin dikelompokkan berdasarkan pasangan kunci-nilai, mereka membentuk seri. Sebuah seri dikelompokkan oleh pengenal string yang disebut pengukuran. InfluxDB sering digunakan untuk pemantauan operasi, IOT data, dan analitik. Bucket adalah sejenis wadah dalam InfluxDB untuk menyimpan data. AWS-Managed InfluxDB adalah InfluxDB dalam ekosistem. AWS InfluxDB menyediakan InfluxDB v2 API untuk mengakses data dan membuat perubahan pada database. InfluxDB v2 API adalah apa yang digunakan skrip migrasi Influx untuk memigrasikan data.

- Skrip migrasi Influx dapat memigrasikan bucket dan metadatanya, memigrasikan semua bucket dari semua organisasi, atau melakukan migrasi penuh, yang menggantikan semua data pada instance tujuan.
- Skrip mencadangkan data dari instance sumber secara lokal, pada sistem apa pun yang mengeksekusi skrip, kemudian mengembalikan data ke instance tujuan. Data disimpan dalam kode> influxdb-backup-<timestamp></timestamp> direktori, satu untuk setiap migrasi.
- Skrip ini menyediakan sejumlah opsi dan konfigurasi termasuk memasang bucket S3 untuk membatasi penggunaan penyimpanan lokal selama migrasi dan memilih organisasi mana yang akan digunakan selama migrasi.

Topik

Persiapan

Migrasi data untuk InfluxDB dilakukan dengan skrip Python yang menggunakan fitur InfluxDB dan CLI InfluxDB v2. API Eksekusi skrip migrasi memerlukan konfigurasi lingkungan berikut:

- Versi yang Didukung: Versi minimum 2.3 InfluxDB dan Influx CLI didukung.
- Variabel Lingkungan Token
  - Buat variabel lingkungan yang `INFLUX_SRC_TOKEN` berisi token untuk instans InfluxDB sumber Anda.
  - Buat variabel lingkungan yang `INFLUX_DEST_TOKEN` berisi token untuk instans InfluxDB tujuan Anda.

- Python 3
  - Periksa instalasi:`python3 --version`.
  - Jika tidak diinstal, instal dari situs web Python. Diperlukan versi minimum 3.7. Pada Windows, alias Python 3 default hanya python.
  - Permintaan modul Python diperlukan. Instal dengan: `shell python3 -m pip install requests`
  - TTheModul Python influxdb_client diperlukan. Instal dengan: `shell python3 -m pip install influxdb_client`
- InfluxDB CLI
  - Konfirmasikan instalasi:`influx version`.
  - Jika tidak diinstal, ikuti panduan instalasi dalam dokumentasi InfluxDB.

    Tambahkan masuknya ke $ PATH Anda.
- Alat Pemasangan S3 (Opsional)

  Saat pemasangan S3 digunakan, semua file cadangan disimpan dalam bucket S3 yang ditentukan pengguna. Pemasangan S3 dapat berguna untuk menghemat ruang pada mesin yang mengeksekusi atau ketika file cadangan perlu dibagikan. Jika pemasangan S3 tidak digunakan, dengan menghilangkan `--s3-bucket` opsi, maka `influxdb-backup-<millisecond timestamp>` direktori lokal akan dibuat untuk menyimpan file cadangan di direktori yang sama dengan skrip yang dijalankan.

  Untuk Linux: mountpoint-s3.

  Untuk Windows: rclone (Konfigurasi rclone sebelumnya diperlukan).
- Ruang Disk
  - Proses migrasi secara otomatis membuat direktori unik untuk menyimpan set file cadangan dan mempertahankan direktori cadangan ini baik di S3 atau pada sistem file lokal, tergantung pada argumen program yang disediakan.
  - Pastikan ada ruang disk yang cukup untuk cadangan basis data, idealnya menggandakan ukuran database InfluxDB yang ada jika Anda memilih untuk menghilangkan `--s3-bucket` opsi dan menggunakan penyimpanan lokal untuk pencadangan dan pemulihan.
  - Periksa ruang dengan `df -h (UNIX/Linux)` atau dengan memeriksa properti drive pada Windows.
- Koneksi Langsung

Pastikan koneksi jaringan langsung ada antara sistem yang menjalankan skrip migrasi dan sistem sumber dan tujuan. `influx ping --host <host>`adalah salah satu cara untuk memverifikasi koneksi langsung.

Cara Menggunakan Script

Contoh sederhana menjalankan skrip adalah perintah:

```
python3 influx_migration.py --src-host <source host> --src-bucket <source bucket> --
dest-host <destination host>
```

Yang memigrasikan satu ember.

Semua opsi dapat dilihat dengan menjalankan:

```
python3 influx_migration.py -h
```

Penggunaan

```
shell   influx_migration.py [-h] [--src-bucket SRC_BUCKET] [--dest-bucket DEST_BUCKET]
 [--src-host SRC_HOST] --dest-host DEST_HOST [--full] [--confirm-full] [--src-org
 SRC_ORG] [--dest-org DEST_ORG] [--csv] [--retry-restore-dir RETRY_RESTORE_DIR] [--dir-
name DIR_NAME] [--log-level LOG_LEVEL] [--skip-verify] [--s3-bucket S3_BUCKET]
```

Pilihan

- -confirm-full (opsional): Menggunakan `--full` tanpa `--csv` akan menggantikan semua token, pengguna, bucket, dasbor, dan data nilai kunci lainnya di database tujuan dengan token, pengguna, bucket, dasbor, dan data nilai kunci lainnya di database sumber. `--full`dengan `--csv` hanya memigrasikan semua metadata bucket dan bucket, termasuk organisasi bucket. Opsi ini (`--confirm-full`) akan mengonfirmasi migrasi penuh dan melanjutkan tanpa masukan pengguna. Jika opsi ini tidak disediakan, dan `--full` telah disediakan dan `--csv` tidak disediakan, maka skrip akan berhenti sementara untuk eksekusi dan menunggu konfirmasi pengguna. Ini adalah tindakan kritis, lanjutkan dengan hati-hati. Default ke false.
- -csv (opsional): Apakah akan menggunakan file csv untuk mencadangkan dan memulihkan. Jika `--full` diteruskan juga maka semua bucket yang ditentukan pengguna di semua organisasi

akan dimigrasikan, bukan bucket sistem, pengguna, token, atau dasbor. Jika organisasi tunggal diinginkan untuk semua bucket di server tujuan alih-alih organisasi sumber yang sudah ada, gunakan. `--dest-org`

- -dest-bucket DEST _ BUCKET (opsional): Nama bucket InfluxDB di server tujuan, tidak boleh berupa bucket yang sudah ada. Default untuk nilai `--src-bucket` atau None jika `--src-bucket` tidak disediakan.
- -dest-host DEST _ HOST: Host untuk server tujuan. Contoh: http://localhost:8086.
- -dest-org DEST _ ORG (opsional): Nama organisasi untuk mengembalikan bucket di server tujuan. Jika ini dihilangkan, maka semua bucket yang dimigrasi dari server sumber akan mempertahankan organisasi aslinya dan bucket yang dimigrasi mungkin tidak terlihat di server tujuan tanpa membuat dan mengalihkan organisasi. Nilai ini akan digunakan dalam semua bentuk pemulihan baik satu bucket, migrasi penuh, atau migrasi apa pun menggunakan file csv untuk pencadangan dan pemulihan.
- -dir-name DIR _ NAME (opsional): Nama direktori cadangan yang akan dibuat. Default ke `influxdb-backup-<timestamp>`. Pasti belum ada.
- -full (opsional): Apakah akan melakukan pemulihan penuh, mengganti semua data di server tujuan dengan semua data dari server sumber dari semua organisasi, termasuk semua data nilai kunci seperti token, dasbor, pengguna, dll. Mengesampingkan dan. `--src-bucket --dest-bucket` Jika digunakan dengan`--csv`, hanya memigrasikan data dan metadata bucket. Default ke false.
- h, --help: Menampilkan pesan bantuan dan keluar.
- -log-level LOG _ LEVEL (opsional): Level log yang akan digunakan selama eksekusi. Pilihannya adalah debug, kesalahan, dan info. Default ke info.
- - retry-restore-dir RETRY _ RESTORE _ DIR (opsional): Direktori yang akan digunakan untuk pemulihan ketika pemulihan sebelumnya gagal, akan melewati pencadangan dan pembuatan direktori, akan gagal jika direktori tidak ada, dapat berupa direktori dalam ember S3. Jika restorasi gagal, jalur direktori cadangan yang dapat digunakan untuk restorasi akan ditunjukkan relatif terhadap direktori saat ini. Ember S3 akan dalam bentuk. `influxdb-backups/<s3 bucket>/<backup directory>` Nama direktori cadangan default adalah`influxdb-backup-<timestamp>`.
- -s3-bucket S3_ BUCKET (opsional): Nama bucket S3 yang akan digunakan untuk menyimpan file cadangan. Di Linux ini hanyalah nama bucket S3, sepertimy`-bucket`, diberikan `AWS_ACCESS_KEY_ID` dan variabel `AWS_SECRET_ACCESS_KEY` lingkungan telah ditetapkan atau `${HOME}/.aws/credentials` ada. Di Windows, ini adalah nama remote dan bucket yang `rclone` dikonfigurasi, sepertimy`-remote:my-bucket`. Semua file cadangan akan tertinggal di

bucket S3 setelah migrasi di `influxdb-backups-<timestamp>` direktori yang dibuat. Sebuah direktori mount sementara bernama `influx-backups` akan dibuat dalam direktori dari mana script ini dijalankan. Jika tidak disediakan, maka semua file cadangan akan disimpan secara lokal di `influxdb-backups-<timestamp>` direktori yang dibuat dari tempat skrip ini dijalankan.

- -lewati verifikasi (opsional): Lewati verifikasi sertifikat. TLS
- -src-bucket SRC _ BUCKET (opsional): Nama bucket InfluxDB di server sumber. Jika tidak disediakan, maka `--full` harus disediakan.
- -src-host SRC _ HOST (opsional): Host untuk server sumber. Default ke http://localhost:8086.

Seperti disebutkan sebelumnya, `mountpoint-s3` dan `rclone --s3-bucket` diperlukan jika akan digunakan, tetapi dapat diabaikan jika pengguna tidak memberikan nilai untuk`--s3-bucket`, dalam hal ini file cadangan akan disimpan dalam direktori unik secara lokal.

Ikhtisar Migrasi

Setelah memenuhi prasyarat:

1. Jalankan Skrip Migrasi: Menggunakan aplikasi terminal pilihan Anda, jalankan skrip Python untuk mentransfer data dari instance InfluxDB sumber ke instans InfluxDB tujuan.
2. Berikan Kredensial: Berikan alamat host dan port sebagai CLI opsi.
3. Verifikasi Data: Pastikan data ditransfer dengan benar oleh:

    a. Menggunakan UI InfluxDB dan memeriksa bucket.

    b. Daftar ember dengan`influx bucket list -t <destination token> --host <destination host address> --skip-verify`.

    c. Menggunakan `influx v1 shell -t <destination token> --host <destination host address> --skip-verify` dan menjalankan `SELECT * FROM <migrated bucket>.<retention period>.<measurement name> LIMIT 100 to view contents of a bucket or SELECT COUNT(*) FROM <migrated bucket>.<retention period>.<measurment name>` untuk memverifikasi jumlah catatan yang benar telah dimigrasikan.

Example Contoh jalankan

1. Buka aplikasi terminal pilihan Anda dan pastikan prasyarat yang diperlukan diinstal dengan benar:

```
~ > python3 --version
Python 3.11.5
~ > influx version
Influx CLI 2.7.3 (git: 8b962c7e75) build_date: 2023-04-28T14:22:49Z
~ > s3fs --version
Amazon Simple Storage Service File System V1.92 (commit:unknown) with GnuTLS(gcrypt)
Copyright (C) 2010 Randy Rizun <rrizun@gmail.com>
License GPL2: GNU GPL version 2 <https://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
~ >
```

2. Arahkan ke skrip migrasi:

```
~ > cd sample-code/influxdb-sample/migration/influxdb
~/sample-code/influxdb-sample/migration/influxdb > ls *.py
influx_migration.py
~/sample-code/influxdb-sample/migration/influxdb >
```

3. Siapkan informasi berikut:

   a. Nama bucket sumber yang akan dimigrasikan.

   b. (Opsional) Pilih nama bucket baru untuk bucket yang dimigrasi di server tujuan.

   c. Token root untuk instance masuknya sumber dan tujuan.

   d. Alamat host dari instance masuknya sumber dan tujuan.

   e. (Opsional) nama bucket S3 dan kredensional; AWS Command Line Interface kredensional harus ditetapkan dalam variabel lingkungan OS.

```
# AWS credentials (for timestream testing)
    export AWS_ACCESS_KEY_ID="xxx"
    export AWS_SECRET_ACCESS_KEY="xxx"
```

   f. Bangun perintah sebagai:

```
python3 influx_migration.py --src-bucket [source-bucket-name]  --dest-bucket
 [dest-bucket-name] --src-host [source host] --dest-host [dest host] --s3-
bucket [s3 bucket name](optional) --log-level debug
```

   g. Jalankan skrip:

```
~/sample-code/influxdb-sample/migration/influxdb > python3 influx_migration.py --src-bucket primary-bucket --src-host $INFLUXDB_1_HOST --dest-host $KRO
NOS_HOST --dest-bucket new-bucket-name
```

   h. Tunggu skrip selesai dieksekusi.

i. Periksa bucket yang baru dimigrasi untuk mengetahui integritas data. `performance.txt` File ini, terletak di bawah direktori yang sama tempat skrip dijalankan, berisi beberapa informasi dasar tentang berapa lama setiap langkah diambil.

Skenario migrasi

Example Contoh 1: Migrasi Sederhana Menggunakan Penyimpanan Lokal

Anda ingin memigrasikan satu bucket, primary bucket, dari server sumber (`http://localhost:8086`) ke server tujuan. (`http://dest-server-address:8086`)

Setelah memastikan Anda memiliki TCP akses (untuk HTTP akses) ke kedua mesin yang menghosting instans InfluxDB pada port 8086 dan Anda memiliki token sumber dan tujuan dan telah menyimpannya sebagai variabel lingkungan `INFLUX_SRC_TOKEN` dan`INFLUX_DEST_TOKEN`, masing-masing, untuk keamanan tambahan:

```
python3 influx_migration.py --src-bucket primary-bucket --src-host http://
localhost:8086 --dest-host http://dest-server-address:8086
```

Outputnya akan serupa dengan yang berikut ini:

```
INFO: influx_migration.py: Backing up bucket data and metadata using the InfluxDB CLI
2023/10/26 10:47:15 INFO: Downloading metadata snapshot
2023/10/26 10:47:15 INFO: Backing up TSM for shard 1
2023/10/26 10:47:15 INFO: Backing up TSM for shard 8245
2023/10/26 10:47:15 INFO: Backing up TSM for shard 8263
[More shard backups . . .]
2023/10/26 10:47:20 INFO: Backing up TSM for shard 8240
2023/10/26 10:47:20 INFO: Backing up TSM for shard 8268
2023/10/26 10:47:20 INFO: Backing up TSM for shard 2
INFO: influx_migration.py: Restoring bucket data and metadata using the InfluxDB CLI
2023/10/26 10:47:20 INFO: Restoring bucket "96c11c8876b3c016" as "primary-bucket"
2023/10/26 10:47:21 INFO: Restoring TSM snapshot for shard 12772
2023/10/26 10:47:22 INFO: Restoring TSM snapshot for shard 12773
[More shard restores . . .]
2023/10/26 10:47:28 INFO: Restoring TSM snapshot for shard 12825
2023/10/26 10:47:28 INFO: Restoring TSM snapshot for shard 12826
INFO: influx_migration.py: Migration complete
```

Direktori `influxdb-backup-<timestamp>` akan dibuat dan disimpan di direktori tempat skrip dijalankan, berisi file cadangan.

Example Contoh 2: Migrasi Penuh Menggunakan Penyimpanan Lokal dan Pencatatan Debug

Sama seperti di atas kecuali Anda ingin memigrasikan semua bucket, token, pengguna, dan dasbor, menghapus bucket di server tujuan, dan melanjutkan tanpa konfirmasi pengguna tentang migrasi database lengkap dengan menggunakan opsi. `--confirm-full` Anda juga ingin melihat pengukuran kinerjanya sehingga Anda mengaktifkan logging debug.

```
python3 influx_migration.py --full --confirm-full --src-host http://localhost:8086 --
dest-host http://dest-server-address:8086 --log-level debug
```

Outputnya akan serupa dengan yang berikut ini:

```
INFO: influx_migration.py: Backing up bucket data and metadata using the InfluxDB CLI
2023/10/26 10:55:27 INFO: Downloading metadata snapshot
2023/10/26 10:55:27 INFO: Backing up TSM for shard 6952
2023/10/26 10:55:27 INFO: Backing up TSM for shard 6953
[More shard backups . . .]
2023/10/26 10:55:36 INFO: Backing up TSM for shard 8268
2023/10/26 10:55:36 INFO: Backing up TSM for shard 2
DEBUG: influx_migration.py: backup started at 2023-10-26 10:55:27 and took 9.41 seconds
 to run.
INFO: influx_migration.py: Restoring bucket data and metadata using the InfluxDB CLI
2023/10/26 10:55:36 INFO: Restoring KV snapshot
2023/10/26 10:55:38 WARN: Restoring KV snapshot overwrote the operator token, ensure
 following commands use the correct token
2023/10/26 10:55:38 INFO: Restoring SQL snapshot
2023/10/26 10:55:39 INFO: Restoring TSM snapshot for shard 6952
2023/10/26 10:55:39 INFO: Restoring TSM snapshot for shard 6953
[More shard restores . . .]
2023/10/26 10:55:49 INFO: Restoring TSM snapshot for shard 8268
2023/10/26 10:55:49 INFO: Restoring TSM snapshot for shard 2
DEBUG: influx_migration.py: restore started at 2023-10-26 10:55:36 and took 13.51
 seconds to run.
INFO: influx_migration.py: Migration complete
```

Example Contoh 3: Penggunaan Migrasi PenuhCSV, Organisasi Tujuan, dan Bucket S3

Sama seperti contoh sebelumnya tetapi menggunakan Linux atau Mac dan menyimpan file di bucket S3,`my-s3-bucket`. Ini menghindari file cadangan yang membebani kapasitas penyimpanan lokal.

```
python3 influx_migration.py --full --src-host http://localhost:8086 --dest-host http://
dest-server-address:8086 --csv --dest-org MyOrg --s3-bucket my-s3-bucket
```

Outputnya akan serupa dengan yang berikut ini:

```
INFO: influx_migration.py: Creating directory influxdb-backups
INFO: influx_migration.py: Mounting influxdb-migration-bucket
INFO: influx_migration.py: Creating directory influxdb-backups/my-s3-bucket/influxdb-
backup-1698352128323
INFO: influx_migration.py: Backing up bucket data and metadata using the InfluxDB v2
 API
INFO: influx_migration.py: Restoring bucket data and metadata from csv
INFO: influx_migration.py: Restoring bucket some-bucket
INFO: influx_migration.py: Restoring bucket another-bucket
INFO: influx_migration.py: Restoring bucket primary-bucket
INFO: influx_migration.py: Migration complete
INFO: influx_migration.py: Unmounting influxdb-backups
INFO: influx_migration.py: Removing temporary mount directory
```

Mengonfigurasi instans DB

Bagian ini menunjukkan cara mengatur Amazon Timestream Anda untuk instans DB InfluxDB. Sebelum membuat instans DB, tentukan kelas instans DB yang akan menjalankan instans DB. Juga, tentukan di mana instans DB akan berjalan dengan memilih AWS Wilayah. Selanjutnya, buat instans DB.

Anda dapat mengonfigurasi instans DB dengan grup parameter DB. Grup parameter DB bertindak sebagai wadah untuk nilai konfigurasi engine yang diterapkan ke satu atau beberapa instance DB.

Parameter yang tersedia tergantung pada mesin DB dan versi mesin DB. Anda dapat menentukan grup parameter DB saat Anda membuat instance DB. Anda juga dapat mengubah instans DB untuk menentukannya.

> **Important**
>
> Saat ini, Anda tidak dapat mengubah konfigurasi komputasi (Jenis instans) dan Penyimpanan (Jenis Penyimpanan) dari instance yang ada.

Membuat instans DB

Menggunakan konsol

1. Masuk ke AWS Management Console dan buka Amazon Timestream untuk InfluxDB.
2. Di sudut kanan atas Amazon Timestream untuk konsol InfluxDB, pilih AWS Wilayah tempat Anda ingin membuat instans DB.
3. Di panel navigasi, pilih Database InfluxDB.
4. Pilih Buat database Influx.
5. Untuk DB Instance Identifier. masukkan nama yang akan mengidentifikasi instance Anda.
6. Berikan parameter konfigurasi dasar InfluxDB Nama Pengguna, Organisasi, Nama Bucket, dan Kata Sandi.

   > **Important**
   >
   > Nama pengguna, organisasi, nama bucket, dan kata sandi Anda akan disimpan sebagai AWS rahasia di Secrets Manager yang akan dibuat untuk akun Anda.

   Jika Anda perlu mengubah kata sandi pengguna setelah instans DB tersedia, Anda dapat memodifikasi menggunakan Influx CLI.
7.
8. Untuk Kelas Instance DB, pilih ukuran instans yang lebih sesuai dengan kebutuhan beban kerja Anda.
9. Untuk DB Storage Class, pilih kelas penyimpanan yang sesuai dengan kebutuhan Anda. Dalam semua kasus, Anda hanya perlu mengkonfigurasi penyimpanan yang dialokasikan.
10. Di bagian konfigurasi Konektivitas, pastikan instans InfluxDB Anda berada di subnet yang sama dengan klien baru Anda yang memerlukan konektivitas ke Timestream Anda untuk instans DB InfluxDB. Anda juga dapat memilih untuk membuat instans DB Anda tersedia untuk umum.
11. Pilih Buat database Influx.
12. Dalam daftar Database, pilih nama instans InfluxDB baru Anda untuk menampilkan detailnya. Instans DB memiliki status Creating hingga siap digunakan.
13. Saat statusnya berubah menjadi Tersedia, Anda dapat terhubung ke instans DB. Tergantung pada kelas instans DB dan jumlah penyimpanan, diperlukan waktu hingga 20 menit sebelum instans baru tersedia.

Menggunakan CLI

Untuk membuat instance DB dengan menggunakan AWS Command Line Interface, panggil `create-db-instance` perintah dengan parameter berikut:

```
--name
--vpc-subnet-ids
--vpc-security-group-ids
--db-instance-type
--db-storage-type
--username
--organization
--password
--allocated-storage
```

Untuk informasi tentang setiap pengaturan, lihat Pengaturan untuk instans DB.

Example Contoh: Menggunakan konfigurasi mesin default

Untuk Linux, macOS, atau Unix:

```
aws timestream-influxdb create-db-instance \
    --name myinfluxDbinstance \
    --allocated-storage 400 \
    --db-instance-type db.influx.4xlarge \
    --vpc-subnet-ids subnetid1 subnetid2
    --vpc-security-group-ids mysecuritygroup \
    --username masterawsuser \
    --password \
    --db-storage-type InfluxIOIncludedT2
```

Untuk Windows:

```
aws timestream-influxdb create-db-instance \
    --name myinfluxDbinstance \
    --allocated-storage 400 \
    --db-instance-type db.influx.4xlarge \
    --vpc-subnet-ids subnetid1 subnetid2
    --vpc-security-group-ids mysecuritygroup \
    --username masterawsuser \
    --password \
    --db-storage-type InfluxIOIncludedT2
```

Menggunakan API

Untuk membuat instance DB dengan menggunakan AWS Command Line Interface, panggil `CreateDBInstance` perintah dengan parameter berikut:

Untuk informasi tentang setiap pengaturan, lihat Pengaturan untuk instans DB.

> **Important**
>
> Bagian dari objek DBInstance respons yang Anda terima influxAuthParametersSecretArn. Ini akan menyimpan SecretsManager rahasia di akun Anda. ARN Ini hanya akan diisi setelah instans DB InfluxDB Anda tersedia. Rahasianya berisi parameter otentikasi masuknya yang disediakan selama proses. `CreateDbInstance` Ini adalah READONLY salinan karena rahasia ini tidak memengaruhi instance DB yang dibuat. updates/modifications/deletions Jika Anda menghapus rahasia ini, API tanggapan kami masih akan merujuk ke rahasia yang dihapusARN.

Setelah Anda selesai membuat Timestream untuk instans DB InfluxDB, kami sarankan Anda mengunduh, menginstal, dan mengkonfigurasi Influx. CLI

Masuknya CLI menyediakan cara sederhana untuk berinteraksi dengan InfluxDB dari baris perintah. Untuk petunjuk penginstalan dan penyiapan terperinci, lihat Menggunakan Influx CLI.

Pengaturan untuk instans DB

Anda dapat membuat instans DB menggunakan konsol, `create-db-instance` CLI perintah, atau `CreateDBInstance` Timestream untuk operasi API InfluxDB.

Tabel berikut memberikan rincian tentang pengaturan yang Anda pilih ketika Anda membuat instans DB.

Pengaturan Konsol	Deskripsi	CLIopsi dan parameter Timestream API
Penyimpanan yang dialokasikan	Jumlah penyimpanan yang dialokasikan untuk instans DB Anda (dalam gibibyte). Dalam beberapa kasus, mengalokasikan jumlah penyimpanan yang lebih	CLI: `allocated -storage`

Pengaturan Konsol	Deskripsi	CLIopsi dan parameter Timestream API
	tinggi untuk instans DB Anda daripada ukuran basis data Anda dapat meningkatkan performa I/O. Untuk informasi selengkapnya, lihat Penyimpanan instans InfluxDB.	API: `allocated storage`
Nama Bucket	Nama bucket untuk menginisialisasi instance InfluxDb	CLI: `bucket` API: `bucket`
Jenis instans DB	Konfigurasi untuk instans DB Anda. Misalnya, kelas instance DB db.influx.large memiliki memori 16 GiB, 2, memori yang dioptimalkan. vCPUs Jika memungkinkan, pilih jenis instans DB yang cukup besar sehingga set kerja kueri tipikal dapat disimpan di memori. Ketika set kerja disimpan di memori, sistem dapat menghindari menulis pada disk, yang akan meningkatkan performa. Untuk informasi selengkapnya, lihat Jenis kelas instans DB.	CLI: `db-instance-type` API: `Dbinstancetype`
Pengidentifikasi instans DB	Nama untuk instans DB Anda. Beri nama instans DB Anda dengan cara yang sama seperti cara Anda menamai server on-premise Anda. Pengidentifikasi instans DB Anda dapat berisi hingga 63 karakter alfanumerik, dan harus unik untuk akun Anda di Wilayah yang Anda pilih. AWS	CLI: `db-instance-identifier` API: `Dbinstanceidentifier`
Grup parameter DB	Grup parameter untuk instans DB Anda. Anda dapat memilih grup parameter default, atau Anda dapat membuat grup parameter kustom. Untuk informasi selengkapnya, lihat Menggunakan grup parameter DB.	CLI: `db-parameter-group-name` API: `DBParameterGroupName`

Pengaturan Konsol	Deskripsi	CLIopsi dan parameter Timestream API
Pengaturan Pengiriman Log	Nama bucket S3 adalah log InfluxDB akan disimpan.	CLI: `LogDeliveryConfiguration` API: `log-delivery-configuration`
Deployment Multi-AZ	Buat instans siaga untuk membuat replika sekunder pasif instans DB Anda di Zona Ketersediaan lainnya untuk dukungan failover. Kami merekomendasikan Multi-AZ untuk beban kerja produksi agar menjaga ketersediaan tetap tinggi. Untuk pengembangan dan pengujian, Anda dapat memilih Jangan buat instans siaga. Untuk informasi selengkapnya, lihat Mengkonfigurasi dan mengelola penyebaran Multi-AZ.	CLI: `MultiAz` API: `multi-az`
Kata sandi	Ini akan menjadi penggunaan kata sandi master Anda untuk Menginisialisasi instans InfluxDB Db Anda. Anda akan menggunakan kata sandi ini untuk masuk ke InfluxUI untuk mendapatkan token operator Anda.	CLI: `password` API: `password`

Pengaturan Konsol	Deskripsi	CLIopsi dan parameter Timestream API
Akses Publik	Ya untuk memberikan instans DB alamat IP publik, yang berarti dapat diakses di luarVPC. Agar dapat diakses publik, instans DB juga harus berada di subnet publik di. VPC Tidak untuk membuat instans DB hanya dapat diakses dari dalamVPC. Untuk terhubung ke instans DB dari luar instans DBVPC, instans DB harus dapat diakses publik. Selain itu, akses harus diberikan menggunakan aturan masuk dari grup keamanan instans DB. Selain itu, persyaratan lain harus dipenuhi.	CLI: `publicly-accessible` API: `PubliclyAccessible`
Jenis Penyimpanan	Jenis penyimpanan untuk instans DB Anda Anda dapat memilih di antara 3 jenis yang berbeda Influx yang disediakan IOPS termasuk penyimpanan sesuai dengan kebutuhan beban kerja Anda: * Masuknya IOPS Termasuk 3000 IOPS * Masuknya IOPS Termasuk 12000 IOPS * INflux IOPS Termasuk 16000 IOPS Untuk informasi selengkapnya, lihat Penyimpanan instans InfluxDB.	CLI: `db-storage-type` API: `DbStorageType`
Nama pengguna awal	Ini akan menjadi pengguna utama untuk menginisi alisasi instans DB InfluxDB Anda. Anda akan menggunakan nama pengguna ini untuk masuk ke InfluxUI untuk mendapatkan token operator Anda.	CLI: `username` API: `Username`

Pengaturan Konsol	Deskripsi	CLIopsi dan parameter Timestream API
Subnet	Subnet vpc untuk diasosiasikan dengan instance DB ini.	CLI: `vpc-subnet-ids` API: `VPCSubnetIds`
VPCGrup Keamanan (firewall)	Grup keamanan untuk dihubungkan dengan instans DB.	CLI: `vpc-security-group-ids` API: `VPCSecurityGroupIds`

Menghubungkan ke Amazon Timestream untuk instans DB InfluxDB

Sebelum dapat menghubungkan ke instans DB, Anda harus membuat instans DB. Untuk informasi, lihat Membuat instans DB. Setelah Amazon Timestream menyediakan instans DB Anda, gunakan InfluxDB, InfluxAPI, atau klien CLI atau utilitas apa pun yang kompatibel untuk InfluxDB untuk terhubung ke instans DB.

Topik

- Menemukan informasi koneksi untuk Amazon Timestream untuk instans DB InfluxDB
- Opsi autentikasi basis data
- Bekerja dengan Grup Parameter

Menemukan informasi koneksi untuk Amazon Timestream untuk instans DB InfluxDB

Informasi koneksi untuk instans DB mencakup titik akhir, port, nama pengguna, kata sandi, dan token akses yang valid, seperti operator atau semua token akses. Misalnya, untuk instans DB InfluxDB, anggaplah nilai endpoint adalah. `influxdb1-123456789.us-east-1.timestream-influxdb.amazonaws.com` Dalam hal ini, nilai port adalah 8086, dan pengguna database adalah admin. Dengan informasi ini, Anda menentukan nilai berikut dalam string koneksi:

- Untuk nama atau nama host atau DNS host, tentukan`influxdb1-123456789.us-east-1.timestream-influxdb.amazonaws.com`.
- Untuk port, tentukan 8086.
- Untuk pengguna, tentukan admin.
- Untuk kata sandi, tentukan yang Anda berikan saat membuat instans DB Anda.

> **Important**
>
> Saat Anda membuat Timestream untuk instans InfluxDB Db, bagian dari objek DBInstance respons yang Anda terima a. `influxAuthParametersSecretArn` Ini akan menyimpan arn ke SecretsManager rahasia di akun Anda. Ini hanya akan diisi setelah instans DB InfluxDB Anda tersedia. Rahasianya berisi parameter otentikasi masuknya yang disediakan selama proses. `CreateDbInstance` Ini adalah READONLY salinan karena rahasia ini tidak memengaruhi instance DB yang dibuat. updates/modifications/deletions Jika Anda menghapus rahasia ini, API tanggapan kami masih akan merujuk ke arn rahasia yang dihapus.

Titik akhir bersifat unik untuk setiap instans DB, dan nilai-nilai port serta pengguna dapat bervariasi. Untuk terhubung ke instans DB, Anda dapat menggunakan Influx, Influx CLIAPI, atau klien apa pun yang kompatibel dengan InfluxDB.

Untuk menemukan informasi koneksi untuk instans DB, gunakan AWS Management Console. Anda juga dapat menggunakan AWS perintah Command Line Interface (AWS CLI) `describe-db-instances` atau operasi API `GetDBInstance` TimeStream-InfluxDB.

Menggunakan AWS Management Console

1. Masuk ke AWS Management Console dan buka konsol Amazon Timestream.
2. Di panel navigasi, pilih Database InfluxDB untuk menampilkan daftar instans DB Anda.
3. Pilih nama instans DB untuk menampilkan detailnya.
4. Pada bagian Ringkasan, salin titik akhir. Selain itu, catat nomor port. Anda memerlukan titik akhir dan nomor port untuk terhubung ke instans DB.

Jika Anda perlu menemukan informasi nama pengguna dan kata sandi, pilih tab Detail Konfigurasi dan pilih `influxAuthParametersSecretArn` untuk mengakses Secrets Manager Anda.

Menggunakan CLI

- Untuk menemukan informasi koneksi untuk instans DB InfluxDB dengan menggunakan AWS CLI, panggil perintah. get-db-instance Dalam panggilan, kueri untuk ID instans DB, titik akhir, port, dan influxAuthParameters Rahasia.

  Untuk Linux, macOS, atau Unix:

  ```
  aws timestream-influxdb get-db-instance --identifier id \
   --query "[name,endpoint,influxAuthParametersSecretArn]"
  ```

  Untuk Windows:

  ```
  aws timestream-influxdb get-db-instance --identifier id \
   --query "[name,endpoint,influxAuthParametersSecretArn]"
  ```

  Output Anda akan terlihat seperti berikut ini. Untuk mengakses informasi nama pengguna, Anda perlu memeriksa`influxAuthParameterSecret`.

  ```
  [
      [
          "mydb",
          "mydb-123456789012.us-east-1.timestream-influxdb.amazonaws.com",
          8086,
      ]
  ]
  ```

Membuat Token Akses

Dengan informasi ini Anda akan dapat terhubung untuk instans Anda untuk mengambil atau membuat token akses Anda. Ada beberapa cara untuk mencapai ini:

Menggunakan CLI

1. Jika Anda belum melakukannya, unduh, instal, dan konfigurasikan masuknya CLI.
2. Saat mengonfigurasi konfigurasi Influx Anda gunakan `--username-password` untuk CLI mengautentikasi.

```
influx config create --config-name YOUR_CONFIG_NAME  --host-url "https://
yourinstance.timestream-influxdb.amazonaws.com:8086"  --org yourorg --username-
password admin --active
```

3. Gunakan perintah influx auth create untuk membuat ulang token operator Anda. Perhatikan bahwa proses ini akan membatalkan token operator lama.

```
influx auth create --org kronos --operator
```

4. Setelah Anda memiliki token operator, Anda dapat menggunakan perintah daftar autentikasi masuknya untuk melihat token semua token Anda. Anda dapat menggunakan perintah influx auth create untuk membuat token semua akses.

> **Important**
>
> Anda harus melakukan langkah ini untuk mendapatkan token operator Anda terlebih dahulu, untuk kemudian dapat membuat token baru menggunakan InfluxDB API atau. CLI

Menggunakan UI Influx

1. Jelajahi Timestream Anda untuk instans InfluxDB menggunakan titik akhir yang dibuat untuk masuk dan mengakses UI InfluxDB. Anda harus menggunakan nama pengguna dan kata sandi yang digunakan untuk membuat instans DB InfluxDB Anda. Anda dapat mengambil informasi ini dari `influxAuthParametersSecretArn` yang ditentukan dalam objek respon dari. `CreateDbInstance`

   Atau Anda dapat membuka InfluxUI dari Timestream untuk Konsol manajemen InfluxDB:

   a. Masuk ke AWS Management Console dan buka Amazon Timestream untuk konsol InfluxDB di. https://console.aws.amazon.com/timestream/

   b. Di sudut kanan atas Amazon Timestream untuk konsol InfluxDB, pilih AWS Wilayah tempat Anda membuat instans DB.

   c. Dalam daftar Database, pilih nama instans InfluxDB Anda untuk menampilkan detailnya. Di pojok kanan atas, pilih Open Influx UI.

2. Setelah masuk ke InfluxUI Anda, navigasikan ke Load Data dan kemudian APIToken menggunakan bilah navigasi kiri.

3. Pilih + GENERATE API TOKEN dan pilih All Access API Token.
4. Masukkan deskripsi untuk API token dan pilih SAVE.
5. Salin token yang dihasilkan dan simpan untuk disimpan dengan aman.

> **Important**
>
> Saat membuat token dari InfluxUI, token yang baru dibuat hanya akan ditampilkan sekali. Pastikan Anda menyalin ini karena jika tidak, Anda harus membuatnya kembali.

Menggunakan InfluxDB API

- Kirim permintaan ke API `/api/v2/authorizations` titik akhir InfluxDB menggunakan metode permintaan. POST

  Sertakan yang berikut ini dengan permintaan Anda:

  a. Header:

     i. Otorisasi: Token < INFLUX _ OPERATOR _ TOKEN >

     ii. Tipe Konten: aplikasi/json

  b. Request JSON body: body dengan properti berikut:

     i. status: “aktif”

     ii. deskripsi: deskripsi API token

     iii. OrgID: ID organisasi InfluxDB

     iv. izin: Array objek di mana setiap objek mewakili izin untuk jenis sumber daya InfluxDB atau sumber daya tertentu. Setiap izin berisi properti berikut:

        A. tindakan: “baca” atau “tulis”

        B. resource: JSON objek yang mewakili sumber daya InfluxDB untuk memberikan izin. Setiap sumber daya berisi setidaknya properti berikut: orgID: InfluxDB organization ID

        C. jenis: Jenis sumber daya. Untuk informasi tentang jenis sumber daya InfluxDB yang ada, gunakan the /api/v2/resources endpoint.

Contoh berikut menggunakan `curl` dan InfluxDB API untuk menghasilkan token semua akses:

```
export INFLUX_HOST=https://influxdb1-123456789.us-east-1.timestream-
influxdb.amazonaws.com
export INFLUX_ORG_ID=<YOUR_INFLUXDB_ORG_ID>
export INFLUX_TOKEN=<YOUR_INFLUXDB_OPERATOR_TOKEN>

curl --request POST \
"$INFLUX_HOST/api/v2/authorizations" \
  --header "Authorization: Token $INFLUX_TOKEN" \
  --header "Content-Type: text/plain; charset=utf-8" \
  --data '{
    "status": "active",
    "description": "All access token for get started tutorial",
    "orgID": "'"$INFLUX_ORG_ID"'",
    "permissions": [
      {"action": "read", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
 "authorizations"}},
      {"action": "write", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
 "authorizations"}},
      {"action": "read", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
 "buckets"}},
      {"action": "write", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
 "buckets"}},
      {"action": "read", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
 "dashboards"}},
      {"action": "write", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
 "dashboards"}},
      {"action": "read", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type": "orgs"}},
      {"action": "write", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type": "orgs"}},
      {"action": "read", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
 "sources"}},
      {"action": "write", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
 "sources"}},
      {"action": "read", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type": "tasks"}},
      {"action": "write", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
 "tasks"}},
      {"action": "read", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
 "telegrafs"}},
      {"action": "write", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
 "telegrafs"}},
      {"action": "read", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type": "users"}},
      {"action": "write", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
 "users"}},
```

```
    {"action": "read", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
"variables"}},
    {"action": "write", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
"variables"}},
    {"action": "read", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
"scrapers"}},
    {"action": "write", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
"scrapers"}},
    {"action": "read", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
"secrets"}},
    {"action": "write", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
"secrets"}},
    {"action": "read", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
"labels"}},
    {"action": "write", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
"labels"}},
    {"action": "read", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type": "views"}},
    {"action": "write", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
"views"}},
    {"action": "read", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
"documents"}},
    {"action": "write", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
"documents"}},
    {"action": "read", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
"notificationRules"}},
    {"action": "write", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
"notificationRules"}},
    {"action": "read", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
"notificationEndpoints"}},
    {"action": "write", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
"notificationEndpoints"}},
    {"action": "read", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
"checks"}},
    {"action": "write", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
"checks"}},
    {"action": "read", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type": "dbrp"}},
    {"action": "write", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type": "dbrp"}},
    {"action": "read", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
"notebooks"}},
    {"action": "write", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
"notebooks"}},
    {"action": "read", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
"annotations"}},
```

```
      {"action": "write", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
 "annotations"}},
      {"action": "read", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
 "remotes"}},
      {"action": "write", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
 "remotes"}},
      {"action": "read", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
 "replications"}},
      {"action": "write", "resource": {"orgID": "'"$INFLUX_ORG_ID"'", "type":
 "replications"}}
    ]
  }
'
```

Opsi autentikasi basis data

Amazon Timestream untuk InfluxDB mendukung cara-cara berikut untuk mengautentikasi pengguna database:

- Autentikasi kata sandi – Instans DB Anda melakukan semua administrasi akun pengguna. Anda membuat pengguna, menentukan kata sandi, dan mengelola token menggunakan InfluxUI, Influx, atau CLI influx. API
- Otentikasi Token — Instans DB Anda melakukan semua administrasi akun pengguna. Anda dapat membuat pengguna, menentukan kata sandi, dan mengelola token menggunakan token operator Anda menggunakan InfluxCLI, dan Influx. API

Koneksi terenkripsi

Anda dapat menggunakan Secure Socket Layer (SSL) atau Transport Layer Security (TLS) dari aplikasi Anda untuk mengenkripsi koneksi ke instans DB. Sertifikat yang diperlukan untuk TLS jabat tangan antara InfluxDB dan aplikasi yang dibuat dan dikelola oleh layanan Kronos. Ketika sertifikat diperbarui, instance secara otomatis diperbarui dengan versi terbaru tanpa memerlukan intervensi pengguna.

Bekerja dengan Grup Parameter

Parameter basis data menentukan konfigurasi basis data. Misalnya, parameter basis data dapat menentukan jumlah sumber daya, seperti memori, yang akan dialokasikan ke basis data.

Anda mengelola konfigurasi database Anda dengan mengaitkan instans DB Anda dengan grup parameter. Amazon Timestream untuk InfluxDB mendefinisikan grup parameter dengan pengaturan default. Anda dapat juga menentukan grup parameter Anda sendiri dengan pengaturan yang disesuaikan.

Ikhtisar grup parameter

Grup parameter DB bertindak sebagai kontainer untuk nilai konfigurasi mesin yang diterapkan pada satu atau beberapa instans DB.

Topik

- Grup parameter kustom dan default
- Membuat grup parameter DB
- Parameter instans DB statis dan dinamis
- Parameter dan nilai parameter yang didukung

Grup parameter kustom dan default

Instans DB menggunakan grup parameter DB. Bagian berikut menjelaskan konfigurasi dan pengelolaan grup parameter instans DB.

Membuat grup parameter DB

Anda dapat membuat grup parameter DB baru menggunakan AWS Management Console, AWS Command Line Interface, atau TimestreamAPI.

Batasan berikut berlaku untuk grup parameter DB:

- Nama harus berisi 1 sampai 255 huruf, angka, atau tanda hubung.
- Nama grup parameter default boleh menyertakan titik, seperti `default.InfluxDB.2.7`. Namun, nama grup parameter kustom tidak boleh menyertakan titik.
- Karakter pertama harus berupa huruf.
- Nama tidak dapat dimulai dengan “dbpg-”
- Nama tidak boleh diakhiri dengan tanda hubung atau berisi dua tanda hubung berturut-turut.
- Jika Anda membuat instans DB tanpa menentukan grup parameter DB, instans DB menggunakan default mesin InfluxDB.

Anda tidak dapat memodifikasi pengaturan parameter dari grup parameter default. Anda dapat melakukan hal berikut sebagai gantinya:

1. Membuat grup parameter baru.
2. Mengubah pengaturan parameter sesuai keinginan Anda. Tidak semua parameter mesin DB dalam grup parameter memenuhi syarat untuk dimodifikasi.
3. Perbarui instans DB Anda untuk menggunakan grup parameter kustom. Untuk informasi tentang memperbarui instans DB, lihatMemperbarui instans DB.

> **Note**
>
> Jika Anda telah memodifikasi instans DB Anda untuk menggunakan grup parameter kustom, dan Anda memulai instans DB, Amazon Timestream untuk InfluxDB secara otomatis me-reboot instans DB sebagai bagian dari proses startup.
> Saat ini, Anda tidak akan dapat memodifikasi grup parameter khusus setelah dibuat. Jika Anda perlu mengubah parameter, Anda harus membuat grup parameter kustom baru dan menetapkannya ke instance yang memerlukan perubahan konfigurasi ini. Jika Anda memperbarui instans DB yang ada untuk menetapkan grup parameter baru, itu akan selalu diterapkan segera dan reboot instance Anda.

Parameter instans DB statis dan dinamis

Parameter instans InfluxDB DB selalu statis. Mereka berperilaku sebagai berikut:

Saat Anda mengubah parameter statis, menyimpan grup parameter DB, dan menetapkannya ke sebuah instance, perubahan parameter akan berlaku secara otomatis setelah instance di-boot ulang.

Saat Anda mengaitkan grup parameter DB baru dengan instans DB, Timestream menerapkan parameter statis yang dimodifikasi hanya setelah instance DB di-boot ulang. Saat ini satu-satunya pilihan adalah segera berlaku.

Untuk informasi selengkapnya tentang cara mengubah grup parameter DB, lihat Memperbarui instans DB.

Parameter dan nilai parameter yang didukung

Untuk menentukan parameter yang didukung untuk instans DB Anda, lihat parameter dalam grup parameter DB yang digunakan oleh instans DB. Untuk informasi selengkapnya, lihat Melihat nilai parameter untuk grup parameter DB.

Untuk informasi selengkapnya tentang semua parameter yang didukung oleh InfluxDB versi open-source, lihat opsi konfigurasi InfluxDB. Saat ini Anda hanya dapat memodifikasi parameter InfluxDB berikut:

Parameter	Deskripsi	Nilai default	Nilai	Rentang yang valid	Catatan
flux-log-enabled	Sertakan opsi untuk menampilkan log rinci untuk kueri Flux	FALSE	benar, salah	N/A	
tingkat log	Tingkat keluaran log. InfluxDB mengeluarkan entri log dengan tingkat keparahan lebih besar dari atau sama dengan tingkat yang ditentukan.	info	debug, info, kesalahan	N/A	
tidak ada tugas	Jumlah kueri yang diizinkan untuk dieksekus	FALSE	benar, salah	N/A	

Parameter	Deskripsi	Nilai default	Nilai	Rentang yang valid	Catatan
	i secara bersamaan . Pengatura n ke 0 memungkin kan jumlah kueri bersamaan yang tidak terbatas.				
pertanyaan-konkurensi	Nonaktifkan penjadwal tugas. Jika tugas bermasala h mencegah InfluxDB dimulai, gunakan opsi ini untuk memulai InfluxDB tanpa menjadwal kan atau menjalankan tugas.	1024		N/A	

Parameter	Deskripsi	Nilai default	Nilai	Rentang yang valid	Catatan
query-queue-size	Jumlah maksimum kueri yang diizinkan dalam antrian eksekusi. Ketika batas antrian tercapai, kueri baru ditolak. Pengaturan ke 0 memungkinkan jumlah kueri yang tidak terbatas dalam antrian.	1024		N/A	
tipe penelusuran	Aktifkan penelusuran di InfluxDB dan tentukan jenis penelusuran. Penelusuran dinonaktifkan secara default.	""	log, jaeger	N/A	

Parameter	Deskripsi	Nilai default	Nilai	Rentang yang valid	Catatan
metrik-di nonaktifkan	Nonaktifk an titik akhir HTTP / metrik yang memperlih atkan metrik InfluxDB internal.	FALSE		N/A	
http-idle-timeout	Durasi maksimum server harus menjaga koneksi tetap hidup sambil menunggu permintaan baru. Setel ke `0` tanpa batas waktu.	3m0s	Durasi dengan unithours,minu nds . Contoh: `durationType=minutes,value=10`	Jam: -Minimal: 0 -Maksimum: 256205 Menit: -Minimal: 0 -Maksimum: 15372286 Detik: -Minimal: 0 -Maksimum: 922337203 Milidetik: -Minimal: 0 -Maksimum : 922337203 685	

Parameter	Deskripsi	Nilai default	Nilai	Rentang yang valid	Catatan
http-read-header-timeout	Durasi maksimum server harus mencoba membaca HTTP header untuk permintaan baru. Setel ke `0` tanpa waktu istirahat.	10s	Durasi dengan unithours,minutes . Contoh: `durationType=minutes,value=10`	Jam: -Minimal: 0 -Maksimum: 256205 Menit: -Minimal: 0 -Maksimum: 15372286 Detik: -Minimal: 0 -Maksimum: 922337203 Milidetik: -Minimal: 0 -Maksimum : 9223372036685	

Parameter	Deskripsi	Nilai default	Nilai	Rentang yang valid	Catatan
http-read-timeout	Durasi maksimum server harus mencoba membaca keseluruhan permintaan baru. Setel ke `0` tanpa waktu istirahat.	0	Durasi dengan unithours,minutes . Contoh: `durationType=minutes,value=10`	Jam: -Minimal: 0 -Maksimum: 256205 Menit: -Minimal: 0 -Maksimum: 15372286 Detik: -Minimal: 0 -Maksimum: 922337203 Milidetik: -Minimal: 0 -Maksimum: 922337203685	

Parameter	Deskripsi	Nilai default	Nilai	Rentang yang valid	Catatan
http-write-timeout	Durasi maksimum server harus menghabiskan pemrosesan dan menanggapi permintaan menulis. Setel ke `0` tanpa waktu istirahat.	0	Durasi dengan unithours,minutes . Contoh: `durationType=minutes,value=10`	Jam: -Minimal: 0 -Maksimum: 256205 Menit: -Minimal: 0 -Maksimum: 15372286 Detik: -Minimal: 0 -Maksimum: 922337203 Milidetik: -Minimal: 0 -Maksimum : 922337203685	

Parameter	Deskripsi	Nilai default	Nilai	Rentang yang valid	Catatan
influxql-max-select-buckets	Jumlah maksimum grup berdasarkan ember waktu yang dapat dibuat SELECT pernyataan. 0memungkinkan jumlah ember yang tidak terbatas.	0	Long	Minimal: 0 Maksimal: 9.223.372.036.854.775.807	

Parameter	Deskripsi	Nilai default	Nilai	Rentang yang valid	Catatan
influxql-max-select-point	Jumlah maksimum poin yang dapat diproses oleh SELECT pernyataan. 0memungkinkan jumlah poin yang tidak terbatas. InfluxDB memeriksa jumlah poin setiap detik (jadi kueri yang melebihi maksimum tidak segera dibatalkan).	0	Long	Minimal: 0 Maksimal: 9.223.372.036.854.775.807	
influxql-max-select-series	Jumlah maksimum seri SELECT pernyataan dapat dikembalikan. 0memungkinkan jumlah seri yang tidak terbatas.	0	Long	Minimal: 0 Maksimal: 9.223.372.036.854.775.807	

Parameter	Deskripsi	Nilai default	Nilai	Rentang yang valid	Catatan
pprof-dinonaktifkan	Nonaktifkan `/debug/pprof` HTTP titik akhir. Titik akhir ini menyediakan data profil runtime dan dapat membantu saat men-debug.	FALSE	Boolean	N/A	
query-initial-memory-bytes	Byte awal memori dialokasikan untuk kueri.	0	Long	Minimal: 0 Maksimum: query-memory-bytes	
query-max-memory-bytes	Total byte maksimum memori yang diizinkan untuk kueri.	0	Long	Minimal: 0 Maksimal: 9.223.372.036.854.775.807	
query-memory-bytes	Menentukan Time to Live (TTL) dalam hitungan menit untuk sesi pengguna yang baru dibuat.	0	Long	Minimal: 0 Maksimal: 2.147.483.647	Harus lebih besar dari atau sama dengan query-initial-memory-bytes.

Parameter	Deskripsi	Nilai default	Nilai	Rentang yang valid	Catatan
panjang sesi	Menentukan Time to Live (TTL) dalam hitungan menit untuk sesi pengguna yang baru dibuat.	60	Bilangan Bulat	Minimal: 0 Maksimal: 2880	

Parameter	Deskripsi	Nilai default	Nilai	Rentang yang valid	Catatan
session-renew-disabled	Menonaktifkan secara otomatis memperpanjang sesi pengguna TTL pada setiap permintaan. Secara default, setiap permintaan menetapkan waktu kedaluwarsa sesi menjadi lima menit dari sekarang. Ketika dinonaktifkan, sesi kedaluwarsa setelah panjang sesi yang ditentukan dan pengguna diarahkan ke halaman login, bahkan	FALSE	Boolean	N/A	

Parameter	Deskripsi	Nilai default	Nilai	Rentang yang valid	Catatan
	jika baru-baru ini aktif.				
storage-cache-max-memory-ukuran	Ukuran maksimum (dalam byte) cache shard dapat mencapai sebelum mulai menolak penulisan.	1073741824	Long	Minimal: 0 Maksimal: 549755813888	Harus lebih rendah dari kapasitas memori total instance. Kami merekomendasikan untuk mengaturnya di bawah 15% dari total kapasitas memori.
storage-cache-snapshot-memory-ukuran	Ukuran (dalam byte) di mana mesin penyimpanan akan memotret cache dan menuliskannya ke TSM file untuk membuat lebih banyak memori tersedia.	26214400	Long	Minimal: 0 Maksimal: 549755813888	Harus lebih rendah dari storage-cache-max-memory - ukuran.

Parameter	Deskripsi	Nilai default	Nilai	Rentang yang valid	Catatan
storage-cache-snapshot-write-durasi-dingin	Durasi di mana mesin penyimpanan akan mengambil snapshot cache dan menuliskannya ke TSM file baru jika pecahan belum menerima penulisan atau penghapusan.	10m0s	Durasi dengan unithours,minutes . Contoh: `durationType=minutes,value=10`	Jam: -Minimal: 0 -Maksimum: 256205 Menit: -Minimal: 0 -Maksimum: 15372286 Detik: -Minimal: 0 -Maksimum: 922337203 Milidetik: -Minimal: 0 -Maksimum : 9223372036685	

Parameter	Deskripsi	Nilai default	Nilai	Rentang yang valid	Catatan
storage-compact-full-write-durasi-dingin	Durasi di mana mesin penyimpanan akan memadatkan semua TSM file dalam pecahan jika belum menerima penulisan atau penghapusan.	4h0m0s	Durasi dengan unithours,minutes . Contoh: `durationType=minutes,value=10`	Jam: -Minimal: 0 -Maksimum: 256205 Menit: -Minimal: 0 -Maksimum: 15372286 Detik: -Minimal: 0 -Maksimum: 922337203 Milidetik: -Minimal: 0 -Maksimum: 9223372036854775807	
storage-compact-throughput-burst	Batas nilai (dalam byte per detik) bahwa TSM pemadatan dapat menulis ke disk.	50331648	Long	Minimal: 0 Maksimal: 9.223.372.036.854.775.807	

Parameter	Deskripsi	Nilai default	Nilai	Rentang yang valid	Catatan
storage-max-concurrent-compactions	Jumlah maksimum pemadatan penuh dan level yang dapat berjalan secara bersamaan. Nilai `0` hasil 50% dari yang `runtime.GOMAXPROCS(0)` digunakan saat runtime. Setiap angka yang lebih besar dari nol membatasi pemadatan ke nilai itu. Pengaturan ini tidak berlaku untuk snapshotting cache.	0	Bilangan Bulat	Minimal: 0 Maksimal: 64	

Parameter	Deskripsi	Nilai default	Nilai	Rentang yang valid	Catatan
storage-max-index-log-ukuran file	Ukuran (dalam byte) di mana file index write-ahead log (WAL) akan dipadatkan menjadi file indeks. Ukuran yang lebih rendah akan menyebabkan file log dipadatkan lebih cepat dan menghasilkan penggunaan heap yang lebih rendah dengan mengorbankan throughput tulis.	1048576	Long	Minimal: 0 Maksimal: 9.223.372.036.854.775.807	

Parameter	Deskripsi	Nilai default	Nilai	Rentang yang valid	Catatan
storage-no-validate-field-ukuran	Lewati validasi ukuran bidang pada permintaan tulis yang masuk.	FALSE	Boolean	N/A	
storage-retention-check-interval	Interval pemeriksaan penegakan kebijakan retensi.	30m0s	Durasi dengan unithours,minutes . Contoh: `durationType=minutes,value=10`	N/A	Jam: -Minimal: 0 -Maksimum: 256205 Menit: -Minimal: 0 -Maksimum: 15372286 Detik: -Minimal: 0 -Maksimum: 922337203 Milidetik: -Minimal: 0 -Maksimum : 922337203685

Parameter	Deskripsi	Nilai default	Nilai	Rentang yang valid	Catatan
storage-series-file-max-concurrent-snapshot-compactions	Jumlah maksimum pemadatan snapshot yang dapat berjalan secara bersamaan di semua partisi seri dalam database.	0	Bilangan Bulat	Minimal: 0 Maksimal: 64	

Parameter	Deskripsi	Nilai default	Nilai	Rentang yang valid	Catatan
storage-series-id-set-ukuran cache	Ukuran cache internal yang digunakan dalam TSI indeks untuk menyimpan hasil seri yang dihitung sebelumnya. Hasil cache dikembalikan dengan cepat daripada perlu dihitung ulang ketika kueri berikutnya dengan kunci tag/predikat nilai yang sama dijalankan. Menyetel nilai ini `0` akan menonaktifkan cache dan dapat menurunkan kinerja kueri.	100	Long	Minimal: 0 Maksimal: 9.223.372.036.854.775.807	

Parameter	Deskripsi	Nilai default	Nilai	Rentang yang valid	Catatan
storage-wal-max-concurrent-menulis	Jumlah maksimum menulis ke WAL direktori untuk mencoba pada saat yang sama.	0	Bilangan Bulat	Minimal: 0 Maksimal: 256	
storage-wal-max-write-penundaan	Jumlah maksimum waktu permintaan tulis ke WAL direktori akan menunggu ketika jumlah maksimum penulisan aktif bersamaan ke WAL direktori telah terpenuhi. Setel `0` untuk menonaktifkan batas waktu.	10m	Durasi dengan unithours,minutes . Contoh: `durationType=minutes,value=10`	Jam: -Minimal: 0 -Maksimum: 256205 Menit: -Minimal: 0 -Maksimum: 15372286 Detik: -Minimal: 0 -Maksimum: 922337203 Milidetik: -Minimal: 0 -Maksimum: 922337203685	

Parameter	Deskripsi	Nilai default	Nilai	Rentang yang valid	Catatan
ui-dinonaktifkan	Nonaktifkan antarmuka pengguna InfluxDB (UI). UI diaktifkan secara default.	FALSE	Boolean	N/A	

Pengaturan parameter yang tidak tepat dalam grup parameter dapat menimbulkan dampak buruk yang tidak diinginkan, termasuk penurunan performa dan ketidakstabilan sistem. Selalu berhati-hati saat memodifikasi parameter database. Coba pengaturan grup parameter berubah pada instance DB pengujian sebelum menerapkan perubahan grup parameter tersebut ke instance DB produksi.

Menggunakan grup parameter DB

Instans DB menggunakan grup parameter DB. Bagian berikut menjelaskan konfigurasi dan pengelolaan grup parameter instans DB.

Topik

- Membuat grup parameter DB
- Mengaitkan grup parameter DB dengan instans DB
- Menampilkan daftar grup parameter DB
- Melihat nilai parameter untuk grup parameter DB

Membuat grup parameter DB

Menggunakan AWS Management Console

1. Masuk ke AWS Management Console dan buka Amazon Timestream untuk konsol InfluxDB.
2. Di panel navigasi, pilih Grup parameter.
3. Pilih Buat grup parameter.
4. Di kotak Nama grup, masukkan nama grup parameter DB baru.
5. Di kotak Deskripsi, masukkan deskripsi untuk grup parameter DB baru.

6. Pilih parameter untuk memodifikasi dan menerapkan nilai yang diinginkan. Untuk informasi selengkapnya tentang parameter yang didukung, lihatParameter dan nilai parameter yang didukung.
7. Pilih Simpan.

Menggunakan AWS Command Line Interface

- Untuk membuat grup parameter DB dengan menggunakan AWS CLI, panggil `create-db-parameter-group` perintah dengan parameter berikut:

```
--db-parameter-group-name <value>
--description <value>
--endpoint_url <value>
--region <value>
--parameters (list) (string)
```

Example Contoh

Untuk informasi tentang setiap pengaturan, lihat Pengaturan untuk instans DB. Contoh ini menggunakan konfigurasi mesin default.

```
aws timestream-influxdb create-db-parameter-group
    --db-parameter-group-name YOUR_PARAM_GROUP_NAME\
    --endpoint-url YOUR_ENDPOINT
    --region YOUR_REGION \
    --parameters
 "InfluxDBv2={logLevel=debug,queryConcurrency=10,metricsDisabled=true}" \" \
    --debug
```

Mengaitkan grup parameter DB dengan instans DB

Anda dapat membuat grup parameter DB Anda sendiri dengan pengaturan yang disesuaikan. Anda dapat mengaitkan grup parameter DB dengan instans DB menggunakan AWS Management Console, the AWS Command Line Interface, atau API TimeStream-InfluxDB. Anda dapat melakukannya saat membuat atau memodifikasi instans DB.

Untuk informasi tentang cara membuat grup parameter DB, lihat Membuat grup parameter DB. Untuk informasi tentang membuat instans DB, lihat Membuat instans DB. Untuk informasi tentang memodifikasi instans DB, lihat.. Memperbarui instans DB

> **Note**
>
> Saat Anda mengaitkan grup parameter DB baru dengan instans DB, parameter statis yang dimodifikasi diterapkan hanya setelah instance DB di-boot ulang. Saat ini, hanya berlaku segera didukung. Timestream untuk InfluxDB hanya mendukung parameter statis.

Menggunakan AWS Management Console

1. Masuk ke AWS Management Console dan buka Amazon Timestream untuk konsol InfluxDB.
2. Di panel navigasi, pilih Database InfluxDB, lalu pilih instans DB yang ingin Anda modifikasi.
3. Pilih Perbarui. Halaman instans Update DB muncul.
4. Ubah pengaturan Grup parameter DB.
5. Pilih Lanjutkan dan periksa ringkasan modifikasi.
6. Saat ini hanya Terapkan segera didukung. Opsi ini dapat menyebabkan pemadaman dalam beberapa kasus karena akan me-reboot instans DB Anda.
7. Di halaman konfirmasi, tinjau perubahan Anda. Jika benar, pilih Perbarui instans DB untuk menyimpan perubahan Anda dan menerapkannya. Atau pilih Kembali untuk mengedit perubahan atau Batalkan untuk membatalkan perubahan Anda.

Menggunakan AWS Command Line Interface

Untuk Linux, macOS, atau Unix:

```
aws timestream-influxdb update-db-instance
--identifier YOUR_DB_INSTANCE_ID \
--region YOUR_REGION \
--db-parameter-group-identifier YOUR_PARAM_GROUP_ID \
--log-delivery-configuration "{\"s3Configuration\": {\"bucketName\":
 \"${LOGGING_BUCKET}\", \"enabled\": false }}"
```

Untuk Windows:

```
aws timestream-influxdb update-db-instance
--identifier YOUR_DB_INSTANCE_ID ^
--region YOUR_REGION ^
--db-parameter-group-identifier YOUR_PARAM_GROUP_ID ^
```

```
--log-delivery-configuration "{\"s3Configuration\": {\"bucketName\":
 \"${LOGGING_BUCKET}\", \"enabled\": false }}"
```

Menampilkan daftar grup parameter DB

Anda dapat mencantumkan grup parameter DB yang telah Anda buat untuk AWS akun Anda.

Menggunakan AWS Management Console

1. Masuk ke AWS Management Console dan buka Amazon Timestream untuk konsol InfluxDB.
2. Di panel navigasi, pilih Grup parameter.
3. Grup parameter DB muncul dalam daftar.

Menggunakan AWS Command Line Interface

Untuk membuat daftar semua grup parameter DB untuk AWS akun, gunakan AWS Command Line Interface `list-db-parameter-groups` perintah.

```
aws timestream-influxdb list-db-parameter-groups --region region
```

Untuk mengembalikan grup parameter DB tertentu untuk AWS akun, gunakan AWS Command Line Interface `get-db-parameter-group` perintah.

```
aws timestream-influxdb get-db-parameter-group --region region --identifier identifier
```

Melihat nilai parameter untuk grup parameter DB

Anda dapat memperoleh daftar semua parameter dalam grup parameter DB dan nilainya.

Menggunakan AWS Management Console

1. Masuk ke AWS Management Console dan buka Amazon Timestream untuk konsol InfluxDB.
2. Di panel navigasi, pilih Grup parameter.
3. Grup parameter DB muncul dalam daftar.
4. Pilih nama grup parameter untuk melihat daftar parameternya.

Menggunakan AWS Command Line Interface

Untuk melihat nilai parameter untuk grup parameter DB, gunakan AWS Command Line Interface `get-db-parameters` perintah dengan parameter yang diperlukan berikut.

```
--db-parameter-group-name
```

Menggunakan API

Untuk melihat nilai parameter untuk grup parameter DB, gunakan API `GetDBParameters` perintah Timestream dengan parameter yang diperlukan berikut.

```
DBParameterGroupName
```

Mengelola instans DB

Bagian ini mencakup berbagai aspek pengelolaan Timestream untuk instans InfluxDB untuk memastikan kinerja, ketersediaan, dan kemampuan pemantauan yang optimal. Ini memberikan panduan tentang memperbarui konfigurasi instance database Anda, menangani penerapan multi-AZ, dan proses failover. Ini juga menjelaskan cara menghapus instance database dan mengatur tampilan log untuk instans InfluxDB Anda.

Topik

Memperbarui instans DB

Anda dapat memperbarui parameter konfigurasi Timestream untuk instans InfluxDB berikut:

- Kelas instans
- Jenis deployment

- Grup parameter
- Konfigurasi pengiriman log

> **Important**
>
> Kami menyarankan Anda untuk menguji semua perubahan pada instance pengujian sebelum memodifikasi instans produksi untuk memahami dampaknya, terutama saat memutakhirkan versi database. Tinjau dampak pada database dan aplikasi Anda sebelum memperbarui pengaturan. Beberapa modifikasi memerlukan reboot instans DB, yang mengakibatkan downtime.

Menggunakan AWS Management Console

1. Masuk ke AWS Management Console dan buka Amazon Timestream untuk konsol InfluxDB.
2. Di panel navigasi, pilih Database InfluxDB, lalu pilih instans DB yang ingin Anda modifikasi.
3. Pilih Ubah.
4. Pada halaman Modify DB instance, buat perubahan yang diinginkan.
5. Pilih Lanjutkan dan periksa ringkasan modifikasi.
6. Pilih Berikutnya.
7. Tinjau perubahan Anda.
8. Pilih Modify instance untuk menerapkan perubahan Anda.

> **Note**
>
> Modifikasi ini memerlukan reboot instans Influx DB, dan dapat menyebabkan pemadaman dalam beberapa kasus.

Menggunakan AWS Command Line Interface

Untuk memperbarui instans DB dengan menggunakan AWS Command Line Interface, panggil `update-db-instance` perintah. Tentukan pengidentifikasi instans DB dan nilai untuk opsi yang ingin Anda modifikasi. Untuk informasi tentang setiap opsi, lihat Pengaturan untuk instans DB.

Example Contoh

Kode berikut memodifikasi mydbinstance dengan menetapkan dbparametergroup yang berbeda. Perubahan diterapkan segera.

Untuk Linux, macOS, atau Unix:

```
aws timestream-influxdb update-db-instance \
    --identifier mydbinstance \
    --db-instance-type desired-instance-type \
    --deployment-type desired-deployment-type \
    --db-parameter-group-name newparamgroup \
    --port 8086
```

Untuk Windows:

```
aws timestream-influxdb update-db-instance ^
    --identifier mydbinstance ^
    --db-instance-type desired-instance-type ^
    --deployment-type desired-deployment-type ^
    --db-parameter-group-name newparamgroup
    --port 8086
```

Memelihara instans DB

Secara berkala, Amazon TimeStream untuk InfluxDB melakukan pemeliharaan di Amazon Timestream untuk sumber daya InfluxDB. Pemeliharaan paling sering melibatkan pembaruan ke sumber daya berikut dalam instans DB Anda:

- Perangkat keras yang mendasarinya
- Sistem operasi yang mendasarinya (OS)
- Versi mesin basis data

Pembaruan pada sistem operasi paling sering terjadi untuk masalah keamanan.

Beberapa item pemeliharaan mengharuskan Amazon TimeStream untuk InfluxDB membuat instans DB Anda offline untuk waktu yang singkat. Item pemeliharaan yang mengharuskan sumber daya untuk offline mencakup patching sistem operasi atau basis data yang diperlukan. Patching yang diperlukan secara otomatis dijadwalkan hanya untuk patch yang terkait dengan keamanan dan

keandalan instans. Patching tersebut jarang terjadi, biasanya sekali setiap beberapa bulan. Ini jarang membutuhkan lebih dari periode pemeliharaan Anda.

- Jendela pemeliharaan dikonfigurasi untuk berlangsung setiap hari antara pukul 12 pagi dan 4 pagi waktu setempat untuk Wilayah instans Anda sedang di-host.
- Sumber daya pelanggan dapat ditambal seminggu sekali di salah satu dari tujuh jendela pemeliharaan dalam seminggu.

Menghapus instans DB

Menghapus instans DB berpengaruh pada pemulihan instans, dan ketersediaan snapshot. Pertimbangkan masalah berikut:

- Jika Anda ingin menghapus semua Timestream untuk sumber daya InfluxDB, perhatikan bahwa sumber daya instans DB dikenakan biaya penagihan.
- Saat status instans DB dihapus, nilai sertifikat CA-nya tidak muncul di konsol Timestream untuk InfluxDB atau dalam output untuk AWS Command Line Interface perintah atau operasi Timestream. API
- Waktu yang diperlukan untuk menghapus instans DB bervariasi tergantung pada seberapa banyak data yang dihapus, dan apakah snapshot akhir diambil.

Anda dapat menghapus instans DB menggunakan AWS Management Console, the AWS Command Line Interface, atau TimestreamAPI. Anda harus memberikan nama instans DB:

Menggunakan AWS Management Console

1. Masuk ke AWS Management Console dan buka Amazon Timestream untuk konsol InfluxDB.
2. Di panel navigasi, pilih Database InfluxDB, lalu pilih instans DB yang ingin Anda hapus.
3. Pilih Hapus.
4. Masukkan konfirmasi di dalam kotak.
5. Pilih Hapus.

Menggunakan AWS Command Line Interface

Untuk menemukan instance instance IDs DB di akun Anda, panggil `list-db-instances` perintah:

```
aws timestream-influxdb list-db-instances \
--endpoint-url YOUR_ENDPOINT \
--region YOUR_REGION
```

Untuk menghapus instans DB dengan menggunakan AWSCLI, panggil `delete-db-instance` perintah dengan opsi berikut:

```
aws timestream-influxdb list-db-instances \
--identifier YOUR_DB_INSTANCE \
```

Example Contoh

Untuk Linux, macOS, atau Unix:

```
aws timestream-influxdb delete-db-instance \
    --identifier mydbinstance
```

Untuk Windows:

```
aws timestream-influxdb delete-db-instance ^
    --identifier mydbinstance
```

Deployment instans DB Multi-AZ

Amazon TimeStream untuk InfluxDB menyediakan ketersediaan tinggi dan dukungan failover untuk instans DB menggunakan penerapan multi-AZ dengan satu instans DB siaga. Jenis deployment ini disebut deployment instans DB Multi-AZ. Amazon Timestream untuk InfluxDB menggunakan teknologi failover Amazon.

Dalam penerapan instans DB multi-AZ, Amazon Timestream secara otomatis menyediakan dan memelihara replika siaga sinkron di Availability Zone yang berbeda. Instans DB primer direplikasi secara sinkron ke seluruh Zona Ketersediaan ke replika siaga untuk memberikan redundansi data. Menjalankan instans DB dengan ketersediaan tinggi dapat meningkatkan ketersediaan selama kegagalan instans DB dan gangguan Availability Zone. Untuk informasi selengkapnya tentang Zona Ketersediaan, lihat AWS Wilayah dan Zona Ketersediaan.

> **Note**
>
> Opsi ketersediaan tinggi bukanlah solusi penskalaan untuk skenario hanya baca. Anda tidak dapat menggunakan replika siaga untuk menyajikan lalu lintas baca.

Menggunakan konsol Amazon Timestream, Anda dapat membuat penerapan instans DB multi-AZ hanya dengan menentukan opsi Buat instans siaga di bagian konfigurasi Ketersediaan dan daya tahan saat membuat instans DB. Anda juga dapat menentukan penerapan instans DB multi-AZ dengan Amazon Timestream AWS Command Line Interface atau Amazon. API Gunakan CLI perintah `create-db-instance` atau, atau `CreateDBInstance` API operasi.

Instans DB yang menggunakan deployment instans DB Multi-AZ dapat meningkatkan latensi tulis dan commit dibandingkan dengan deployment AZ Tunggal. Hal ini karena replikasi data sinkron yang terjadi. Anda mungkin mengalami perubahan latensi jika penerapan Anda gagal ke replika siaga, meskipun direkayasa dengan konektivitas jaringan latensi rendah AWS antara Availability Zones. Untuk beban kerja produksi, kami menyarankan Anda menggunakan penyimpanan 12K atau 16K yang IOPS disertakan IOPS untuk kinerja yang cepat dan konsisten. Untuk informasi selengkapnya tentang kelas instans DB, lihat Kelas instans DB.

Mengkonfigurasi dan mengelola penyebaran Multi-AZ

Timestream untuk penyebaran InfluxDB multi-AZ hanya dapat memiliki satu siaga. Ketika deployment memiliki satu instans DB siaga, itu disebut deployment instans DB Multi-AZ. Deployment instans DB Multi-AZ memiliki satu instans DB siaga yang menyediakan dukungan failover, tetapi tidak melayani lalu lintas baca.

> **Important**
>
> Instans Anda harus memiliki setidaknya dua subnet yang terkait dengannya untuk menjalankan pembaruan Single-AZ ke Multi-AZ. Setelah instans dibuat, Anda tidak dapat memodifikasi mode penerapan dari Single-AZ ke Multi-AZ.

Anda dapat menggunakan AWS Management Console untuk menentukan apakah instans DB Anda adalah penerapan Single-AZ atau Multi-AZ.

Menggunakan AWS Management Console

1. Masuk ke AWS Management Console dan buka Amazon Timestream untuk konsol InfluxDB.
2. Di panel navigasi, pilih Database InfluxDB, lalu pilih DB identifier.

Deployment instans DB Multi-AZ memiliki karakteristik sebagai berikut:

- Hanya ada satu baris untuk instans DB.
- Nilai Peran adalah Instans atau Primer.
- Nilai Multi-AZ adalah Ya.

Proses failover untuk Amazon Timestream

Jika pemadaman instans DB yang direncanakan atau tidak direncanakan disebabkan oleh cacat infrastruktur, Amazon TimeStream untuk InfluxDB secara otomatis beralih ke replika siaga di zona ketersediaan lain jika Anda telah mengaktifkan Multi-AZ. Durasi penyelesaian failover bergantung pada aktivitas basis data dan kondisi lain pada saat instans DB primer tidak tersedia. Durasi failover biasanya antara 60–120 detik. Namun, transaksi besar atau proses pemulihan yang panjang dapat meningkatkan waktu failover. Ketika failover selesai, dibutuhkan waktu tambahan untuk konsol Timestream untuk mencerminkan zona ketersediaan baru.

> **Note**
>
> Amazon Timestream menangani failover secara otomatis sehingga Anda dapat melanjutkan operasi database secepat mungkin tanpa intervensi administratif. Instans DB primer otomatis beralih ke replika siaga jika salah satu dari kondisi yang dijelaskan dalam tabel berikut terjadi.

Alasan failover	Deskripsi
Sistem operasi yang mendasari instance database Timestream sedang ditambal dalam operasi offline.	Failover dipicu selama periode pemeliharaan untuk patch OS atau pembaruan keamanan.

Alasan failover	Deskripsi
Host utama instans Timestream Multi-AZ tidak sehat.	Deployment instans DB Multi-AZ mendeteksi instans DB primer mengalami gangguan dan failover.
Host utama instans Timestream Multi-AZ tidak dapat dijangkau karena hilangnya konektivitas jaringan.	Pemantauan Timestream mendeteksi kegagalan jangkauan jaringan ke instans DB utama dan memicu failover.
Instans Timestream dimodifikasi oleh pelanggan.	Modifikasi instans Timesteam untuk InfluxDB DB memicu failover. Untuk informasi selengkap nya, lihat Memperbarui instans DB.
Instans utama Timestream Multi-AZ sibuk dan tidak responsif.	Instans DB primer tidak responsif. Kami menyarankan Anda melakukan hal berikut: * Periksa acara untuk penggunaan berlebiha nCPU, memori, atau ruang swap. * Evaluasi beban kerja Anda untuk menentukan apakah Anda menggunakan kelas instans DB yang sesuai. Untuk informasi selengkapnya, lihat Kelas instans DB.
Volume penyimpanan yang mendasari host utama instans Timestream Multi-AZ mengalami kegagalan.	Deployment instans DB Multi-AZ mendeteksi masalah penyimpanan pada instans DB primer dan mengalami failover.

Mengatur JVM TTL pencarian DNS nama

Mekanisme failover secara otomatis mengubah catatan Domain Name System (DNS) dari instans DB untuk menunjuk ke instans DB siaga. Alhasil, Anda perlu membentuk kembali koneksi yang ada dengan instans basis data Anda. Dalam lingkungan mesin virtual Java (JVM), karena cara kerja mekanisme DNS caching Java, Anda mungkin perlu mengkonfigurasi ulang pengaturanJVM.

Pencarian DNS nama JVM cache. Ketika JVM menyelesaikan nama host ke alamat IP, itu cache alamat IP untuk jangka waktu tertentu, yang dikenal sebagai (). time-to-liveTTL

Karena AWS sumber daya menggunakan entri DNS nama yang terkadang berubah, kami sarankan Anda mengonfigurasi JVM dengan TTL nilai tidak lebih dari 60 detik. Melakukan hal ini memastikan bahwa ketika alamat IP sumber daya berubah, aplikasi Anda dapat menerima dan menggunakan alamat IP baru sumber daya dengan meminta. DNS

Pada beberapa konfigurasi Java, JVM default TTL diatur sehingga tidak pernah menyegarkan DNS entri sampai restart. JVM Jadi, jika alamat IP untuk AWS sumber daya berubah saat aplikasi Anda masih berjalan, itu tidak dapat menggunakan sumber daya itu sampai Anda secara manual me-restart JVM dan informasi IP cache di-refresh. Dalam hal ini, sangat penting untuk mengatur JVM's TTL sehingga secara berkala menyegarkan informasi IP cache.

Anda bisa mendapatkan JVM TTL default dengan mengambil nilai `networkaddress.cache.ttl` properti:

```
String ttl = java.security.Security.getProperty("networkaddress.cache.ttl");
```

> **Note**
>
> Default TTL dapat bervariasi sesuai dengan versi Anda JVM dan apakah manajer keamanan diinstal. Banyak yang JVMs memberikan default TTL kurang dari 60 detik. Jika Anda menggunakan seperti itu JVM dan tidak menggunakan manajer keamanan, Anda dapat mengabaikan sisa topik ini.
> Untuk memodifikasi JVM'sTTL, atur nilai properti networkaddress.cache.ttl. Gunakan salah satu metode berikut, tergantung pada kebutuhan Anda:
>
> - Untuk menetapkan nilai properti secara global untuk semua aplikasi yang menggunakanJVM, atur `networkaddress.cache.ttl` dalam `$JAVA_HOME/jre/lib/security/java.security` file.
>
>   ```
>   networkaddress.cache.ttl=60
>   ```
>
> - Untuk menetapkan properti secara lokal hanya untuk aplikasi Anda, tetapkan `networkaddress.cache.ttl` dalam kode inisialisasi aplikasi Anda sebelum koneksi jaringan dibuat.
>
>   ```
>   java.security.Security.setProperty("networkaddress.cache.ttl" , "60");
>   ```

Pengaturan untuk melihat log InfluxDB pada Instans Timestream Influxdb

Secara default InfluxDB menghasilkan log yang masuk ke stdout. Untuk informasi selengkapnya, lihat Mengelola log InfluxDB

Untuk melihat log InfluxDB yang dihasilkan dari Instance yang telah Anda buat melalui Timestream InfluxDB, kami memberikan kesempatan untuk menyediakan log per jam. Log ini akan masuk ke bucket S3 tertentu yang harus Anda buat sebelum membuat instance Anda.

- Sebelum membuat instance, bucket Amazon S3 yang disediakan juga harus memberikan izin TimeStream-InfluxDB untuk mengirim log ke bucket ini dengan menyediakan kebijakan bucket dengan Timestream InfluxDB Service Principal sebagai berikut (ganti *{BUCKET_NAME}* dengan nama sebenarnya dari bucket Amazon S3 Anda:

```
{
    "Version": "2012-10-17",
    "Id": "PolicyForInfluxLogs",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "timestream-influxdb.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::{BUCKET_NAME}/InfluxLogs/*"
        }
    ]
}
```

- Bucket yang disediakan harus berada di akun yang sama dan Region yang sama dari instans Timestream InfluxDB yang Anda buat

  Berikut adalah perintah yang dapat Anda panggil untuk membuat instance untuk menerima log masuknya:

```
aws timestream-influxdb create-db-instance \
    --name myinfluxDbinstance \
    --allocated-storage 400 \
    --db-instance-type db.influx.4xlarge \
    --vpc-subnet-ids subnetid1 subnetid2
    --vpc-security-group-ids mysecuritygroup \
    --username masterawsuser \
```

```
    --password \
    --db-storage-type InfluxIOIncludedT2
```

Berikut adalah format parameter ini.

```
-- log-delivery-configuration
{
    "S3Configuration": {
      "BucketName": "string",
      "Enabled": true|false
    }
}
```

- Bidang ini tidak diperlukan dan logging tidak diaktifkan secara default.
- Tidak menyetel bidang ini sama dengan tidak mengaktifkan log.
- Log akan dikirim ke bucket yang ditentukan dengan awalan. `InfluxLogs/`
- Setelah membuat instance, Anda dapat memodifikasi konfigurasi pengiriman log dengan `update-db-instance` API perintah.

InfluxDB menawarkan berbagai jenis log. Ini dapat dikonfigurasi dengan mengatur Parameter InfluxDB. Gunakan parameter flux-log-enabled dan log-level untuk mengonfigurasi jenis log yang dipancarkan dari instance. Untuk informasi selengkapnya, lihat Parameter dan nilai parameter yang didukung.

Menambahkan tanda dan label untuk sumber daya

Anda dapat memberi label Amazon Timestream untuk sumber daya InfluxDB menggunakan tag. Tag memungkinkan Anda mengkategorikan sumber daya dengan cara yang berbeda—misalnya, berdasarkan tujuan, pemilik, lingkungan, atau kriteria lainnya. Tanda membantu Anda melakukan hal berikut:

- Identifikasi sumber daya dengan cepat berdasarkan tanda yang Anda tetapkan padanya.
- Lihat AWS tagihan yang dipecah berdasarkan tag.

Penandaan didukung oleh AWS layanan seperti Amazon Elastic Compute Cloud (AmazonEC2), Amazon Simple Storage Service (Amazon S3), Timestream untuk InfluxDB, dan banyak lagi.

Penandaan yang efisien dapat memberikan wawasan biaya dengan memungkinkan Anda membuat laporan di seluruh layanan yang membawa tanda tertentu.

Terakhir, merupakan praktik yang baik untuk mengikuti strategi penandaan yang optimal. Untuk informasi, lihat Strategi Pemberian Tag AWS.

Pembatasan penandaan

Setiap tanda terdiri dari kunci dan nilai, yang keduanya Anda tentukan. Pembatasan berikut berlaku:

- Setiap Timestream untuk instans DB InfluxDB hanya dapat memiliki satu tag dengan kunci yang sama. Jika Anda mencoba menambahkan tag yang ada, nilai tag yang ada diperbarui ke nilai baru.
- Nilai bertindak sebagai deskriptor dalam kategori tag. Di Timestream untuk InfluxDB nilainya tidak bisa kosong atau null.
- Kunci dan nilai tanda peka huruf besar-kecil.
- Panjang kunci maksimum adalah 128 karakter Unicode.
- Panjang nilai maksimum adalah 256 karakter Unicode.
- Karakter yang diperbolehkan adalah huruf, spasi kosong, dan angka, ditambah karakter khusus berikut: `+ - = . _ : /`
- Jumlah maksimum tanda per sumber daya adalah 50.
- AWS-Assigned nama tag dan nilai secara otomatis diberi `aws:` awalan, yang tidak dapat Anda tetapkan. AWS-nama-nama tag yang ditetapkan tidak dihitung terhadap batas tag 50. Nama tanda yang ditetapkan pengguna memiliki prefiks `user:` dalam laporan alokasi biaya.
- Anda tidak dapat mengubah penerapan tanda ke versi sebelumnya.

Praktik terbaik keamanan untuk Timestream untuk InfluxDB

Optimalkan menulis ke InfluxDB

Seperti database deret waktu lainnya, InfluxDB dibangun untuk dapat menelan dan memproses data secara real-time. Agar sistem tetap berkinerja terbaik, kami sarankan pengoptimalan berikut saat menulis data ke InfluxDB:

- Batch Writes: Saat menulis data ke InfluxDB, tulis data dalam batch untuk meminimalkan overhead jaringan yang terkait dengan setiap permintaan penulisan. Ukuran batch optimal adalah 5000 baris

protokol baris per permintaan tulis. Untuk menulis beberapa baris dalam satu permintaan, setiap baris protokol baris harus dibatasi oleh baris baru (\n).

- Urutkan tag berdasarkan kunci: Sebelum menulis titik data ke InfluxDB, urutkan tag berdasarkan kunci dalam urutan leksikografis.

```
measurement,tagC=therefore,tagE=am,tagA=i,tagD=i,tagB=think fieldKey=fieldValue
 1562020262

# Optimized line protocol example with tags sorted by key
measurement,tagA=i,tagB=think,tagC=therefore,tagD=i,tagE=am fieldKey=fieldValue
 1562020262
```

- Gunakan presisi waktu yang paling kasar: — InfluxDB menulis data dalam presisi nanodetik, namun jika data Anda tidak dikumpulkan dalam nanodetik, tidak perlu menulis pada presisi itu. Untuk kinerja yang lebih baik, gunakan presisi paling kasar yang mungkin untuk stempel waktu. Anda dapat menentukan presisi penulisan saat:
  - Saat menggunakan SDK Anda dapat menentukan WritePrecision saat mengatur atribut waktu poin Anda. Untuk informasi selengkapnya tentang pustaka klien InfluxDB, lihat Dokumentasi InfluxDB.
  - Saat menggunakan Telegraf, Anda mengonfigurasi presisi waktu dalam konfigurasi agen Telegraf. Presisi ditentukan sebagai interval dengan unit integer + (misalnya 0s, 10ms, 2us, 4s). Satuan waktu yang valid adalah “ns”, “us”, “ms”, dan “s”.

```
[agent]
 interval ="10s"
 metric_batch_size="5000"
 precision = "0s"
```

- Gunakan kompresi gzip: — Gunakan kompresi gzip untuk mempercepat penulisan ke InfluxDB dan mengurangi bandwidth jaringan. Benchmark telah menunjukkan peningkatan kecepatan hingga 5x saat data dikompresi.
  - Saat menggunakan Telegraf, dalam konfigurasi plugin keluaran InfluxDB_v2 di telegraf.conf Anda, atur opsi content_encoding ke gzip:

```
[[outputs.influxdb_v2]]
  urls = ["http://localhost:8086"]
  # ...
  content_encoding = "gzip"
```

- Saat menggunakan pustaka klien, setiap pustaka klien InfluxDB menyediakan opsi untuk mengompresi permintaan tulis atau memberlakukan kompresi secara default. Metode untuk mengaktifkan kompresi berbeda untuk setiap pustaka. Untuk petunjuk spesifik, lihat Dokumentasi InfluxDB
- Saat menggunakan API `/api/v2/write` endpoint InfluxDB untuk menulis data, kompres data dengan gzip dan atur header Content-Encoding ke gzip.

Desain untuk kinerja

Rancang skema Anda untuk kueri kinerja yang lebih sederhana dan lebih banyak. Panduan berikut akan memastikan bahwa skema Anda akan mudah untuk query dan memaksimalkan kinerja kueri:

- Desain untuk kueri: Pilih pengukuran, kunci tag, dan kunci bidang yang mudah untuk query. Untuk mencapai tujuan ini, ikuti prinsip-prinsip ini:
  - Gunakan pengukuran yang memiliki nama sederhana dan jelaskan skema secara akurat.
  - Hindari menggunakan nama yang sama untuk kunci tag dan kunci bidang dalam skema yang sama.
  - Hindari menggunakan kata kunci Flux yang dicadangkan dan karakter khusus dalam kunci tag dan bidang.
  - Tag menyimpan metadata yang menggambarkan bidang dan umum di banyak titik data.
  - Bidang menyimpan data unik atau sangat bervariasi, biasanya titik data numerik.
  - Pengukuran dan kunci tidak boleh berisi data, tetapi digunakan untuk mengumpulkan atau mendeskripsikan data. Data akan disimpan dalam nilai tag dan bidang.
- Jaga agar kardinalitas deret waktu Anda tetap terkendali Kardinalitas seri tinggi adalah salah satu penyebab utama penurunan kinerja tulis dan baca di InfluxDB. Dalam konteks InfluxDB kardinalitas tinggi mengacu pada kehadiran sejumlah besar nilai tag unik. Nilai tag diindeks di InfluxDB yang berarti bahwa jumlah nilai unik yang sangat tinggi akan menghasilkan indeks yang lebih besar yang dapat memperlambat konsumsi data dan kinerja kueri.

  Untuk lebih memahami dan menyelesaikan potensi masalah terkait kardinalitas tinggi, Anda dapat mengikuti langkah-langkah berikut:
  - Memahami penyebab kardinalitas tinggi
  - Ukur kardinalitas ember Anda
  - Ambil tindakan untuk menyelesaikan kardinalitas tinggi

- Penyebab kardinalitas seri tinggi InfluxDB mengindeks data berdasarkan pengukuran dan tag untuk mempercepat pembacaan data. Setiap set elemen data yang diindeks membentuk kunci seri. Tag yang berisi informasi yang sangat bervariasi seperti string unikIDs, hash, dan acak mengarah ke sejumlah besar seri, juga dikenal sebagai kardinalitas seri tinggi. Kardinalitas seri tinggi adalah pendorong utama penggunaan memori tinggi di InfluxDB.
- Mengukur kardinalitas seri Jika Anda mengalami perlambatan kinerja atau melihat penggunaan memori yang terus meningkat di Timestream untuk instans InfluxDB, kami sarankan untuk mengukur kardinalitas seri bucket Anda.

  InfluxDB menyediakan fungsi yang memungkinkan Anda mengukur kardinalitas seri baik di Flux maupun InfluxQL.

  - Dalam Flux gunakan fungsi `influxdb.cardinality()`
  - Di FluxQL gunakan perintah `SHOW SERIES CARDINALITY`

  Dalam kedua kasus, mesin akan mengembalikan jumlah kunci seri unik dalam data Anda. Perlu diingat bahwa tidak disarankan untuk memiliki lebih dari 10 juta kunci seri pada Timestream Anda untuk instans InfluxDB.
- Penyebab kardinalitas seri tinggi Jika Anda menemukan bahwa salah satu ember Anda memiliki kardinalitas tinggi, ada beberapa langkah koreksi yang dapat Anda ambil untuk memperbaikinya:
  - Tinjau tag Anda: Pastikan beban kerja Anda tidak menghasilkan kasus jika tag memiliki nilai unik untuk sebagian besar entri. Ini bisa terjadi dalam kasus di mana jumlah nilai tag unik selalu bertambah dari waktu ke waktu, atau jika pesan jenis log ditulis ke database di mana setiap pesan akan memiliki kombinasi unik stempel waktu, tag, dll. Anda dapat menggunakan kode Flux berikut untuk membantu Anda mengetahui Tag mana yang paling berkontribusi terhadap masalah kardinalitas tinggi Anda:

```
// Count unique values for each tag in a bucketimport "influxdata/influxdb/schema"

cardinalityByTag = (bucket) => schema.tagKeys(bucket: bucket)
    |> map(
        fn: (r) => ({
            tag: r._value,
            _value: if contains(set: ["_stop", "_start"], value: r._value) then
                0
            else
                (schema.tagValues(bucket: bucket, tag: r._value)
                    |> count()
                    |> findRecord(fn: (key) => true, idx: 0))._value,
```

```
        }),
    )
    |> group(columns: ["tag"])
    |> sum()

cardinalityByTag(bucket: "example-bucket")
```

Jika Anda mengalami kardinalitas yang sangat tinggi, kueri di atas mungkin habis. Jika Anda mengalami batas waktu, jalankan kueri di bawah ini — satu per satu.

Menghasilkan daftar tag:

```
// Generate a list of tagsimport "influxdata/influxdb/schema"

schema.tagKeys(bucket: "example-bucket")
```

Hitung nilai tag unik untuk setiap tag:

```
// Run the following for each tag to count the number of unique tag valuesimport
 "influxdata/influxdb/schema"

tag = "example-tag-key"

schema.tagValues(bucket: "my-bucket", tag: tag)
    |> count()
```

Kami menyarankan Anda menjalankan ini pada titik waktu yang berbeda untuk mengidentifikasi tag mana yang tumbuh lebih cepat.

- Tingkatkan skema Anda: Ikuti rekomendasi pemodelan yang dibahas di kamiPraktik terbaik keamanan untuk Timestream untuk InfluxDB.
- Hapus atau agregat data lama untuk mengurangi kardinalitas: Pertimbangkan apakah kasus penggunaan Anda memerlukan semua data yang menyebabkan masalah kardinalitas tinggi Anda atau tidak. Jika data ini tidak lagi diperlukan atau sering diakses, Anda dapat menggabungkannya, menghapusnya, atau mengekspornya ke mesin lain seperti Timestream untuk Live Analytics untuk penyimpanan dan analisis jangka panjang.

Pemecahan Masalah

Peringatan versi “dev” tidak dikenali

Peringatan 'WARN: Tidak dapat mengurai versi “dev” yang dilaporkan oleh server, dengan asumsi pencadangan/pemulihan terbaru APIs didukung' dapat ditampilkan selama migrasi. Peringatan ini bisa diabaikan.

Migrasi gagal selama tahap restorasi

Jika terjadi migrasi yang gagal selama tahap restorasi, pengguna dapat menggunakan `--retry-restore-dir` bendera untuk mencoba kembali restorasi. Gunakan `--retry-restore-dir` bendera dengan jalur ke direktori yang sebelumnya dicadangkan untuk melewati tahap pencadangan dan coba lagi tahap pemulihan. Direktori cadangan yang dibuat yang digunakan untuk migrasi akan ditunjukkan jika migrasi gagal selama pemulihan.

Kemungkinan alasan kegagalan pemulihan meliputi:

- Token tujuan InfluxDB tidak valid - Bucket yang ada di instance tujuan dengan nama yang sama seperti pada instance sumber. Untuk migrasi bucket individual, gunakan `--dest-bucket` opsi untuk menetapkan nama unik untuk bucket yang dimigrasi
- Kegagalan konektivitas, baik dengan host sumber atau tujuan atau dengan bucket S3 opsional.

Amazon Timestream untuk pedoman operasional dasar InfluxDB

Berikut ini adalah pedoman operasional dasar yang harus diikuti setiap orang saat bekerja dengan Amazon TimeStream untuk InfluxDB. Perhatikan bahwa Amazon Timestream untuk Perjanjian Tingkat Layanan InfluxDB mengharuskan Anda mengikuti panduan ini:

- Gunakan metrik untuk memantau memori Anda,CPU, dan penggunaan penyimpanan. Anda dapat mengatur Amazon CloudWatch untuk memberi tahu Anda saat pola penggunaan berubah atau saat Anda mendekati kapasitas penerapan. Dengan begitu, Anda dapat mempertahankan performa sistem dan ketersediaan.
- Tingkatkan skala instans DB Anda saat mendekati batas kapasitas penyimpanan. Anda akan memiliki buffer dalam penyimpanan dan memori untuk mengakomodasi peningkatan permintaan yang tidak terduga dari aplikasi Anda. Ingatlah bahwa saat ini, Anda perlu membuat instance baru dan memigrasikan data Anda untuk mencapai hal ini.

- Jika beban kerja basis data Anda memerlukan lebih banyak I/O daripada yang Anda sediakan, pemulihan setelah failover atau kegagalan basis data akan lambat. Untuk meningkatkan kapasitas I/O instans DB, lakukan salah satu atau semua hal berikut:
  - Migrasi ke instans DB yang berbeda dengan kapasitas I/O yang lebih tinggi.
  - Jika Anda sudah menggunakan penyimpanan penyimpanan IOPS termasuk Influx, sediakan jenis penyimpanan yang IOPS disertakan lebih tinggi.
- Jika aplikasi klien Anda menyimpan data Domain Name Service (DNS) dari instans DB Anda, tetapkan nilai time-to-live (TTL) kurang dari 30 detik. Alamat IP yang mendasari untuk instans DB dapat berubah setelah failover. Caching DNS data untuk waktu yang lama dapat menyebabkan kegagalan koneksi. Aplikasi Anda mungkin mencoba untuk menghubungkan ke alamat IP yang sudah tidak berada dalam layanan.

RAMRekomendasi instans DB

Praktik terbaik kinerja Amazon TimeStream untuk InfluxDB adalah mengalokasikan cukup RAM sehingga set kerja Anda hampir sepenuhnya berada dalam memori. Set kerja adalah data dan indeks yang sering Anda gunakan pada instans. Makin banyak Anda menggunakan instans DB, makin banyak set kerja yang akan tumbuh.

Keamanan di Timestream untuk InfluxDB

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

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. Model tanggung jawab bersama menjelaskan hal ini sebagai keamanan cloud dan keamanan dalam cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan AWS layanan di AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Efektivitas keamanan kami diuji dan diverifikasi secara rutin oleh auditor pihak ketiga sebagai bagian dari program kepatuhan AWS. Untuk mempelajari tentang program kepatuhan yang berlaku untuk Timestream for InfluxDB, lihat AWS Layanan dalam Lingkup berdasarkan Program Kepatuhan.

- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, termasuk sensitivitas data, persyaratan perusahaan, serta hukum dan peraturan yang berlaku.

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

Topik

Gambaran Umum

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan Amazon Timestream untuk InfluxDB. Topik berikut menunjukkan cara mengonfigurasi Amazon Timestream untuk InfluxDB untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga mempelajari cara menggunakan AWS layanan lain yang membantu Anda memantau dan mengamankan Amazon Timestream untuk sumber daya InfluxDB.

Anda dapat mengelola akses ke Amazon Timestream untuk sumber daya InfluxDB dan database Anda pada instans DB. Metode yang Anda gunakan untuk mengelola akses bergantung pada jenis tugas yang perlu dilakukan pengguna dengan Amazon Timestream untuk InfluxDB:

- Jalankan instans DB Anda di Virtual Private Cloud (VPC) berdasarkan VPC layanan Amazon untuk kontrol akses jaringan.
- Gunakan kebijakan AWS Identity and Access Management (IAM) untuk menetapkan izin yang menentukan siapa yang diizinkan mengelola Amazon Timestream untuk sumber daya InfluxDB. Misalnya, Anda dapat menggunakan IAM untuk menentukan siapa yang diizinkan untuk membuat, mendeskripsikan, memodifikasi, dan menghapus instans DB, menandai sumber daya, atau memodifikasi grup keamanan.
- Gunakan grup keamanan untuk mengontrol alamat IP atau EC2 instans Amazon yang dapat terhubung ke database Anda pada instans DB. Saat pertama kali membuat instans DB, instans hanya dapat diakses melalui aturan yang ditentukan oleh grup keamanan terkait.
- Gunakan koneksi Secure Socket Layer (SSL) atau Transport Layer Security (TLS) dengan instans DB Anda.
- Gunakan fitur keamanan mesin InfluxDB Anda untuk mengontrol siapa yang dapat masuk ke database pada instans DB. Fitur ini berfungsi seolah-olah basis data berada di jaringan lokal Anda. Untuk informasi selengkapnya, lihat Keamanan di Timestream untuk InfluxDB.

> **Note**
>
> Anda hanya perlu mengonfigurasikan keamanan untuk kasus penggunaan Anda. Anda tidak perlu mengonfigurasi akses keamanan untuk proses yang dikelola Amazon TimeStream untuk InfluxDB. Ini termasuk membuat cadangan, mereplikasi data antara instans DB utama dan replika baca, dan proses lainnya.

Topik

Keamanan umum

Topik

- Akses jaringan
- Dependensi
- Bucket S3

Izin

Pengguna InfluxDB harus diberikan izin hak istimewa paling sedikit. Hanya token yang diberikan kepada pengguna tertentu, bukan token operator, yang harus digunakan selama migrasi.

Timestream untuk InfluxDB menggunakan IAM izin untuk mengontrol izin pengguna. Kami menyarankan pengguna diberikan akses ke tindakan dan sumber daya spesifik yang mereka butuhkan. Untuk informasi selengkapnya, lihat Berikan akses hak istimewa paling sedikit.

Akses jaringan

Skrip migrasi Influx dapat berfungsi secara lokal, memigrasikan data antara dua instance InfluxDB pada sistem yang sama, tetapi diasumsikan bahwa kasus penggunaan utama untuk migrasi akan memigrasikan data ke seluruh jaringan, baik jaringan lokal maupun publik. Dengan ini muncul pertimbangan keamanan. Skrip migrasi Influx akan, secara default, memverifikasi TLS sertifikat untuk instance dengan TLS diaktifkan: kami menyarankan agar pengguna mengaktifkan TLS instans InfluxDB mereka dan tidak menggunakan opsi untuk skrip. `--skip-verify`

Kami menyarankan Anda menggunakan daftar izin untuk membatasi lalu lintas jaringan dari sumber yang Anda harapkan. Anda dapat melakukan ini dengan membatasi lalu lintas jaringan ke instans InfluxDB hanya dari yang diketahui. IPs

Dependensi

Versi utama terbaru dari semua dependensi harus digunakan, termasuk Influx, InfluxDBCLI, Python, modul Requests, dan dependensi opsional seperti dan. `mountpoint-s3` `rclone`

Bucket S3

Jika bucket S3 digunakan sebagai penyimpanan sementara untuk migrasi, sebaiknya aktifkan, versiTLS, dan nonaktifkan akses publik.

Menggunakan bucket S3 untuk migrasi

1. Buka AWS Management Console, navigasikan ke Amazon Simple Storage Service dan kemudian pilih Bucket.

2. Pilih ember yang ingin Anda gunakan.
3. Pilih tab Izin.
4. Di bagian bawah Blokir akses publik (pengaturan bucket), pilih Edit.
5. Periksa Blokir semua akses publik.
6. Pilih Simpan perubahan.
7. Di Bawah Kebijakan bucket, pilih Edit.
8. Masukkan yang berikut ini, ganti **<example-bucket>**dengan nama bucket Anda, untuk menerapkan penggunaan TLS versi 1.2 atau yang lebih baru untuk koneksi:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnforceTLSv12orHigher",
            "Principal": {
                "AWS": "*"
            },
            "Action": [
                "s3:*"
            ],
            "Effect": "Deny",
            "Resource": [
                "arn:aws:s3:::<example bucket>/*",
                "arn:aws:s3:::<example bucket>"
            ],
            "Condition": {
                "NumericLessThan": {
                    "s3:TlsVersion": 1.2
                }
            }
        }
    ]
}
```

9. Pilih Simpan perubahan.
10. Pilih tab Properti.
11. Di bawah Penentuan Versi Bucket, pilih Edit.
12. Periksa Aktifkan.
13. Pilih Simpan perubahan.

Untuk informasi tentang praktik keamanan terbaik Amazon S3 bucket, lihat Praktik terbaik keamanan untuk Amazon Simple Storage Service.

Otentikasi database dengan Amazon Timestream untuk InfluxDB

Amazon Timestream untuk InfluxDB mendukung dua cara untuk mengautentikasi pengguna database.

Kata sandi dan akses Otentikasi basis data Token menggunakan metode otentikasi yang berbeda ke database. Oleh karena itu, pengguna tertentu dapat masuk ke basis data dengan menggunakan hanya satu metode autentikasi. Dalam kedua kasus InfluxDB melakukan semua administrasi akun pengguna dan API token.

Autentikasi kata sandi

Selama proses pembuatan instans DB InfluxDB, Anda membuat organisasi, pengguna, dan kata sandi. Pengguna memiliki izin untuk mengelola semua yang ada di Timestream Anda untuk instans DB InfluxDB. Dengan kombinasi nama pengguna dan kata sandi ini, Anda akan dapat LogIn masuk ke instans Anda menggunakan InfluxUI dan juga menggunakan Influx CLI untuk menghasilkan token operator.

Token operator diperlukan untuk membuat pengguna, menghapus bucket, organisasi, dll. Untuk informasi selengkapnya, lihat Opsi autentikasi basis data.

APItoken

APIToken InfluxDB memastikan interaksi yang aman antara InfluxDB dan alat eksternal seperti klien atau aplikasi. APIToken milik pengguna tertentu dan mengidentifikasi izin InfluxDB dalam organisasi pengguna.

Ada tiga jenis API token di InfluxDB:

- Token Operator: Memberikan akses baca dan tulis lengkap ke semua organisasi dan semua sumber daya organisasi di OSS InfluxDB 2.x. Beberapa operasi, misalnya, mengambil konfigurasi server, memerlukan izin operator. Untuk membuat token operator secara manual dengan UI InfluxDB, `api/v2`API, atau Influx CLI setelah proses penyiapan selesai, Anda harus menggunakan token operator yang ada atau nama pengguna dan kata sandi Anda. Untuk membuat token operator baru tanpa menggunakan yang sudah ada, lihat autentikasi pemulihan influxd. CLI

> **Important**
>
> Karena token operator memiliki akses baca dan tulis penuh ke semua organisasi dalam database, kami sarankan untuk membuat token All-Access untuk setiap organisasi dan menggunakannya untuk mengelola InfluxDB. Ini membantu mencegah interaksi yang tidak disengaja di seluruh organisasi.

- All-Access API Token: Memberikan akses baca dan tulis lengkap ke semua sumber daya dalam suatu organisasi.
- Baca/Tulis Token: Memberikan akses baca, akses tulis, atau keduanya ke bucket tertentu dalam suatu organisasi.

Semua InfluxDb token adalah token berumur panjang tanpa tanggal kedaluwarsa yang ditetapkan, jadi tidak disarankan untuk menggunakan operator Anda atau semua token akses untuk mengirim data pemantauan dari klien Anda atau agen Telegraf tidak untuk menyematkannya dalam aplikasi dasbor Anda. Untuk aplikasi ini buat token baca/tulis hanya dengan izin yang diperlukan untuk menyelesaikan pekerjaan. Untuk informasi selengkapnya tentang cara membuat token InfluxDB, lihat Membuat token.

Rahasia

Token operator InfluxDB dihasilkan pada pengaturan instans; jenis token lainnya, seperti token all-access dan read/write, dapat dibuat menggunakan fungsi Influx, Influx CLI v2API, atau Timestream for InfluxDB Multi-user rotation. Lihat Mengelola API token untuk mengetahui cara membuat, melihat, menetapkan, dan menghapus token.

Kami menyarankan Anda memutar Timestream untuk token InfluxDB yang sering menggunakan AWS Secrets Manager dan menyimpan token melalui variabel lingkungan. Lihat Gunakan Token untuk penggunaan token dalam variabel lingkungan dan Memutar rahasianya cara memutar Timestream untuk pengguna dan token InfluxDB.

Lihat juga:

- Keamanan infrastruktur di Amazon Timestream untuk InfluxDB
- Praktik terbaik keamanan untuk Timestream untuk InfluxDB

Bagaimana Amazon Timestream untuk InfluxDB menggunakan rahasia

Timestream untuk InfluxDB mendukung otentikasi nama pengguna dan kata sandi melalui antarmuka pengguna, dan kredensi token untuk koneksi klien dan aplikasi yang paling tidak memiliki hak istimewa. Timestream untuk pengguna InfluxDB memiliki `allAccess` izin dalam organisasi mereka sementara token dapat memiliki serangkaian izin apa pun. Mengikuti praktik terbaik untuk manajemen API token yang aman, pengguna harus dibuat untuk mengelola token untuk akses halus dalam suatu organisasi. Informasi tambahan tentang praktik terbaik admin dengan Timestream untuk InfluxDB dapat ditemukan di dokumentasi Influxdata.

AWS Secrets Manager adalah layanan penyimpanan rahasia yang dapat Anda gunakan untuk melindungi kredensi database, API kunci, dan informasi rahasia lainnya. Kemudian dalam kode Anda, Anda dapat mengganti kredensi hardcode dengan panggilan ke API Secrets Manager. Ini membantu memastikan bahwa rahasia tidak dapat dikompromikan oleh seseorang yang memeriksa kode Anda, karena rahasianya tidak ada. Untuk gambaran umum tentang Secrets Manager, lihat Apa itu AWS Secrets Manager.

Saat Anda membuat instance database, Timestream untuk InfluxDB secara otomatis membuat rahasia admin untuk Anda gunakan dengan fungsi rotasi multi-pengguna. AWS Lambda Untuk memutar Timestream untuk pengguna dan token InfluxDB, Anda harus membuat rahasia baru dengan tangan untuk setiap pengguna atau token yang ingin Anda putar. Setiap rahasia dapat dikonfigurasi untuk memutar sesuai jadwal dengan menggunakan fungsi Lambda. Proses untuk mengatur rahasia berputar baru terdiri dari mengunggah kode fungsi Lambda, mengonfigurasi peran Lambda, mendefinisikan rahasia baru, dan mengonfigurasi jadwal rotasi rahasia.

Apa yang ada di rahasia

Saat Anda menyimpan Timestream untuk kredensi pengguna InfluxDB secara rahasia, gunakan format berikut.

Pengguna tunggal:

```
{
  "engine": "<required: must be set to 'timestream-influxdb'>",
  "username": "<required: username>",
  "password": "<required: password>",
  "dbIdentifier": "<required: DB identifier>"
}
```

Saat Anda membuat Timestream untuk instans InfluxDB, rahasia admin secara otomatis disimpan di Secrets Manager dengan kredenal yang akan digunakan dengan fungsi Lambda multi-pengguna. Atur `adminSecretArn` ke `Authentication Properties Secret Manager ARN` nilai yang ditemukan di halaman ringkasan instans DB atau ke rahasia admin. ARN Untuk membuat rahasia admin baru, Anda harus sudah memiliki kredensi terkait dan kredensialnya harus memiliki hak istimewa admin.

Saat Anda menyimpan Timestream untuk kredenal token InfluxDB dalam rahasia, gunakan format berikut.

Multi-pengguna:

```
{
  "engine": "<required: must be set to 'timestream-influxdb'>",
  "org": "<required: organization to associate token with>",
  "adminSecretArn": "<required: ARN of the admin secret>",
  "type": "<required: allAccess or operator or custom>",
  "dbIdentifier": "<required: DB identifier>",
  "token": "<required unless generating a new token: token being rotated>",
  "writeBuckets": "<optional: list of bucketIDs for custom type token, must be input
 within plaintext panel, for example ['id1','id2']>",
  "readBuckets": "<optional: list of bucketIDs for custom type token, must be input
 within plaintext panel, for example ['id1','id2']>",
  "permissions": "<optional: list of permissions for custom type token, must be input
 within plaintext panel, for example ['write-tasks','read-tasks']>"
}
```

Saat Anda menyimpan Timestream untuk kredensi admin InfluxDB secara rahasia, gunakan format berikut:

Rahasia admin:

```
{
  "engine": "<required: must be set to 'timestream-influxdb'>",
  "username": "<required: username>",
  "password": "<required: password>",
  "dbIdentifier": "<required: DB identifier>",
  "organization": "<optional: initial organization>",
  "bucket": "<optional: initial bucket>"
}
```

Untuk mengaktifkan rotasi otomatis untuk rahasia, rahasianya harus dalam JSON struktur yang benar. Lihat Memutar rahasianya cara memutar Timestream untuk rahasia InfluxDB.

Memodifikasi rahasia

Kredensi yang dihasilkan selama proses pembuatan instans Timestream for InfluxDB disimpan dalam rahasia Secrets Manager di akun Anda. Objek GetDbInstancerespon berisi `influxAuthParametersSecretArn` yang menyimpan Amazon Resource Name (ARN) ke rahasia tersebut. Rahasianya hanya akan diisi setelah Timestream Anda untuk instans InfluxDB tersedia. Ini adalah READONLY salinan karena rahasia ini tidak memengaruhi instance DB yang dibuat. updates/modifications/deletions Jika Anda menghapus rahasia ini, APIrespons masih akan merujuk ke rahasia yang dihapusARN.

Untuk membuat token baru di instans Timestream for InfluxDB daripada menyimpan kredenal token yang ada, Anda dapat membuat token non-operator dengan membiarkan `token` nilai kosong dalam rahasia dan menggunakan fungsi rotasi multi-pengguna dengan variabel lingkungan Lambda yang disetel ke`AUTHENTICATION_CREATION_ENABLED`. `true` Jika Anda membuat token baru, izin yang ditentukan dalam rahasia ditetapkan ke token dan tidak dapat diubah setelah rotasi pertama yang berhasil. Untuk informasi selengkapnya tentang memutar rahasia, lihat Rotating Secrets Manager AWS Secrets Secrets.

Jika rahasia dihapus, pengguna atau token terkait di Timestream untuk instans InfluxDB tidak akan dihapus.

Memutar rahasianya

Anda menggunakan Timestream untuk fungsi Lambda rotasi tunggal dan multi-pengguna InfluxDB untuk memutar Timestream untuk pengguna InfluxDB dan kredenal token. Gunakan fungsi Lambda pengguna tunggal untuk memutar kredensional pengguna untuk Timestream Anda untuk instans InfluxDB, dan gunakan fungsi Lambda multi-pengguna untuk memutar kredenal token untuk Timestream Anda untuk instans InfluxDB.

Memutar pengguna dan token dengan fungsi Lambda tunggal dan multi-pengguna adalah opsional. Timestream untuk kredensial InfluxDB tidak pernah kedaluwarsa dan kredenal apa pun yang terbuka menimbulkan risiko tindakan jahat terhadap instans DB Anda. Keuntungan memutar Timestream untuk kredensial InfluxDB dengan Secrets Manager adalah lapisan keamanan tambahan yang membatasi vektor serangan kredenal yang terbuka ke jendela waktu hingga siklus rotasi berikutnya. Jika tidak ada mekanisme rotasi untuk instans DB Anda, kredenal apa pun yang terbuka akan valid sampai dihapus secara manual.

Anda dapat mengonfigurasi Secrets Manager untuk secara otomatis memutar rahasia untuk Anda sesuai dengan jadwal yang Anda tentukan. Ini memungkinkan Anda mengganti rahasia jangka panjang dengan rahasia jangka pendek, yang membantu mengurangi risiko kompromi secara signifikan. Untuk informasi selengkapnya tentang memutar rahasia dengan Secrets Manager, lihat Rotate AWS Secrets Manager Secrets Secrets.

Memutar pengguna

Saat Anda memutar pengguna dengan fungsi Lambda pengguna tunggal, kata sandi acak baru akan diberikan kepada pengguna setelah setiap rotasi. Untuk informasi selengkapnya tentang cara mengaktifkan rotasi otomatis, lihat Mengatur rotasi otomatis untuk AWS rahasia Secrets Manager non-database.

Memutar rahasia admin

Untuk memutar rahasia admin, Anda menggunakan fungsi rotasi pengguna tunggal. Anda perlu menambahkan `dbIdentifier` nilai `engine` dan ke rahasia karena nilai-nilai tersebut tidak secara otomatis diisi pada inisialisasi DB. Lihat Apa yang ada di rahasia untuk template rahasia lengkap.

Untuk menemukan rahasia admin untuk instans Timestream untuk InfluxDB, Anda menggunakan rahasia admin ARN dari halaman ringkasan instans Timestream untuk InfluxDB. Disarankan agar Anda memutar semua Timestream untuk rahasia admin InfluxDB karena pengguna admin telah meningkatkan izin untuk Timestream untuk instans InfluxDB.

Fungsi rotasi Lambda

Anda dapat memutar Timestream untuk pengguna InfluxDB dengan fungsi rotasi pengguna tunggal dengan menggunakan rahasia baru dan menambahkan bidang yang diperlukan untuk Timestream Anda untuk pengguna InfluxDB. Apa yang ada di rahasia Untuk informasi selengkapnya tentang rotasi rahasia fungsi Lambda, lihat Rotasi oleh fungsi Lambda.

Fungsi rotasi pengguna tunggal mengautentikasi dengan Timestream untuk instans DB InfluxDB menggunakan kredensional yang ditentukan dalam rahasia, kemudian menghasilkan kata sandi acak baru dan menetapkan kata sandi baru untuk pengguna. Untuk informasi selengkapnya tentang rotasi rahasia fungsi Lambda, lihat Rotasi oleh fungsi Lambda.

Izin peran eksekusi fungsi Lambda

Gunakan IAM kebijakan berikut sebagai peran untuk fungsi Lambda pengguna tunggal. Kebijakan ini memberi fungsi Lambda izin yang diperlukan untuk melakukan rotasi rahasia Timestream bagi pengguna InfluxDB.

Ganti semua item yang tercantum di bawah ini dalam IAM kebijakan dengan nilai dari AWS akun Anda:

- {rotating_secret_arn} — Rahasia yang diputar dapat ditemukan di detail rahasia Secrets Manager. ARN
- {db_instance_arn} — Timestream untuk instans InfluxDB ARN dapat ditemukan di halaman ringkasan instans Timestream untuk InfluxDB.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "secretsmanager:UpdateSecretVersionStage"
            ],
     "Resource": "{rotating_secret_arn}"
        },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword"
            ],
            "Resource": "*"
        },
        {
            "Action": [
                "timestream-influxdb:GetDbInstance"
            ],
     "Resource": "{db_instance_arn}",
            "Effect": "Allow"
        }
    ]
}
```

Token berputar

Anda dapat memutar Timestream untuk token InfluxDB dengan fungsi rotasi multi-pengguna dengan menggunakan rahasia baru dan menambahkan bidang yang diperlukan untuk Timestream Anda untuk token InfluxDB. Apa yang ada di rahasia Untuk informasi selengkapnya tentang rotasi rahasia fungsi Lambda, lihat Rotasi oleh fungsi Lambda.

Anda dapat memutar Timestream untuk token InfluxDB dengan menggunakan Timestream untuk fungsi Lambda multi-pengguna InfluxDB. Setel variabel `AUTHENTICATION_CREATION_ENABLED` lingkungan ke `true` dalam konfigurasi Lambda untuk mengaktifkan pembuatan token. Untuk membuat token baru, gunakan Apa yang ada di rahasia untuk nilai rahasia Anda. Hilangkan pasangan `token` kunci-nilai dalam rahasia baru dan atur `type` ke`allAccess`, atau tentukan izin tertentu dan atur jenisnya. `custom` Fungsi rotasi akan membuat token baru selama siklus rotasi pertama. Anda tidak dapat mengubah izin token dengan mengedit rahasia setelah rotasi dan rotasi berikutnya akan menggunakan izin yang ditetapkan dalam instans DB.

Fungsi rotasi Lambda

Fungsi rotasi multi-pengguna memutar kredensi token dengan membuat token identik izin baru menggunakan kredensi admin dalam rahasia admin. Fungsi Lambda memvalidasi nilai token dalam rahasia sebelum membuat token pengganti, menyimpan nilai token baru dalam rahasia, dan menghapus token lama. Jika fungsi Lambda membuat token baru, pertama-tama akan memvalidasi bahwa variabel `AUTHENTICATION_CREATION_ENABLED` lingkungan disetel ke`true`, bahwa tidak ada nilai token dalam rahasia, dan bahwa jenis token bukan operator tipe.

Izin peran eksekusi fungsi Lambda

Gunakan IAM kebijakan berikut sebagai peran untuk fungsi Lambda multi-pengguna. Kebijakan ini memberi fungsi Lambda izin yang diperlukan untuk melakukan rotasi rahasia Timestream untuk token InfluxDB.

Ganti semua item yang tercantum di bawah ini dalam IAM kebijakan dengan nilai dari AWS akun Anda:

- {rotating_secret_arn} — Rahasia yang diputar dapat ditemukan di detail rahasia Secrets Manager. ARN
- {authentication_properties_admin_secret_arn} — Timestream untuk rahasia admin InfluxDB dapat ditemukan di halaman ringkasan instans Timestream untuk InfluxDB. ARN
- {db_instance_arn} — Timestream untuk instans InfluxDB ARN dapat ditemukan di halaman ringkasan instans Timestream untuk InfluxDB.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "secretsmanager:UpdateSecretVersionStage"
            ],
     "Resource": "{rotating_secret_arn}"
        },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
     "Resource": "{authentication_properties_admin_secret_arn}"
        },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword"
            ],
            "Resource": "*"
        },
        {
            "Action": [
                "timestream-influxdb:GetDbInstance"
            ],
     "Resource": "{db_instance_arn}",
            "Effect": "Allow"
        }
    ]
}
```

Perlindungan data di Timestream untuk InfluxDB

Model tanggung jawab AWS bersama model tanggung berlaku untuk perlindungan data di Amazon Timestream untuk InfluxDB. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk

mempertahankan kendali atas konten yang di-host pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Untuk informasi selengkapnya tentang privasi data, lihat Privasi Data FAQ. Untuk informasi tentang perlindungan data di Eropa, lihat Model Tanggung Jawab AWS Bersama dan posting GDPR blog di Blog AWS Keamanan.

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensil dan mengatur pengguna individu dengan 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 otentikasi multi-faktor (MFA) dengan setiap akun.
- GunakanSSL/TLSuntuk berkomunikasi dengan AWS sumber daya. Kami membutuhkan TLS 1.2 dan merekomendasikan TLS 1.3.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail. Untuk informasi tentang penggunaan CloudTrail jejak untuk menangkap AWS aktivitas, lihat Bekerja dengan CloudTrail jejak di AWS CloudTrail Panduan Pengguna.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya 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 FIPS 140-3 modul kriptografi yang divalidasi saat mengakses AWS melalui antarmuka baris perintah atau, gunakan titik akhir. API FIPS Untuk informasi selengkapnya tentang FIPS titik akhir yang tersedia, lihat Federal Information Processing Standard (FIPS) 140-3.

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 Timestream untuk InfluxDB atau lainnya Layanan AWS menggunakan konsol,,API, AWS CLI atau. AWS SDKs 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. Jika Anda memberikan URL ke server eksternal, kami sangat menyarankan agar Anda tidak menyertakan informasi kredensil dalam URL untuk memvalidasi permintaan Anda ke server tersebut.

Untuk informasi lebih rinci tentang Timestream untuk topik perlindungan data InfluxDB seperti Enkripsi saat Istirahat dan Manajemen Kunci, pilih salah satu topik yang tersedia di bawah ini.

Topik

Enkripsi diam

Timestream untuk enkripsi InfluxDB saat istirahat memberikan keamanan yang ditingkatkan dengan mengenkripsi semua data Anda saat istirahat menggunakan kunci enkripsi yang disimpan di ().AWS Key Management ServiceAWS KMS Fungsi ini membantu mengurangi beban operasional dan kompleksitas yang terlibat dalam melindungi data sensitif. Dengan enkripsi saat diam, Anda dapat membuat aplikasi yang sensitif terhadap keamanan yang memenuhi persyaratan kepatuhan dan peraturan enkripsi yang ketat.

- Enkripsi diaktifkan secara default pada Timestream Anda untuk instans DB InfluxDB, dan tidak dapat dimatikan. Algoritma enkripsi standar industri AES -256 adalah algoritma enkripsi default yang digunakan.
- AWS KMS digunakan untuk enkripsi saat istirahat di Timestream untuk InfluxDB.
- Anda tidak perlu memodifikasi aplikasi klien instans DB Anda untuk menggunakan enkripsi.

Enkripsi bergerak

Semua data Timestream untuk InfluxDB Anda dienkripsi dalam perjalanan. Secara default, semua komunikasi ke dan dari Timestream untuk InfluxDB dilindungi dengan menggunakan enkripsi Transport Layer Security ()TLS.

Lalu lintas ke dan dari Amazon Timestream untuk InfluxDB diamankan menggunakan versi 1.2 atau 1.3 yang didukungTLS.

Identity and Access Management untuk Amazon Timestream untuk InfluxDB

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. IAMadministrator mengontrol siapa yang

dapat diautentikasi (masuk) dan diberi wewenang (memiliki izin) untuk menggunakan Timestream untuk sumber daya InfluxDB. IAMadalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

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 IAM pengguna, atau dengan mengambil peranIAM.

Anda dapat masuk AWS sebagai identitas federasi dengan menggunakan kredensi yang disediakan melalui sumber identitas. AWS IAM Identity Center Pengguna (Pusat IAM Identitas), autentikasi masuk tunggal perusahaan Anda, dan kredensi Google atau Facebook Anda adalah contoh identitas federasi. Saat Anda masuk sebagai identitas federasi, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan IAM peran. 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 menggunakan

metode yang disarankan untuk menandatangani permintaan sendiri, lihat Versi AWS Tanda Tangan 4 untuk API permintaan di Panduan IAM Pengguna.

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

Pengguna dan grup IAM

IAMPengguna adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, sebaiknya mengandalkan kredensi sementara daripada membuat IAM pengguna yang memiliki kredensi jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan khusus yang memerlukan kredensil jangka panjang dengan IAM pengguna, kami sarankan Anda memutar kunci akses. Untuk informasi selengkapnya, lihat Memutar kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensi jangka panjang di IAMPanduan Pengguna.

IAMGrup adalah identitas yang menentukan kumpulan IAM pengguna. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin bagi beberapa pengguna sekaligus. Grup mempermudah manajemen izin untuk sejumlah besar pengguna sekaligus. Misalnya, Anda dapat memiliki grup bernama IAMAdminsdan memberikan izin grup tersebut untuk mengelola sumber dayaIAM.

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

IAMperan

IAMPeran adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus. Ini mirip dengan IAM pengguna, tetapi tidak terkait dengan orang tertentu. Untuk mengambil IAM peran sementara di dalam AWS Management Console, Anda dapat beralih dari pengguna ke IAM peran (konsol). Anda dapat mengambil peran dengan memanggil AWS CLI atau AWS API operasi atau dengan menggunakan kustomURL. Untuk informasi selengkapnya tentang metode penggunaan peran, lihat Metode untuk mengambil peran dalam Panduan IAM Pengguna.

IAMperan dengan kredensi sementara berguna dalam situasi berikut:

- Akses pengguna terfederasi – Untuk menetapkan izin ke identitas terfederasi, Anda membuat peran dan menentukan izin untuk peran tersebut. Ketika identitas terfederasi mengautentikasi, identitas tersebut terhubung dengan peran dan diberi izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat Membuat peran untuk Penyedia Identitas pihak ketiga di Panduan IAM Pengguna. Jika Anda menggunakan Pusat IAM Identitas, Anda mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah diautentikasi, Pusat IAM Identitas menghubungkan izin yang disetel ke peran. IAM Untuk informasi tentang set izin, lihat Set izin dalam Panduan Pengguna AWS IAM Identity Center .
- Izin IAM pengguna sementara — IAM Pengguna atau peran dapat mengambil IAM peran untuk sementara mengambil izin yang berbeda untuk tugas tertentu.
- Akses lintas akun — Anda dapat menggunakan IAM peran untuk memungkinkan seseorang (prinsipal tepercaya) di akun lain mengakses sumber daya di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapa Layanan AWS, Anda dapat melampirkan kebijakan secara langsung ke sumber daya (alih-alih menggunakan peran sebagai proxy). Untuk mempelajari perbedaan antara peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Akses sumber daya lintas akun di IAM Panduan Pengguna. IAM
- Akses lintas layanan — Beberapa Layanan AWS menggunakan fitur lain Layanan AWS. Misalnya, saat Anda melakukan panggilan dalam suatu layanan, biasanya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Sebuah layanan mungkin melakukannya menggunakan izin prinsipal yang memanggil, menggunakan peran layanan, atau peran terkait layanan.
  - Sesi akses teruskan (FAS) — Saat Anda menggunakan IAM pengguna atau peran untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FASmenggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. FASPermintaan 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 saat membuat FAS permintaan, lihat Meneruskan sesi akses.
  - Peran layanan — Peran layanan adalah IAMperan yang diasumsikan layanan untuk melakukan tindakan atas nama Anda. IAMAdministrator dapat membuat, memodifikasi, dan menghapus peran layanan dari dalamIAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS dalam IAMPanduan Pengguna.

  - Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke peran layanan. Layanan AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. IAMAdministrator dapat melihat, tetapi tidak mengedit izin untuk peran terkait layanan.
- Aplikasi yang berjalan di Amazon EC2 — Anda dapat menggunakan IAM peran untuk mengelola kredensil sementara untuk aplikasi yang berjalan pada EC2 instance dan membuat AWS CLI atau AWS API meminta. Ini lebih baik untuk menyimpan kunci akses dalam EC2 instance. Untuk menetapkan AWS peran ke EC2 instance dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instance yang dilampirkan ke instance. Profil instance berisi peran dan memungkinkan program yang berjalan pada EC2 instance untuk mendapatkan kredensi sementara. Untuk informasi selengkapnya, lihat Menggunakan IAM peran untuk memberikan izin ke aplikasi yang berjalan di EC2 instans Amazon di IAMPanduan Pengguna.

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 JSON dokumen. Untuk informasi selengkapnya tentang struktur dan isi dokumen JSON kebijakan, lihat Ringkasan JSON kebijakan di Panduan IAM Pengguna.

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

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

IAMkebijakan menentukan izin untuk tindakan terlepas dari metode yang Anda gunakan untuk melakukan operasi. Misalnya, anggaplah Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan itu bisa mendapatkan informasi peran dari AWS Management Console, AWS CLI, atau AWS API.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan JSON izin yang dapat Anda lampirkan ke identitas, seperti pengguna, grup IAM pengguna, atau peran. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Menentukan IAM izin khusus dengan kebijakan yang dikelola pelanggan di Panduan Pengguna. IAM

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan yang dikelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran dalam. Akun AWS Kebijakan AWS terkelola mencakup kebijakan terkelola dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan terkelola atau kebijakan sebaris, lihat Memilih antara kebijakan terkelola dan kebijakan sebaris di IAMPanduan Pengguna.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen JSON kebijakan yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan IAM peran dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan prinsipal dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

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

Daftar kontrol akses (ACLs)

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

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung. ACLs Untuk mempelajari selengkapnyaACLs, lihat Ikhtisar daftar kontrol akses (ACL) di Panduan Pengembang Layanan Penyimpanan Sederhana Amazon.

Jenis-jenis kebijakan lain

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

- Batas izin — Batas izin adalah fitur lanjutan tempat Anda menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas (pengguna atau peran). IAM IAM Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya tentang batas izin, lihat Batas izin untuk IAM entitas di IAMPanduan Pengguna.
- Kebijakan kontrol layanan (SCPs) — SCPs adalah JSON kebijakan yang menentukan izin maksimum untuk organisasi atau unit organisasi (OU) di AWS Organizations. AWS Organizations adalah layanan untuk mengelompokkan dan mengelola secara terpusat beberapa Akun AWS yang dimiliki bisnis Anda. Jika Anda mengaktifkan semua fitur dalam suatu organisasi, maka Anda dapat menerapkan kebijakan kontrol layanan (SCPs) ke salah satu atau semua akun Anda. SCPMembatasi izin untuk entitas di akun anggota, termasuk masing-masing Pengguna root akun AWS. Untuk informasi selengkapnya tentang Organizations danSCPs, lihat Kebijakan kontrol layanan di Panduan AWS Organizations Pengguna.
- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda berikan sebagai parameter ketika Anda membuat sesi sementara secara programatis untuk peran atau pengguna terfederasi. Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan secara tegas dalam salah satu kebijakan ini membatalkan izin. Untuk informasi selengkapnya, lihat Kebijakan sesi di Panduan IAM Pengguna.

Berbagai jenis kebijakan

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

Bagaimana Amazon Timestream untuk InfluxDB bekerja dengan IAM

IAMfitur yang dapat Anda gunakan dengan Amazon Timestream untuk InfluxDB

IAMfitur	Timestream untuk dukungan InfluxDB
Kebijakan berbasis identitas	Ya
Kebijakan berbasis sumber daya	Tidak
Tindakan kebijakan	Ya
Sumber daya kebijakan	Ya
Kunci kondisi kebijakan	Tidak
ACLs	Tidak
ABAC(tag dalam kebijakan)	Ya
Kredensial sementara	Ya
Izin prinsipal	Ya
Peran layanan	Tidak
Peran terkait layanan	Ya

Untuk mendapatkan tampilan tingkat tinggi tentang bagaimana Timestream untuk InfluxDB dan AWS layanan lainnya bekerja dengan sebagian besar IAM fitur, lihat AWS layanan yang bekerja dengan IAM dalam Panduan Pengguna. IAM

Kebijakan berbasis identitas untuk Timestream untuk InfluxDB

Mendukung kebijakan berbasis identitas: Ya

Kebijakan berbasis identitas adalah dokumen kebijakan JSON izin yang dapat Anda lampirkan ke identitas, seperti pengguna, grup IAM pengguna, atau peran. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Menentukan IAM izin khusus dengan kebijakan yang dikelola pelanggan di Panduan Pengguna. IAM

Dengan kebijakan IAM berbasis identitas, Anda dapat menentukan tindakan dan sumber daya yang diizinkan atau ditolak serta kondisi di mana tindakan diizinkan atau ditolak. Anda tidak dapat menentukan secara spesifik prinsipal dalam sebuah kebijakan berbasis identitas karena prinsipal berlaku bagi pengguna atau peran yang melekat kepadanya. Untuk mempelajari semua elemen yang dapat Anda gunakan dalam JSON kebijakan, lihat referensi elemen IAM JSON kebijakan di Panduan IAM Pengguna.

Contoh kebijakan berbasis identitas untuk Timestream untuk InfluxDB

Untuk melihat contoh Timestream untuk kebijakan berbasis identitas InfluxDB, lihat.. Contoh kebijakan berbasis identitas untuk Amazon Timestream untuk InfluxDB

Kebijakan berbasis sumber daya dalam Timestream untuk InfluxDB

Mendukung kebijakan berbasis sumber daya: Tidak

Kebijakan berbasis sumber daya adalah dokumen JSON kebijakan yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan IAM peran dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan prinsipal dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Untuk mengaktifkan akses lintas akun, Anda dapat menentukan seluruh akun atau IAM entitas di akun lain sebagai prinsipal dalam kebijakan berbasis sumber daya. Menambahkan prinsipal akun silang ke kebijakan berbasis sumber daya hanya setengah dari membangun hubungan kepercayaan. Ketika prinsipal dan sumber daya berbeda Akun AWS, IAM administrator di akun tepercaya juga harus memberikan izin entitas utama (pengguna atau peran) untuk mengakses sumber daya. Mereka memberikan izin dengan melampirkan kebijakan berbasis identitas kepada entitas. Namun, jika kebijakan berbasis sumber daya memberikan akses ke prinsipal dalam akun yang sama, tidak diperlukan kebijakan berbasis identitas tambahan. Untuk informasi selengkapnya, lihat Akses sumber daya lintas akun IAM di Panduan IAM Pengguna.

Tindakan kebijakan untuk Timestream untuk InfluxDB

Mendukung tindakan kebijakan: Ya

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

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

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

Untuk melihat daftar Timestream untuk tindakan InfluxDB, lihat Tindakan yang Ditentukan oleh Amazon Timestream untuk InfluxDB di Referensi Otorisasi Layanan.

Tindakan kebijakan di Timestream untuk InfluxDB menggunakan awalan berikut sebelum tindakan:

```
timestream-influxdb
```

Untuk menetapkan secara spesifik beberapa tindakan dalam satu pernyataan, pisahkan tindakan tersebut dengan koma.

```
"Action": [
      "timestream-influxdb:action1",
      "timestream-influxdb:action2"
         ]
```

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

```
"Action": "timestream-influxdb:Describe*"
```

Sumber daya kebijakan untuk Timestream untuk InfluxDB

Mendukung sumber daya kebijakan: Ya

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

Elemen `Resource` JSON kebijakan menentukan objek atau objek yang tindakan tersebut berlaku. Pernyataan harus menyertakan elemen `Resource` atau `NotResource`. Sebagai praktik terbaik, 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 menunjukkan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

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

Untuk melihat daftar Timestream untuk jenis sumber daya InfluxDB dan jenisnyaARNs, lihat Sumber Daya yang Ditentukan oleh Amazon Timestream untuk InfluxDB di Referensi Otorisasi Layanan. Untuk mempelajari tindakan yang dapat Anda tentukan ARN dari setiap sumber daya, lihat Tindakan yang Ditentukan oleh Amazon Timestream untuk InfluxDB.

Kunci kondisi kebijakan untuk Timestream untuk InfluxDB

Mendukung kunci kondisi kebijakan khusus layanan: Tidak

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

Elemen `Condition` (atau blok `Condition`) akan memungkinkan Anda menentukan kondisi yang menjadi dasar suatu pernyataan berlaku. Elemen `Condition` bersifat opsional. Anda dapat membuat ekspresi bersyarat 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 sebuah pernyataan, atau beberapa kunci dalam elemen `Condition` tunggal, maka AWS akan mengevaluasinya 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 IAM pengguna untuk mengakses sumber daya hanya jika ditandai dengan nama

IAM pengguna mereka. Untuk informasi selengkapnya, lihat elemen IAM kebijakan: variabel dan tag di Panduan IAM Pengguna.

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 IAM Pengguna.

Daftar kontrol akses (ACLs) di Timestream untuk InfluxDB

MendukungACLs: Tidak

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

Kontrol akses berbasis atribut (ABAC) dengan Timestream untuk InfluxDB

Mendukung ABAC (tag dalam kebijakan): Ya

Attribute-based access control (ABAC) adalah strategi otorisasi yang mendefinisikan izin berdasarkan atribut. Dalam AWS, atribut ini disebut tag. Anda dapat melampirkan tag ke IAM entitas (pengguna atau peran) dan ke banyak AWS sumber daya. Menandai entitas dan sumber daya adalah langkah pertama dari. ABAC Kemudian Anda merancang ABAC kebijakan untuk mengizinkan operasi ketika tag prinsipal cocok dengan tag pada sumber daya yang mereka coba akses.

ABACmembantu dalam lingkungan yang berkembang pesat dan membantu dengan situasi di mana manajemen kebijakan menjadi rumit.

Untuk mengendalikan akses berdasarkan tag, berikan informasi tentang tag di elemen kondisi dari kebijakan 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 untuk hanya beberapa jenis sumber daya, nilainya adalah Parsial.

Untuk informasi selengkapnyaABAC, lihat Menentukan izin dengan ABAC otorisasi di IAMPanduan Pengguna. Untuk melihat tutorial dengan langkah-langkah penyiapanABAC, lihat Menggunakan kontrol akses berbasis atribut (ABAC) di IAMPanduan Pengguna.

Menggunakan kredensi Temporary dengan Timestream untuk InfluxDB

Mendukung kredensi 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 berfungsi IAM di IAMPanduan Pengguna.

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 link sign-on (SSO) tunggal perusahaan Anda, proses tersebut secara otomatis membuat kredensi sementara. Anda juga akan secara otomatis membuat kredensial sementara ketika Anda masuk ke konsol sebagai seorang pengguna lalu beralih peran. Untuk informasi selengkapnya tentang beralih peran, lihat Beralih dari pengguna ke IAM peran (konsol) di Panduan IAM Pengguna.

Anda dapat secara manual membuat kredensi sementara menggunakan atau. AWS CLI AWS API 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 Kredensi keamanan sementara di. IAM

Izin utama lintas layanan untuk Timestream untuk InfluxDB

Mendukung sesi akses maju (FAS): Ya

Saat Anda menggunakan IAM pengguna atau peran untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FASmenggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. FASPermintaan 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 saat membuat FAS permintaan, lihat Meneruskan sesi akses.

Peran layanan untuk Timestream untuk InfluxDB

Mendukung peran layanan: Tidak

Peran layanan adalah IAMperan yang diasumsikan layanan untuk melakukan tindakan atas nama Anda. IAMAdministrator dapat membuat, memodifikasi, dan menghapus peran layanan dari dalamIAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS dalam IAMPanduan Pengguna.

> **Warning**
>
> Mengubah izin untuk peran layanan dapat merusak Timestream untuk fungsionalitas InfluxDB. Edit peran layanan hanya jika Timestream untuk InfluxDB memberikan panduan untuk melakukannya.

Peran terkait layanan untuk Timestream untuk InfluxDB

Mendukung peran terkait layanan: Ya

Peran terkait layanan adalah jenis peran layanan yang ditautkan ke. Layanan AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. IAMAdministrator dapat melihat, tetapi tidak mengedit izin untuk peran terkait layanan.

Untuk detail tentang membuat atau mengelola peran terkait layanan, lihat AWS layanan yang berfungsi dengannya. IAM Cari 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 Amazon Timestream untuk InfluxDB

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi Timestream untuk sumber daya InfluxDB. 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 butuhkan, IAM administrator dapat membuat IAM kebijakan. Administrator kemudian dapat menambahkan IAM kebijakan ke peran, dan pengguna dapat mengambil peran.

Untuk mempelajari cara membuat kebijakan IAM berbasis identitas menggunakan contoh dokumen kebijakan ini, lihat Membuat JSON IAM kebijakan (konsol) di Panduan Pengguna. IAM

Untuk detail tentang tindakan dan jenis sumber daya yang ditentukan oleh Timestream untuk InfluxDB, termasuk format ARNs untuk setiap jenis sumber daya, lihat Tindakan, Sumber Daya, dan Kunci Kondisi untuk Amazon Timestream for InfluxDB di Referensi Otorisasi Layanan.

Topik

- Praktik terbaik kebijakan
- Menggunakan Timestream untuk konsol InfluxDB

- Mengizinkan pengguna melihat izin mereka sendiri
- Mengakses satu bucket Amazon S3
- Mengizinkan semua operasi
- Buat, jelaskan, hapus, dan perbarui instans DB

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus Timestream untuk sumber daya InfluxDB di akun Anda. Tindakan ini membuat Akun AWS Anda dikenai biaya. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulailah dengan kebijakan AWS terkelola dan beralih ke izin hak istimewa paling sedikit — Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan AWS terkelola yang memberikan izin untuk banyak kasus penggunaan umum. Mereka tersedia di Anda Akun AWS. Kami menyarankan Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola AWS pelanggan yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat kebijakan AWSAWS terkelola atau kebijakan terkelola untuk fungsi pekerjaan di Panduan IAM Pengguna.
- Menerapkan izin hak istimewa paling sedikit — Saat Anda menetapkan izin dengan IAM kebijakan, berikan hanya izin yang diperlukan untuk melakukan tugas. Anda melakukannya dengan mendefinisikan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, yang juga dikenal sebagai izin dengan hak akses paling rendah. Untuk informasi selengkapnya tentang penggunaan IAM untuk menerapkan izin, lihat Kebijakan dan izin IAM di IAM Panduan Pengguna.
- Gunakan ketentuan dalam IAM kebijakan untuk membatasi akses lebih lanjut — Anda dapat menambahkan kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Misalnya, Anda dapat menulis kondisi kebijakan untuk menentukan bahwa semua permintaan harus dikirim menggunakanSSL. 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 IAM JSON kebijakan: Kondisi dalam Panduan IAM Pengguna.
- Gunakan IAM Access Analyzer untuk memvalidasi IAM kebijakan Anda guna memastikan izin yang aman dan fungsional — IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan mematuhi bahasa IAM kebijakan () JSON dan praktik terbaik. IAM IAMAccess 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 Memvalidasi kebijakan dengan IAM Access Analyzer di IAMPanduan Pengguna.

- Memerlukan otentikasi multi-faktor (MFA) - Jika Anda memiliki skenario yang mengharuskan IAM pengguna atau pengguna root di dalam Anda Akun AWS, aktifkan MFA untuk keamanan tambahan. Untuk meminta MFA kapan API operasi dipanggil, tambahkan MFA kondisi ke kebijakan Anda. Untuk informasi selengkapnya, lihat APIAkses aman dengan MFA di Panduan IAM Pengguna.

Untuk informasi selengkapnya tentang praktik terbaik diIAM, lihat Praktik terbaik keamanan IAM di Panduan IAM Pengguna.

Menggunakan Timestream untuk konsol InfluxDB

Untuk mengakses Amazon Timestream untuk konsol InfluxDB, Anda harus memiliki set izin minimum. Izin ini harus memungkinkan Anda untuk membuat daftar dan melihat detail tentang Timestream untuk sumber daya InfluxDB di sumber daya 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 Sebagai gantinya, izinkan akses hanya ke tindakan yang cocok dengan API operasi yang mereka coba lakukan.

Untuk memastikan bahwa pengguna dan peran masih dapat menggunakan konsol Timestream for InfluxDB, lampirkan juga Timestream for InfluxDB `ConsoleAccess` atau `ReadOnly` AWS kebijakan terkelola ke entitas. Untuk informasi selengkapnya, lihat Menambahkan izin ke pengguna di Panduan IAM Pengguna.

Mengizinkan pengguna melihat izin mereka sendiri

Contoh ini menunjukkan cara Anda membuat kebijakan yang memungkinkan IAM pengguna melihat kebijakan sebaris dan terkelola yang dilampirkan pada identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan ini di konsol atau secara terprogram menggunakan atau. AWS CLI AWS API

```
{
    "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": "*"
        }
    ]
}
```

Mengakses satu bucket Amazon S3

Dalam contoh ini, Anda ingin memberi IAM pengguna di AWS akun Anda akses ke salah satu bucket Amazon S3 Anda. `examplebucket` Anda juga ingin mengizinkan pengguna untuk menambah, memperbarui, dan menghapus objek.

Selain memberikan izin `s3:PutObject`, `s3:GetObject`, dan `s3:DeleteObject` bagi pengguna, kebijakan tersebut juga memberikan izin `s3:ListAllMyBuckets`, `s3:GetBucketLocation`, dan `s3:ListBucket`. Izin-izin tersebut adalah izin tambahan yang diperlukan oleh konsol tersebut. Selain itu, tindakan `s3:PutObjectAcl` dan `s3:GetObjectAcl` diperlukan untuk dapat menyalin, memotong, dan menempel objek di konsol. Untuk contoh panduan yang memberikan izin kepada

pengguna dan mengujinya menggunakan konsol, lihat Contoh panduan: Menggunakan kebijakan pengguna untuk mengontrol akses ke bucket Anda.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"ListBucketsInConsole",
         "Effect":"Allow",
         "Action":[
            "s3:ListAllMyBuckets"
         ],
         "Resource":"arn:aws:s3:::*"
      },
      {
         "Sid":"ViewSpecificBucketInfo",
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket",
            "s3:GetBucketLocation"
         ],
         "Resource":"arn:aws:s3:::examplebucket"
      },
      {
         "Sid":"ManageBucketContents",
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:GetObject",
            "s3:GetObjectAcl",
            "s3:DeleteObject"
         ],
         "Resource":"arn:aws:s3:::examplebucket/*"
      }
   ]
}
```

Mengizinkan semua operasi

Berikut ini adalah contoh kebijakan yang memungkinkan semua operasi di Timestream untuk InfluxDB.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream-influxdb:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Buat, jelaskan, hapus, dan perbarui instans DB

Kebijakan contoh berikut memungkinkan pengguna untuk membuat, mendeskripsikan, menghapus, dan memperbarui instans DB`sampleDB`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream-influxdb:CreateDbInstance",
                "timestream-influxdb:GetDbInstance",
                "timestream-influxdb:DeleteDbInstance",
                "timestream-influxdb:UpdateDbInstance"
            ],
            "Resource": "arn:aws:timestream-influxdb:us-east-1:<account_ID>:dbinstance/
sampleDB"
        }
    ]
}
```

Memecahkan masalah Amazon Timestream untuk identitas dan akses InfluxDB

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan Timestream untuk InfluxDB dan. IAM

Topik

- Saya tidak berwenang untuk melakukan tindakan di Timestream untuk InfluxDB
- Saya ingin mengizinkan orang di luar AWS akun saya untuk mengakses Timestream saya untuk sumber daya InfluxDB

Saya tidak berwenang untuk melakukan tindakan di Timestream untuk InfluxDB

Jika AWS Management Console memberitahu Anda bahwa Anda tidak berwenang untuk melakukan tindakan, maka Anda harus menghubungi administrator Anda untuk bantuan. Administrator Anda adalah orang yang memberikan nama pengguna dan kata sandi Anda.

Contoh kesalahan berikut terjadi ketika pengguna `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya *`my-example-widget`* fiktif, tetapi tidak memiliki izin `timestream-influxdb:`*`GetWidget`* fiktif.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 timestream-influxdb:GetWidget on resource: my-example-widget
```

Dalam hal ini, Mateo meminta administratornya untuk memperbarui kebijakannya untuk mengizinkan dia mengakses sumber daya *`my-example-widget`* menggunakan tindakan `timestream-influxdb:`*`GetWidget`*.

Saya ingin mengizinkan orang di luar AWS akun saya untuk mengakses Timestream saya untuk sumber daya InfluxDB

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACLs), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa referensi berikut:

- Mengontrol akses ke instans DB dalam a VPC
- Untuk mengetahui apakah Timestream for InfluxDB mendukung fitur-fitur ini, lihat Cara kerja Amazon Timestream for InfluxDB. IAM

- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh AWS akun yang Anda miliki, lihat Menyediakan akses ke IAM pengguna di AWS akun lain yang Anda miliki di Panduan IAM Pengguna.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda ke AWS akun pihak ketiga, lihat Menyediakan akses ke AWS akun yang dimiliki oleh pihak ketiga dalam Panduan IAM Pengguna.
- Untuk mempelajari cara menyediakan akses melalui federasi identitas, lihat Menyediakan akses ke pengguna yang diautentikasi secara eksternal (federasi identitas) di Panduan Pengguna. IAM
- Untuk mempelajari perbedaan antara menggunakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Perbedaan IAM peran dari kebijakan berbasis sumber daya di Panduan Pengguna. IAM

Mengontrol akses ke instans DB dalam a VPC

Menggunakan Amazon Virtual Private Cloud (AmazonVPC), Anda dapat meluncurkan AWS sumber daya, seperti Amazon Timestream untuk instans DB InfluxDB, ke cloud pribadi virtual (). VPC Saat Anda menggunakan AmazonVPC, Anda memiliki kendali atas lingkungan jaringan virtual Anda. Anda dapat memilih rentang alamat IP Anda sendiri, membuat subnet, serta mengonfigurasi perutean dan daftar kontrol akses.

Grup VPC keamanan mengontrol akses ke instans DB di dalam file. VPC Setiap aturan grup VPC keamanan memungkinkan sumber tertentu untuk mengakses instans DB dalam VPC yang terkait dengan grup VPC keamanan tersebut. Sumbernya bisa berupa berbagai alamat (misalnya, 203.0.113.0/24), atau grup keamanan lainnya. VPC Dengan menentukan grup VPC keamanan sebagai sumber, Anda mengizinkan lalu lintas masuk dari semua instance (biasanya server aplikasi) yang menggunakan grup keamanan sumberVPC. Sebelum mencoba terhubung ke instans DB Anda, konfigurasikan VPC untuk kasus penggunaan Anda. Berikut ini adalah skenario umum untuk mengakses instans DB di: VPC

Instans DB yang VPC diakses oleh EC2 instans Amazon dalam hal yang sama VPC

> Penggunaan umum dari instance DB dalam a VPC adalah untuk berbagi data dengan server aplikasi yang berjalan dalam EC2 instance yang samaVPC. EC2Instance mungkin menjalankan server web dengan aplikasi yang berinteraksi dengan instans DB.

Sebuah instans DB dalam VPC diakses oleh sebuah EC2 instance di yang berbeda VPC

Dalam beberapa kasus, instans DB Anda berbeda VPC dari EC2 instance yang Anda gunakan untuk mengaksesnya. Jika demikian, Anda dapat menggunakan VPC peering untuk mengakses instans DB.

Sebuah instans DB dalam VPC diakses oleh aplikasi klien melalui Internet

Untuk mengakses instans DB dalam aplikasi VPC dari klien melalui Internet, Anda mengkonfigurasi VPC dengan subnet publik tunggal dan menggunakan subnet publik untuk membuat instans DB. Anda juga mengkonfigurasi gateway internet di VPC untuk mengaktifkan komunikasi melalui Internet. Untuk terhubung ke instans DB dari luar instansVPC, instans DB harus dapat diakses publik. Selain itu, akses harus diberikan menggunakan aturan masuk grup keamanan instans DB, dan persyaratan lain harus terpenuhi.

Untuk informasi selengkapnya tentang grup VPC keamanan, lihat Grup keamanan di Panduan Pengguna Amazon Virtual Private Cloud.

Untuk detail tentang cara menyambung ke Timestream untuk instans DB InfluxDB, lihat. Menghubungkan ke Amazon Timestream untuk instans DB InfluxDB

Skenario grup keamanan

Penggunaan umum instans DB dalam a VPC adalah untuk berbagi data dengan server aplikasi yang berjalan di EC2 instance Amazon dalam hal yang samaVPC, yang diakses oleh aplikasi klien di luar fileVPC. Untuk skenario ini, Anda menggunakan Timestream untuk InfluxDB dan VPC halaman pada AWS Management Console atau Timestream untuk InfluxDB dan EC2 API operasi untuk membuat instance dan grup keamanan yang diperlukan:

1. Buat grup VPC keamanan (misalnya,`sg-0123ec2example`) dan tentukan aturan masuk yang menggunakan alamat IP aplikasi klien sebagai sumbernya. Grup keamanan ini memungkinkan aplikasi klien Anda untuk terhubung ke EC2 instance di VPC yang menggunakan grup keamanan ini.
2. Buat EC2 instance untuk aplikasi dan tambahkan EC2 instance ke grup VPC keamanan (`sg-0123ec2example`) yang Anda buat di langkah sebelumnya.
3. Buat grup VPC keamanan kedua (misalnya,`sg-6789rdsexample`) dan buat aturan baru dengan menentukan grup VPC keamanan yang Anda buat di langkah 1 (`sg-0123ec2example`) sebagai sumber.

4. Buat instans DB baru dan tambahkan instans DB ke grup VPC keamanan (`sg-6789rdsexample`) yang Anda buat di langkah sebelumnya. Saat Anda membuat DB, gunakan nomor port yang sama dengan yang ditentukan untuk aturan grup VPC keamanan (`sg-6789rdsexample`) yang Anda buat di langkah 3.

Membuat grup VPC keamanan

Anda dapat membuat grup VPC keamanan untuk instans DB dengan menggunakan VPC konsol. Untuk informasi tentang membuat grup keamanan, lihat Grup keamanan di Panduan Pengguna Amazon Virtual Private Cloud.

Mengaitkan grup keamanan dengan instans DB

Anda dapat mengaitkan grup keamanan dengan instans DB dengan menggunakan Update pada konsol Timestream untuk InfluxDB, `UpdateDBInstance` Timestream untuk API InfluxDB, atau perintah. `update-db-instance` AWS CLI

CLIContoh berikut mengaitkan grup VPC keamanan tertentu dan menghapus grup keamanan DB dari instans DB

```
aws timestream-influxdb update-db-instance --identifier dbName --vpc-security-group-
ids sg-ID
```

Untuk informasi tentang cara mengubah instans DB, lihat Memperbarui instans DB.

Menggunakan Peran Tertaut Layanan untuk Amazon Timestream untuk InfluxDB

Amazon Timestream untuk InfluxDB menggunakan AWS Identity and Access Management () IAM peran terkait layanan. Peran terkait layanan adalah jenis peran unik yang ditautkan langsung ke AWS layanan, seperti Amazon Timestream untuk InfluxDB. IAM Amazon Timestream untuk peran terkait layanan InfluxDB telah ditentukan sebelumnya oleh Amazon Timestream untuk InfluxDB. Mereka mencakup semua izin yang diperlukan layanan untuk memanggil AWS layanan atas nama dbinstances Anda.

Peran terkait layanan membuat pengaturan Amazon Timestream untuk InfluxDB lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. Peran sudah ada dalam AWS akun Anda tetapi ditautkan ke Amazon Timestream untuk kasus penggunaan InfluxDB dan memiliki izin yang telah ditentukan sebelumnya. Hanya Amazon Timestream untuk InfluxDB yang dapat mengambil peran ini, dan hanya peran ini yang dapat menggunakan kebijakan izin yang telah ditentukan sebelumnya. Anda dapat menghapus peran tersebut hanya setelah pertama kali

menghapus sumber dayanya yang terkait. Ini melindungi Amazon Timestream untuk sumber daya InfluxDB karena Anda tidak dapat secara tidak sengaja menghapus izin yang diperlukan untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, lihat AWS Layanan yang Bekerja dengan IAM dan cari layanan yang memiliki Ya di kolom Peran Tertaut Layanan. Pilih Ya bersama tautan untuk melihat dokumentasi peran tertaut layanan untuk layanan tersebut.

Daftar Isi

Izin Peran Tertaut Layanan untuk Amazon Timestream untuk InfluxDB

Amazon TimeStream untuk InfluxDB menggunakan peran terkait layanan bernama AmazonTimestreamInfluxDBServiceRolePolicy— Kebijakan ini memungkinkan Timestream untuk InfluxDB mengelola AWS sumber daya atas nama Anda sebagaimana diperlukan untuk mengelola klaster Anda.

Kebijakan izin peran AmazonTimestreamInflux DBServiceRolePolicy terkait layanan memungkinkan Amazon TimeStream untuk InfluxDB menyelesaikan tindakan berikut pada sumber daya yang ditentukan:

```
{
 "Version": "2012-10-17",
 "Statement": [
```

```
{
 "Sid": "DescribeNetworkStatement",
 "Effect": "Allow",
 "Action": [
  "ec2:DescribeSubnets",
  "ec2:DescribeVpcs",
  "ec2:DescribeNetworkInterfaces"
 ],
 "Resource": "*"
},
{
 "Sid": "CreateEniInSubnetStatement",
 "Effect": "Allow",
 "Action": [
  "ec2:CreateNetworkInterface"
 ],
 "Resource": [
  "arn:aws:ec2:*:*:subnet/*",
  "arn:aws:ec2:*:*:security-group/*"
 ]
},
{
 "Sid": "CreateEniStatement",
 "Effect": "Allow",
 "Action": [
  "ec2:CreateNetworkInterface"
 ],
 "Resource": "arn:aws:ec2:*:*:network-interface/*",
 "Condition": {
  "Null": {
   "aws:RequestTag/AmazonTimestreamInfluxDBManaged": "false"
  }
 }
},
{
 "Sid": "CreateTagWithEniStatement",
 "Effect": "Allow",
 "Action": [
  "ec2:CreateTags"
 ],
 "Resource": "arn:aws:ec2:*:*:network-interface/*",
 "Condition": {
  "Null": {
   "aws:RequestTag/AmazonTimestreamInfluxDBManaged": "false"
```

```
    },
    "StringEquals": {
     "ec2:CreateAction": [
      "CreateNetworkInterface"
     ]
    }
   }
  },
  {
   "Sid": "ManageEniStatement",
   "Effect": "Allow",
   "Action": [
    "ec2:CreateNetworkInterfacePermission",
    "ec2:DeleteNetworkInterface"
   ],
   "Resource": "arn:aws:ec2:*:*:network-interface/*",
   "Condition": {
    "Null": {
     "aws:ResourceTag/AmazonTimestreamInfluxDBManaged": "false"
    }
   }
  },
  {
   "Sid": "PutCloudWatchMetricsStatement",
   "Effect": "Allow",
   "Action": [
    "cloudwatch:PutMetricData"
   ],
   "Condition": {
    "StringEquals": {
     "cloudwatch:namespace": [
      "AWS/Timestream/InfluxDB",
      "AWS/Usage"
     ]
    }
   },
   "Resource": [
    "*"
   ]
  },
  {
   "Sid": "ManageSecretStatement",
   "Effect": "Allow",
   "Action": [
```

```
    "secretsmanager:CreateSecret",
    "secretsmanager:DeleteSecret"
   ],
   "Resource": [
    "arn:aws:secretsmanager:*:*:secret:READONLY-InfluxDB-auth-parameters-*"
   ],
   "Condition": {
    "StringEquals": {
     "aws:ResourceAccount": "${aws:PrincipalAccount}"
    }
   }
  }
 ]
}
```

Untuk mengizinkan IAM entitas membuat peran AmazonTimestreamInflux DBServiceRolePolicy terkait layanan

Tambahkan pernyataan kebijakan berikut ke izin untuk IAM entitas tersebut:

```
{
    "Effect": "Allow",
    "Action": [
        "iam:CreateServiceLinkedRole",
        "iam:PutRolePolicy"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/
timestreamforinfluxdb.amazonaws.com/AmazonTimestreamInfluxDBServiceRolePolicy*",
    "Condition": {"StringLike": {"iam:AWS ServiceName":
 "timestreamforinfluxdb.amazonaws.com"}}
}
```

Untuk mengizinkan IAM entitas menghapus peran AmazonTimestreamInflux DBServiceRolePolicy terkait layanan

Tambahkan pernyataan kebijakan berikut ke izin untuk IAM entitas tersebut:

```
{
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
```

```
    "Resource": "arn:aws:iam::*:role/aws-service-role/
timestreamforinfluxdb.amazonaws.com/AmazonTimestreamInfluxDBServiceRolePolicy*",
    "Condition": {"StringLike": {"iam:AWS ServiceName":
 "timestreamforinfluxdb.amazonaws.com"}}
}
```

Atau, Anda dapat menggunakan kebijakan AWS terkelola untuk menyediakan akses penuh ke Amazon Timestream untuk InfluxDB.

Membuat Peran Tertaut Layanan (IAM)

Anda tidak perlu membuat peran terkait layanan secara manual. Saat Anda membuat instans DB, Amazon Timestream untuk InfluxDB membuat 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 membuat instans DB, Amazon Timestream untuk InfluxDB membuat peran terkait layanan untuk Anda lagi.

Mengedit Deskripsi Peran Tertaut Layanan untuk Amazon Timestream untuk InfluxDB

Amazon Timestream untuk InfluxDB tidak memungkinkan Anda mengedit peran terkait layanan. AmazonTimestreamInflux DBServiceRolePolicy Setelah Anda membuat peran terkait layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin mereferensikan peran tersebut. Namun, Anda dapat mengedit deskripsi peran menggunakanIAM.

Mengedit Deskripsi Peran Tertaut Layanan (Konsol) IAM

Anda dapat menggunakan IAM konsol untuk mengedit deskripsi peran terkait layanan.

Untuk mengedit deskripsi peran terkait layanan (konsol)

1. Di panel navigasi kiri IAM konsol, pilih Peran.
2. Pilih nama peran yang akan diubah.
3. Di ujung kanan Deskripsi peran, pilih Edit.
4. Masukkan deskripsi baru di kotak, lalu pilih Simpan.

Mengedit Deskripsi Peran Tertaut Layanan () IAM CLI

Anda dapat menggunakan IAM operasi dari AWS Command Line Interface untuk mengedit deskripsi peran terkait layanan.

Untuk mengubah deskripsi peran terkait layanan () CLI

1. (Opsional) Untuk melihat deskripsi saat ini untuk peran, gunakan AWS CLI untuk IAM operasi`get-role`.

   Example

   ```
   $ aws iam get-role --role-name AmazonTimestreamInfluxDBServiceRolePolicy
   ```

   Gunakan nama peran, bukanARN, untuk merujuk ke peran dengan CLI operasi. Misalnya, jika peran memiliki yang berikutARN:`arn:aws:iam::123456789012:role/myrole`, lihat peran sebagai**`myrole`**.
2. Untuk memperbarui deskripsi peran terkait layanan, gunakan AWS CLI for IAM operation. `update-role-description`

   Linux dan macOS

   ```
   $ aws iam update-role-description \
       --role-name AmazonTimestreamInfluxDBServiceRolePolicy \
       --description "new description"
   ```

   Windows

   ```
   $ aws iam update-role-description ^
       --role-name AmazonTimestreamInfluxDBServiceRolePolicy ^
       --description "new description"
   ```

Mengedit Deskripsi Peran Tertaut Layanan () IAM API

Anda dapat menggunakan IAM API untuk mengedit deskripsi peran terkait layanan.

Untuk mengubah deskripsi peran terkait layanan () API

1. (Opsional) Untuk melihat deskripsi saat ini untuk peran, gunakan IAM API operasi GetRole.

   Example

   ```
   https://iam.amazonaws.com/
      ?Action=GetRole
   ```

```
   &RoleName=AmazonTimestreamInfluxDBServiceRolePolicy
   &Version=2010-05-08
   &AUTHPARAMS
```

2. Untuk memperbarui deskripsi peran, gunakan IAM API operasi UpdateRoleDescription.

   Example

```
https://iam.amazonaws.com/
   ?Action=UpdateRoleDescription
   &RoleName=AmazonTimestreamInfluxDBServiceRolePolicy
   &Version=2010-05-08
   &Description="New description"
```

Menghapus Peran Tertaut Layanan untuk Amazon Timestream untuk InfluxDB

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, kami merekomendasikan Anda menghapus peran tersebut. Dengan begitu, Anda tidak perlu lagi memantau atau memelihara entitas yang tidak digunakan. Namun, Anda harus membersihkan peran terkait layanan sebelum dapat menghapusnya.

Amazon Timestream untuk InfluxDB tidak menghapus peran terkait layanan untuk Anda.

Membersihkan Peran Terkait Layanan

Sebelum Anda dapat menggunakan IAM untuk menghapus peran terkait layanan, konfirmasikan terlebih dahulu bahwa peran tersebut tidak memiliki sumber daya (cluster) yang terkait dengannya.

Untuk memeriksa apakah peran yang ditautkan layanan memiliki sesi aktif di konsol IAM

1. Masuk ke AWS Management Console dan buka IAM konsol di https://console.aws.amazon.com/iam/.
2. Di panel navigasi kiri IAM konsol, pilih Peran. Kemudian pilih nama (bukan kotak centang) AmazonTimestreamInflux DBServiceRolePolicy peran.
3. Di halaman Ringkasan untuk peran yang dipilih, pilih tab Penasihat Akses.
4. Di tab Penasihat Akses, tinjau aktivitas terbaru untuk peran terkait layanan tersebut.

Menghapus Peran Tertaut Layanan (Konsol) IAM

Anda dapat menggunakan IAM konsol untuk menghapus peran terkait layanan.

Untuk menghapus peran terkait layanan (konsol)

1. Masuk ke AWS Management Console dan buka IAM konsol di https://console.aws.amazon.com/iam/.
2. Di panel navigasi kiri IAM konsol, pilih Peran. Kemudian, pilih kotak centang di sebelah nama peran yang ingin dihapus, bukan nama atau baris itu sendiri.
3. Untuk Tindakan peran di bagian atas halaman, pilih Hapus peran.
4. Di halaman konfirmasi, tinjau data layanan yang terakhir diakses, yang menunjukkan kapan masing-masing peran yang dipilih terakhir mengakses AWS layanan. Hal ini membantu Anda mengonfirmasi aktif tidaknya peran tersebut saat ini. Jika Anda ingin melanjutkan, pilih Ya, Hapus guna mengirimkan peran terkait layanan untuk penghapusan.
5. Tonton notifikasi IAM konsol untuk memantau kemajuan penghapusan peran terkait layanan. Karena penghapusan peran IAM terkait layanan bersifat asinkron, setelah Anda mengirimkan peran untuk dihapus, tugas penghapusan dapat berhasil atau gagal. Jika tugas tersebut gagal, Anda dapat memilih Lihat detail atau Lihat Sumber Daya dari notifikasi untuk mempelajari alasan gagalnya penghapusan.

Menghapus Peran Tertaut Layanan () IAM CLI

Anda dapat menggunakan IAM operasi dari AWS Command Line Interface untuk menghapus peran terkait layanan.

Untuk menghapus peran terkait layanan () CLI

1. Jika Anda tidak tahu nama peran terkait layanan yang ingin dihapus, masukkan perintah berikut. Perintah ini mencantumkan peran dan Nama Sumber Daya Amazon mereka (ARNs) di akun Anda.

   ```
   $ aws iam get-role --role-name role-name
   ```

   Gunakan nama peran, bukanARN, untuk merujuk ke peran dengan CLI operasi. Misalnya, jika peran memiliki ARN`arn:aws:iam::123456789012:role/myrole`, Anda merujuk ke peran sebagai**`myrole`**.
2. 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 kondisi ini tidak terpenuhi. Anda harus menangkap `deletion-task-id` dari

tanggapan untuk memeriksa status tugas penghapusan. Masukkan perintah berikut untuk mengirimkan permintaan penghapusan peran terkait layanan.

```
$ aws iam delete-service-linked-role --role-name role-name
```

3. Masukkan perintah berikut untuk memeriksa status tugas penghapusan.

```
$ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
```

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.

Menghapus Peran Tertaut Layanan () IAM API

Anda dapat menggunakan IAM API untuk menghapus peran terkait layanan.

Untuk menghapus peran terkait layanan () API

1. Untuk mengirimkan permintaan penghapusan untuk roll terkait layanan, hubungi DeleteServiceLinkedRole. Dalam permintaan, tentukan nama peran.

   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 kondisi ini tidak terpenuhi. Anda harus menangkap `DeletionTaskId` dari tanggapan untuk memeriksa status tugas penghapusan.
2. Untuk memeriksa status penghapusan, hubungi GetServiceLinkedRoleDeletionStatus. Dalam permintaan, tentukan`DeletionTaskId`.

   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.

Wilayah yang Didukung untuk Amazon Timestream untuk Peran Tertaut Layanan InfluxDB

Amazon Timestream untuk InfluxDB mendukung penggunaan peran terkait layanan di semua Wilayah tempat layanan tersedia. Untuk informasi selengkapnya, lihat AWS titik akhir layanan.

AWS kebijakan terkelola untuk Amazon Timestream untuk InfluxDB

Untuk menambahkan izin ke pengguna, grup, dan peran, lebih mudah menggunakan kebijakan AWS terkelola daripada menulis kebijakan sendiri. Butuh waktu dan keahlian untuk membuat kebijakan terkelola IAM pelanggan yang hanya memberi tim Anda izin yang mereka butuhkan. 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 AWS AWS terkelola, lihat kebijakan terkelola di Panduan IAM Pengguna.

AWS layanan memelihara dan memperbarui kebijakan AWS terkelola. Anda tidak dapat mengubah izin dalam kebijakan AWS terkelola. 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 dari kebijakan AWS terkelola, sehingga pembaruan kebijakan tidak akan merusak izin yang ada.

Selain itu, AWS mendukung kebijakan terkelola untuk fungsi pekerjaan yang mencakup beberapa layanan. Misalnya, kebijakan ReadOnlyAccess AWS terkelola menyediakan akses hanya-baca ke semua AWS layanan dan sumber daya. Saat layanan meluncurkan fitur baru, AWS tambahkan izin hanya-baca untuk operasi dan sumber daya baru. Untuk daftar dan deskripsi kebijakan fungsi pekerjaan, lihat kebijakan AWS terkelola untuk fungsi pekerjaan di Panduan IAM Pengguna.

AWS kebijakan terkelola: AmazonTimestreamInflux DBServiceRolePolicy

Anda tidak dapat melampirkan kebijakan AmazonTimestreamInflux DBServiceRolePolicy AWS terkelola ke identitas di akun Anda. Kebijakan ini merupakan bagian dari peran terkait layanan AWS TimestreamforInflux DB. Peran ini memungkinkan layanan untuk mengelola antarmuka jaringan dan grup keamanan di akun Anda.

Timestream untuk InfluxDB menggunakan izin dalam kebijakan ini untuk mengelola grup EC2 keamanan dan antarmuka jaringan. Ini diperlukan untuk mengelola Timestream untuk instans DB InfluxDB.

Untuk meninjau kebijakan ini dalam JSON format, lihat AmazonTimestreamInfluxDBServiceRolePolicy.

AWS-kebijakan terkelola untuk Amazon Timestream untuk InfluxDB

AWS mengatasi banyak kasus penggunaan umum dengan menyediakan IAM kebijakan mandiri yang dibuat dan dikelola oleh AWS. Kebijakan terkelola ini memberikan izin yang diperlukan untuk kasus penggunaan umum sehingga Anda tidak perlu menyelidiki izin apa yang diperlukan. Untuk informasi selengkapnya, lihat Kebijakan AWS Terkelola di Panduan IAM Pengguna.

Kebijakan AWS terkelola berikut, yang dapat Anda lampirkan ke pengguna di akun Anda, khusus untuk Timestream untuk InfluxDB:

AmazonTimestreamInfluxDBFullAccess

Anda dapat melampirkan `AmazonTimestreamInfluxDBFullAccess` kebijakan ke IAM identitas Anda. Kebijakan ini memberikan izin administratif yang memungkinkan akses penuh ke semua Timestream untuk sumber daya InfluxDB.

Anda juga dapat membuat IAM kebijakan kustom Anda sendiri untuk mengizinkan izin untuk Amazon Timestream untuk API tindakan InfluxDB. Anda dapat melampirkan kebijakan khusus ini ke IAM pengguna atau grup yang memerlukan izin tersebut.

Untuk meninjau kebijakan ini dalam JSON format, lihat AmazonTimestreamInfluxDBFullAccess.

Timestream untuk pembaruan InfluxDB ke kebijakan terkelola AWS

Lihat detail tentang pembaruan kebijakan AWS terkelola untuk Timestream untuk InfluxDB sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan RSS umpan di halaman riwayat Dokumen Timestream untuk InfluxDB.

Perubahan	Deskripsi	Tanggal
AmazonTimestreamInfluxDBFullAccess —	Menambahkan `ec2:DescribeRouteTables` tindakan ke kebijakan `AmazonTim`	10/08/2024

Perubahan	Deskripsi	Tanggal
Permbaruan ke kebijakan yang sudah ada	`estreamInfluxDBFul lAccess` terkelola yang ada. Tindakan ini digunakan untuk mendeskripsikan tabel rute Anda	
AWS kebijakan terkelola : AmazonTimestreamIn flux DBServiceRolePolicy – Kebijakan baru	Amazon Timestream untuk InfluxDB menambahk an kebijakan baru yang memungkinkan layanan mengelola antarmuka jaringan dan grup keamanan di akun Anda.	03/14/2024
AmazonTimestreamIn fluxDBFullAccess – Kebijakan baru	Amazon TimeStream untuk InfluxDB menambahk an kebijakan baru untuk menyediakan akses administr atif penuh untuk membuat, memperbarui, menghapus , dan mencantumkan instans Amazon TimeStrea m InfluxDB serta membuat serta mencantumkan grup parameter.	03/14/2024

Menghubungkan ke Timestream untuk InfluxDB melalui titik akhir VPC

Anda dapat terhubung langsung ke Timestream untuk InfluxDB melalui titik akhir antarmuka pribadi di cloud pribadi virtual Anda (). VPC Saat Anda menggunakan VPC titik akhir antarmuka, komunikasi antara Anda VPC dan Timestream untuk InfluxDB dilakukan sepenuhnya di dalam jaringan. AWS

Timestream untuk InfluxDB mendukung titik akhir Amazon Virtual Private Cloud (AmazonVPC) yang didukung oleh. AWS PrivateLink Setiap VPC endpoint diwakili oleh satu atau lebih Elastic Network Interfaces (ENIs) dengan alamat IP pribadi di subnet AndaVPC.

VPCTitik akhir antarmuka menghubungkan Anda VPC langsung ke Timestream untuk InfluxDB tanpa gateway internet, NAT perangkat, koneksi, atau VPN koneksi. AWS Direct Connect Instans di Anda VPC tidak memerlukan alamat IP publik untuk berkomunikasi dengan Timestream untuk InfluxDB.

Wilayah

Timestream untuk InfluxDB mendukung kebijakan VPC titik akhir dan VPC titik akhir Wilayah AWS di mana Timestream untuk InfluxDB didukung.

Topik

- Pertimbangan Timestream untuk titik akhir InfluxDB VPC
- Membuat VPC titik akhir untuk Timestream untuk InfluxDB
- Menghubungkan ke Timestream untuk titik akhir VPC InfluxDB
- Mengontrol akses ke titik VPC akhir
- Menggunakan VPC titik akhir dalam pernyataan kebijakan
- Mencatat titik VPC akhir Anda

Pertimbangan Timestream untuk titik akhir InfluxDB VPC

Sebelum Anda menyiapkan VPC titik akhir antarmuka untuk Timestream untuk InfluxDB, tinjau topik properti dan batasan titik akhir Antarmuka di Panduan.AWS PrivateLink

Timestream untuk dukungan InfluxDB untuk VPC titik akhir mencakup yang berikut ini.

- Anda dapat menggunakan VPC endpoint Anda untuk memanggil semua Timestream untuk operasi InfluxDB API dari Anda. VPC
- Anda dapat menggunakan AWS CloudTrail log untuk mengaudit penggunaan Timestream untuk sumber daya InfluxDB melalui titik akhir. VPC Untuk detailnya, lihat Mencatat titik VPC akhir Anda.

Membuat VPC titik akhir untuk Timestream untuk InfluxDB

Anda dapat membuat VPC titik akhir untuk Timestream untuk InfluxDB dengan menggunakan konsol Amazon VPC atau Amazon. VPC API Untuk informasi selengkapnya, lihat Membuat titik akhir antarmuka di AWS PrivateLink Panduan.

- Untuk membuat VPC endpoint Timestream untuk InfluxDB, gunakan nama layanan berikut:

```
com.amazonaws.region.timestream-influxdb
```

Sebagai contoh, di Wilayah US West (Oregon) (`us-west-2`), nama layanan akan menjadi:

```
com.amazonaws.us-west-2.timestream-influxdb
```

Untuk mempermudah penggunaan VPC titik akhir, Anda dapat mengaktifkan DNSnama pribadi untuk titik VPC akhir Anda. Jika Anda memilih opsi Aktifkan DNS Nama, Timestream standar untuk DNS nama host InfluxDB menyelesaikan ke titik akhir Anda. VPC Misalnya, `https://timestream-influxdb.us-west-2.amazonaws.com` akan menyelesaikan ke VPC titik akhir yang terhubung ke nama `com.amazonaws.us-west-2.timestream-influxdb` layanan.

Opsi ini membuatnya lebih mudah untuk menggunakan VPC titik akhir. Itu AWS SDKs dan AWS CLI gunakan Timestream standar untuk DNS nama host InfluxDB secara default, sehingga Anda tidak perlu menentukan VPC titik akhir dalam aplikasi dan perintahURL.

Untuk informasi selengkapnya, lihat Mengakses layanan melalui titik akhir antarmuka di Panduan.AWS PrivateLink

Menghubungkan ke Timestream untuk titik akhir VPC InfluxDB

Anda dapat terhubung ke Timestream untuk InfluxDB melalui VPC titik akhir dengan menggunakan, atau AWS SDK. AWS CLI AWS Tools for PowerShell Untuk menentukan VPC titik akhir, gunakan DNS namanya.

Jika Anda mengaktifkan nama host pribadi saat membuat VPC titik akhir, Anda tidak perlu menentukan VPC titik akhir URL dalam CLI perintah atau konfigurasi aplikasi Anda. Timestream standar untuk DNS nama host InfluxDB diselesaikan ke titik akhir Anda. VPC SDKsGunakan AWS CLI dan gunakan nama host ini secara default, sehingga Anda dapat mulai menggunakan VPC titik akhir untuk terhubung ke Timestream untuk titik akhir regional InfluxDB tanpa mengubah apa pun di skrip dan aplikasi Anda.

Untuk menggunakan nama host pribadi, `enableDnsSupport` atribut `enableDnsHostnames` dan atribut Anda VPC harus disetel ke`true`. Untuk mengatur atribut ini, gunakan ModifyVpcAttributeoperasi. Untuk detailnya, lihat Melihat dan memperbarui DNS atribut untuk Anda VPC di Panduan VPC Pengguna Amazon.

Mengontrol akses ke titik VPC akhir

Untuk mengontrol akses ke VPC titik akhir Timestream untuk InfluxDB, lampirkan kebijakan titik akhir ke VPCtitik akhir Anda. VPC Kebijakan endpoint menentukan apakah prinsipal dapat menggunakan VPC endpoint untuk memanggil Timestream untuk operasi InfluxDB di Timestream untuk sumber daya InfluxDB.

Anda dapat membuat kebijakan VPC titik akhir saat membuat titik akhir, dan Anda dapat mengubah kebijakan VPC titik akhir kapan saja. Gunakan konsol VPC manajemen, atau ModifyVpcEndpointoperasi CreateVpcEndpointatau. Anda juga dapat membuat dan mengubah kebijakan VPC endpoint dengan menggunakan AWS CloudFormation template. Untuk bantuan menggunakan konsol VPC manajemen, lihat Membuat titik akhir antarmuka dan Memodifikasi titik akhir antarmuka dalam Panduan.AWS PrivateLink

> **Note**
>
> Timestream untuk InfluxDB mendukung kebijakan VPC titik akhir yang dimulai pada Juli 2020. VPCendpoint untuk Timestream untuk InfluxDB yang dibuat sebelum tanggal tersebut memiliki kebijakan VPC endpoint default, tetapi Anda dapat mengubahnya kapan saja.

Topik

- Tentang VPC kebijakan endpoint
- Kebijakan VPC titik akhir default
- Membuat kebijakan VPC endpoint
- Melihat kebijakan VPC titik akhir

Tentang VPC kebijakan endpoint

Agar permintaan Timestream untuk InfluxDB yang menggunakan VPC titik akhir berhasil, prinsipal memerlukan izin dari dua sumber:

- IAMKebijakan harus memberikan izin utama untuk memanggil operasi pada sumber daya.
- Kebijakan VPC endpoint harus memberikan izin utama untuk menggunakan endpoint untuk membuat permintaan.

Kebijakan VPC titik akhir default

Setiap VPC titik akhir memiliki kebijakan VPC titik akhir, tetapi Anda tidak diharuskan untuk menentukan kebijakan tersebut. Jika Anda tidak menentukan kebijakan, kebijakan titik akhir default memungkinkan semua operasi oleh semua prinsipal di semua sumber daya pada titik akhir.

Namun, untuk Timestream untuk sumber daya InfluxDB, prinsipal juga harus memiliki izin untuk memanggil operasi dari IAMkebijakan. Oleh karena itu, dalam praktiknya, kebijakan default mengatakan bahwa jika kepala sekolah memiliki izin untuk memanggil operasi pada sumber daya, mereka juga dapat memanggilnya dengan menggunakan titik akhir.

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Principal": "*",
      "Resource": "*"
    }
  ]
}
```

Untuk mengizinkan prinsipal menggunakan VPC titik akhir hanya untuk sebagian dari operasi yang diizinkan, buat atau perbarui kebijakan titik akhir. VPC

Membuat kebijakan VPC endpoint

Kebijakan VPC endpoint menentukan apakah prinsipal memiliki izin untuk menggunakan VPC titik akhir untuk melakukan operasi pada sumber daya. Untuk Timestream untuk sumber daya InfluxDB, prinsipal juga harus memiliki izin untuk melakukan operasi dari kebijakan, IAM

Setiap pernyataan kebijakan VPC endpoint membutuhkan elemen-elemen berikut:

- Prinsip-prinsip yang dapat melakukan tindakan.
- Tindakan yang dapat dilakukan
- Sumber daya yang padanya tindakan dapat dilakukan

Pernyataan kebijakan tidak menentukan VPC titik akhir. Sebaliknya, ini berlaku untuk VPC titik akhir apa pun yang dilampirkan kebijakan tersebut. Untuk informasi selengkapnya, lihat Mengontrol akses ke layanan dengan VPC titik akhir di Panduan VPC Pengguna Amazon.

AWS CloudTrail mencatat semua operasi yang menggunakan VPC titik akhir.

Melihat kebijakan VPC titik akhir

Untuk melihat kebijakan VPC titik akhir untuk titik akhir, gunakan konsol VPC manajemen atau operasi. DescribeVpcEndpoints

AWS CLI Perintah berikut mendapatkan kebijakan untuk titik akhir dengan ID VPC titik akhir yang ditentukan.

Sebelum menggunakan perintah ini, ganti ID titik akhir contoh dengan yang valid dari akun Anda.

```
$ aws ec2 describe-vpc-endpoints \
--query 'VpcEndpoints[?VpcEndpointId==`vpc-endpoint-id`].[PolicyDocument]'
--output text
```

Menggunakan VPC titik akhir dalam pernyataan kebijakan

Anda dapat mengontrol akses ke Timestream untuk sumber daya dan operasi InfluxDB saat permintaan berasal dari VPC atau menggunakan titik akhir. VPC Untuk melakukannya, gunakan salah satu kunci kondisi global berikut dalam IAMkebijakan.

- Gunakan tombol `aws:sourceVpce` kondisi untuk memberikan atau membatasi akses berdasarkan titik VPC akhir.
- Gunakan kunci `aws:sourceVpc` kondisi untuk memberikan atau membatasi akses berdasarkan VPC yang menghosting titik akhir pribadi.

> **Note**
>
> Berhati-hatilah saat membuat kebijakan dan IAM kebijakan utama berdasarkan VPC titik akhir Anda. Jika pernyataan kebijakan mengharuskan permintaan berasal dari VPC titik akhir VPC atau tertentu, permintaan dari AWS layanan terintegrasi yang menggunakan Timestream untuk sumber daya InfluxDB atas nama Anda mungkin gagal.
> Juga, kunci `aws:sourceIP` kondisi tidak efektif ketika permintaan berasal dari VPCtitik akhir Amazon. Untuk membatasi permintaan ke VPC titik akhir, gunakan kunci `aws:sourceVpce` atau `aws:sourceVpc` kondisi. Untuk informasi selengkapnya, lihat Identitas dan manajemen akses untuk layanan VPC titik VPC akhir dan titik akhir di Panduan.AWS PrivateLink

Anda dapat menggunakan kunci kondisi global ini untuk mengontrol akses ke operasi seperti CreateDbInstanceitu tidak bergantung pada sumber daya tertentu.

Mencatat titik VPC akhir Anda

AWS CloudTrail mencatat semua operasi yang menggunakan VPC titik akhir. Ketika permintaan ke Timestream untuk InfluxDB menggunakan VPC titik akhir, ID VPC titik akhir akan muncul di entri AWS CloudTrail log yang mencatat permintaan tersebut. Anda dapat menggunakan ID endpoint untuk mengaudit penggunaan Timestream Anda untuk titik akhir VPC InfluxDB.

Namun, CloudTrail log Anda tidak menyertakan operasi yang diminta oleh prinsipal di akun lain atau permintaan Timestream untuk operasi InfluxDB di Timestream untuk sumber daya InfluxDB dan alias di akun lain. Selain itu, untuk melindungi AndaVPC, permintaan yang ditolak oleh kebijakan VPC titik akhir, tetapi sebaliknya diizinkan, tidak dicatat. AWS CloudTrail

Pencatatan dan pemantauan di Timestream untuk InfluxDB

Pemantauan adalah bagian penting dalam menjaga keandalan, ketersediaan, dan kinerja Timestream untuk InfluxDB dan solusi Anda. AWS Anda harus mengumpulkan data pemantauan dari semua bagian AWS solusi Anda sehingga Anda dapat lebih mudah men-debug kegagalan multi-titik jika terjadi. Namun, sebelum Anda mulai memantau Timestream untuk InfluxDB, Anda harus membuat rencana pemantauan yang mencakup jawaban atas pertanyaan-pertanyaan berikut:

- Apa tujuan pemantauan Anda?
- Sumber daya apa yang akan Anda pantau?
- Seberapa sering Anda akan memantau sumber daya ini?
- Alat pemantauan apa yang akan Anda gunakan?
- Siapa yang akan melakukan tugas pemantauan?
- Siapa yang harus diberi tahu saat terjadi kesalahan?

Langkah selanjutnya adalah menetapkan garis dasar untuk TimeStream normal untuk kinerja InfluxDB di lingkungan Anda, dengan mengukur kinerja pada berbagai waktu dan dalam kondisi beban yang berbeda. Saat Anda memantau Timestream untuk InfluxDB, simpan data pemantauan historis sehingga Anda dapat membandingkannya dengan data kinerja saat ini, mengidentifikasi pola kinerja normal dan anomali kinerja, dan merancang metode untuk mengatasi masalah.

Untuk menetapkan baseline, Anda harus, setidaknya, memantau item-item berikut:

- Kesalahan sistem, sehingga Anda dapat menentukan apakah ada permintaan yang mengakibatkan kesalahan.

Topik

Alat pemantauan

AWS menyediakan berbagai alat yang dapat Anda gunakan untuk memantau Timestream untuk InfluxDB. Anda dapat mengonfigurasi beberapa alat ini untuk melakukan pemantauan untuk Anda, sementara beberapa alat memerlukan intervensi manual. Kami menyarankan agar Anda mengautomasi tugas pemantauan sebanyak mungkin.

Topik

Alat pemantauan otomatis

Anda dapat menggunakan alat pemantauan otomatis berikut untuk menonton Timestream untuk InfluxDB dan melaporkan ketika ada sesuatu yang salah:

- CloudWatch Alarm Amazon — Tonton satu metrik selama periode waktu yang Anda tentukan, dan lakukan satu atau beberapa tindakan berdasarkan nilai metrik relatif terhadap ambang batas tertentu selama beberapa periode waktu. Tindakannya adalah pemberitahuan yang dikirim ke topik Amazon Simple Notification Service (AmazonSNS) atau kebijakan Amazon EC2 Auto Scaling. CloudWatch alarm tidak memanggil tindakan hanya karena mereka berada dalam keadaan tertentu; negara harus telah berubah dan dipertahankan untuk sejumlah periode tertentu. Untuk informasi selengkapnya, lihat Pemantauan CloudWatch dengan Amazon.

Alat pemantauan manual

Bagian penting lainnya dari pemantauan Timestream untuk InfluxDB melibatkan pemantauan secara manual item yang tidak tercakup oleh CloudWatch alarm. Timestream untuk InfluxDB CloudWatch,

Trusted Advisor,, dan AWS Management Console dasbor lainnya memberikan at-a-glance tampilan status lingkungan Anda. AWS

- CloudWatch Halaman beranda menunjukkan yang berikut:
  - Alarm dan status saat ini
  - Grafik alarm dan sumber daya
  - Status kesehatan layanan

  Selain itu, Anda dapat menggunakan CloudWatch untuk melakukan hal berikut:
  - Membuat dasbor yang disesuaikan untuk memantau layanan yang penting bagi Anda
  - Data metrik grafik untuk memecahkan masalah dan mengungkap tren
  - Cari dan telusuri semua metrik AWS sumber daya Anda
  - Membuat dan mengedit alarm untuk menerima notifikasi terkait masalah

Logging Timestream untuk panggilan API InfluxDB dengan AWS CloudTrail

Timestream untuk InfluxDB terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di Timestream untuk InfluxDB. CloudTrail menangkap API panggilan Data Definition Language (DDL) untuk Timestream untuk InfluxDB sebagai peristiwa. Panggilan yang ditangkap termasuk panggilan dari Timestream untuk konsol InfluxDB dan panggilan kode ke Timestream untuk operasi InfluxDB. API Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail peristiwa secara terus menerus ke bucket Amazon Simple Storage Service (Amazon S3), termasuk peristiwa untuk Timestream untuk InfluxDB. 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 ke Timestream untuk InfluxDB, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari selengkapnya CloudTrail, lihat Panduan AWS CloudTrail Pengguna.

Timestream untuk informasi InfluxDB di CloudTrail

CloudTrail diaktifkan di AWS akun Anda saat Anda membuat akun. Ketika aktivitas terjadi di Timestream untuk InfluxDB, aktivitas tersebut direkam dalam suatu CloudTrail peristiwa bersama dengan peristiwa AWS layanan lainnya dalam riwayat Acara. Anda dapat melihat, mencari, dan mengunduh peristiwa terbaru di akun AWS . Untuk informasi selengkapnya, lihat Melihat Acara dengan Riwayat CloudTrail Acara.

Untuk catatan peristiwa yang sedang berlangsung di AWS akun Anda, termasuk acara untuk Timestream untuk InfluxDB, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon 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 Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log.

Untuk informasi selengkapnya, lihat topik berikut di Panduan Pengguna AWS CloudTrail :

- Gambaran Umum untuk Membuat Jejak
- CloudTrail Layanan dan Integrasi yang Didukung
- Mengkonfigurasi SNS Pemberitahuan Amazon untuk CloudTrail
- Menerima File CloudTrail Log dari Beberapa Wilayah
- Menerima File CloudTrail Log dari Beberapa Akun
- Pencatatan peristiwa data

Setiap entri peristiwa atau log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan berikut ini:

- Apakah permintaan dibuat dengan root atau AWS Identity and Access Management (IAM) kredensi pengguna
- Baik permintaan tersebut dibuat dengan kredensial keamanan sementara untuk peran atau pengguna gabungan
- Apakah permintaan itu dibuat oleh AWS layanan lain

Untuk informasi lebih lanjut, lihat CloudTrail userIdentityElemen.

Validasi kepatuhan untuk Amazon Timestream untuk InfluxDB

Auditor pihak ketiga menilai keamanan dan kepatuhan Amazon Timestream untuk InfluxDB sebagai bagian dari beberapa program kepatuhan. AWS Sumber daya yang dimaksud meliputi:

- GDPR
- HIPAA
- PCI

- SOC

Untuk mempelajari apakah an Layanan AWS berada dalam lingkup program kepatuhan tertentu, lihat Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS dan pilih program kepatuhan yang Anda minati. Untuk informasi umum, lihat Program AWS Kepatuhan Program AWS .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifact .

Tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- Panduan Memulai Cepat Keamanan dan Kepatuhan — Panduan penerapan ini membahas pertimbangan arsitektur dan memberikan langkah-langkah untuk menerapkan lingkungan dasar AWS yang berfokus pada keamanan dan kepatuhan.
- Arsitektur untuk HIPAA Keamanan dan Kepatuhan di Amazon Web Services — Whitepaper ini menjelaskan bagaimana perusahaan dapat menggunakan AWS untuk membuat HIPAA aplikasi yang memenuhi syarat.

  > **Note**
  >
  > Tidak semua Layanan AWS HIPAA memenuhi syarat. Untuk informasi selengkapnya, lihat Referensi Layanan yang HIPAA Memenuhi Syarat.

- AWS Sumber Daya AWS — Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- AWS Panduan Kepatuhan Pelanggan - Memahami model tanggung jawab bersama melalui lensa kepatuhan. Panduan ini merangkum praktik terbaik untuk mengamankan Layanan AWS dan memetakan panduan untuk kontrol keamanan di berbagai kerangka kerja (termasuk Institut Standar dan Teknologi Nasional (NIST), Dewan Standar Keamanan Industri Kartu Pembayaran (PCI), dan Organisasi Internasional untuk Standardisasi ()). ISO
- Mengevaluasi Sumber Daya dengan Aturan dalam Panduan AWS Config Pengembang — AWS Config Layanan menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- AWS Security Hub— Ini Layanan AWS memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS. Security Hub menggunakan kontrol keamanan untuk sumber

daya AWS Anda serta untuk memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik. Untuk daftar layanan dan kontrol yang didukung, lihat Referensi kontrol Security Hub.

- Amazon GuardDuty — Ini Layanan AWS mendeteksi potensi ancaman terhadap beban kerja Akun AWS, kontainer, dan data Anda dengan memantau lingkungan Anda untuk aktivitas mencurigakan dan berbahaya. GuardDuty dapat membantu Anda mengatasi berbagai persyaratan kepatuhan, seperti PCIDSS, dengan memenuhi persyaratan deteksi intrusi yang diamanatkan oleh kerangka kerja kepatuhan tertentu.
- AWS Audit ManagerIni Layanan AWS membantu Anda terus mengaudit AWS penggunaan Anda untuk menyederhanakan cara Anda mengelola risiko dan kepatuhan terhadap peraturan dan standar industri.

Ketahanan di Amazon Timestream untuk InfluxDB

Infrastruktur AWS global dibangun di sekitar AWS Wilayah dan Zona Ketersediaan. AWS Wilayah menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang 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 AWS Wilayah dan Availability Zone, lihat Infrastruktur AWS Global.

Amazon TimeStream untuk InfluxDB secara berkala mengambil cadangan internal dan mempertahankannya selama 24 jam untuk mendukung ketersediaan dan daya tahan. Snapshot diambil selama penghapusan dan disimpan selama 30 hari untuk mendukung pemulihan. Untuk mengakses atau menggunakan ini, ajukan tiket di AWS dukungan.

Anda dapat membuat instans dengan kemampuan pemulihan Multi-AZ. Untuk informasi selengkapnya, lihat Penerapan instans DB multi-AZ.

Keamanan infrastruktur di Amazon Timestream untuk InfluxDB

Sebagai layanan terkelola, Amazon Timestream for InfluxDB dilindungi oleh prosedur keamanan jaringan AWS global yang dijelaskan dalam whitepaper Amazon Web Services: Overview of Security Processes.

Anda menggunakan API panggilan pesawat kontrol yang AWS dipublikasikan untuk mengakses Timestream untuk InfluxDB melalui jaringan. Untuk informasi selengkapnya, lihat Mengontrol pesawat dan pesawat data. Klien harus mendukung Transport Layer Security (TLS) 1.2 atau yang lebih baru. Kami merekomendasikan TLS 1.2 atau 1.3. Klien juga harus mendukung cipher suite dengan perfect forward secrecy (PFS) seperti Ephemeral Diffie-Hellman () atau Elliptic Curve Ephemeral Diffie-Hellman (). DHE 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 IAM prinsipal. Atau Anda dapat menggunakan AWS Security Token Service (AWS STS) untuk menghasilkan kredensial keamanan sementara untuk menandatangani permintaan.

Timestream untuk InfluxDB dirancang sehingga lalu lintas Anda diisolasi ke AWS Wilayah tertentu tempat Timestream Anda untuk instans InfluxDB berada.

Grup keamanan

Kelompok keamanan mengontrol akses yang dimiliki oleh lalu lintas dalam dan keluar dari instans DB. Secara default, akses jaringan dinonaktifkan ke instans DB. Anda dapat menentukan aturan dalam kelompok keamanan yang memungkinkan akses dari rentang alamat IP, port, atau kelompok keamanan. Setelah aturan masuk dikonfigurasi, aturan yang sama berlaku untuk semua instans DB yang dikaitkan dengan grup keamanan tersebut.

Untuk informasi selengkapnya, lihat Mengontrol akses ke instans DB dalam a VPC.

Analisis konfigurasi dan kerentanan di Timestream untuk InfluxDB

Konfigurasi dan kontrol TI adalah tanggung jawab bersama antara AWS dan Anda, pelanggan kami. Untuk informasi lebih lanjut, lihat model tanggung jawab AWS bersama. Selain model tanggung jawab bersama, Timestream untuk pengguna InfluxDB harus mengetahui hal-hal berikut:

- Pelanggan bertanggung jawab untuk menambalkan patch aplikasi klien mereka dengan dependensi sisi klien yang relevan.
- Pelanggan harus mempertimbangkan pengujian penetrasi jika sesuai (lihat pengujian https://aws.amazon.com/security/penetrasi/.)

Respons insiden di Timestream untuk InfluxDB

Amazon Timestream untuk insiden layanan InfluxDB dilaporkan di Dasbor Personal Health. Anda dapat mempelajari lebih lanjut tentang dasbor dan AWS Health di sini.

Timestream untuk InfluxDB mendukung pelaporan menggunakan. AWS CloudTrail Untuk informasi selengkapnya, lihat Logging Timestream untuk panggilan API InfluxDB dengan AWS CloudTrail.

Amazon Timestream untuk InfluxDB API dan titik akhir antarmuka () VPC AWS PrivateLink

Anda dapat membuat koneksi pribadi antara Amazon Timestream Anda VPC dan Amazon Timestream untuk API titik akhir bidang kontrol InfluxDB dengan membuat titik akhir antarmuka. VPC Endpoint antarmuka didukung oleh AWS PrivateLink. AWS PrivateLink memungkinkan Anda mengakses Amazon Timestream secara pribadi untuk operasi API InfluxDB tanpa gateway internetNAT, perangkat, koneksiVPN, atau AWS koneksi Direct Connect.

Instans di Anda VPC tidak memerlukan alamat IP publik untuk berkomunikasi dengan Amazon Timestream untuk API titik akhir InfluxDB. Instans Anda juga tidak memerlukan alamat IP publik untuk menggunakan Timestream yang tersedia untuk operasi API InfluxDB. Lalu lintas antara Timestream Anda VPC dan Amazon untuk InfluxDB tidak meninggalkan jaringan Amazon. Setiap titik akhir antarmuka direpresentasikan oleh satu atau beberapa antarmuka jaringan elastis di subnet Anda. Untuk informasi selengkapnya tentang antarmuka jaringan elastis, lihat Antarmuka jaringan elastis di EC2Panduan Pengguna Amazon.

- Untuk informasi selengkapnya tentang VPC titik akhir, lihat VPCTitik akhir antarmuka (AWS PrivateLink) di VPCPanduan Pengguna Amazon.
- Untuk informasi selengkapnya tentang Timestream untuk operasi InfluxDB, lihat Timestream untuk API operasi InfluxDB. API

Setelah Anda membuat VPC titik akhir antarmuka, jika Anda mengaktifkan DNS nama host pribadi untuk titik akhir, Timestream default untuk titik akhir InfluxDB (https://timestream-influxb.*Region*.amazonaws.com) menyelesaikan ke titik akhir Anda. VPC Jika Anda tidak mengaktifkan DNS nama host pribadi, Amazon VPC menyediakan nama DNS titik akhir yang dapat Anda gunakan dalam format berikut:

```
VPC_Endpoint_ID.timestream-influxb.Region.vpce.amazonaws.com
```

Untuk informasi selengkapnya, lihat VPCTitik Akhir Antarmuka (AWS PrivateLink) di Panduan VPC Pengguna Amazon. Timestream untuk InfluxDB mendukung membuat panggilan ke semua APITindakan di dalam Anda. VPC

> **Note**
>
> DNSNama host pribadi dapat diaktifkan hanya untuk satu VPC titik akhir di. VPC Jika Anda ingin membuat VPC titik akhir tambahan maka DNS nama host pribadi harus dinonaktifkan untuk itu.

Pertimbangan untuk titik akhir VPC

Sebelum menyiapkan VPC titik akhir antarmuka untuk Amazon Timestream untuk titik akhir API InfluxDB, pastikan Anda meninjau properti dan batasan titik akhir Antarmuka di Panduan Pengguna Amazon. VPC Semua API operasi Timestream untuk InfluxDB yang relevan untuk mengelola Amazon Timestream untuk sumber daya InfluxDB tersedia dari penggunaan Anda. VPC AWS PrivateLink VPCkebijakan endpoint didukung untuk Timestream untuk titik akhir API InfluxDB. Secara default, akses penuh ke Timestream untuk API operasi InfluxDB diizinkan melalui titik akhir. Untuk informasi selengkapnya, lihat Mengontrol akses ke layanan dengan VPC titik akhir di Panduan VPC Pengguna Amazon.

Membuat VPC titik akhir antarmuka untuk Timestream untuk InfluxDB API

Anda dapat membuat VPC titik akhir untuk Amazon Timestream untuk API InfluxDB menggunakan konsol Amazon VPC atau. AWS CLI Untuk informasi selengkapnya, lihat Membuat titik akhir antarmuka di Panduan VPC Pengguna Amazon.

Setelah Anda membuat VPC titik akhir antarmuka, Anda dapat mengaktifkan nama DNS host pribadi untuk titik akhir. Ketika Anda melakukannya, Amazon Timestream default untuk titik akhir InfluxDB (https://timestream-influxb.*Region*.amazonaws.com) menyelesaikan ke titik akhir Anda. VPC Untuk informasi selengkapnya, lihat Mengakses layanan melalui titik akhir antarmuka di VPCPanduan Pengguna Amazon.

Membuat kebijakan VPC endpoint untuk Amazon Timestream untuk InfluxDB API

Anda dapat melampirkan kebijakan endpoint ke VPC endpoint yang mengontrol akses ke Timestream untuk InfluxDB. API Kebijakan menentukan informasi berikut:

- Prinsipal yang dapat melakukan tindakan.

- Tindakan yang dapat dilakukan.
- Sumber daya yang menjadi target tindakan.

Untuk informasi selengkapnya, lihat Mengontrol akses ke layanan dengan VPC titik akhir di Panduan VPC Pengguna Amazon.

Example VPCkebijakan endpoint untuk Timestream untuk tindakan InfluxDB API

Berikut ini adalah contoh kebijakan endpoint untuk Timestream untuk InfluxDB. API Saat dilampirkan ke titik akhir, kebijakan ini memberikan akses ke Timestream untuk API tindakan InfluxDB yang terdaftar untuk semua prinsipal di semua sumber daya.

```
{
 "Statement": [{
  "Principal": "*",
  "Effect": "Allow",
  "Action": [
   "timestream-influxb:CreateDbInstance",
   "timestream-influxb:UpdateDbInstance"
  ],
  "Resource": "*"
 }]
}
```

Example VPCkebijakan endpoint yang menolak semua akses dari akun tertentu AWS

Kebijakan VPC endpoint berikut menyangkal akun AWS *123456789012* semua akses ke sumber daya menggunakan titik akhir. Kebijakan ini mengizinkan semua tindakan dari akun lainnya.

```
{
 "Statement": [{
   "Action": "*",
   "Effect": "Allow",
   "Resource": "*",
   "Principal": "*"
  },
  {
   "Action": "*",
   "Effect": "Deny",
   "Resource": "*",
   "Principal": {
```

```
    "AWS": [
     "123456789012"
    ]
   }
  }
 ]
}
```

Praktik terbaik keamanan untuk Timestream untuk InfluxDB

Amazon TimeStream untuk InfluxDB menyediakan sejumlah fitur keamanan untuk dipertimbangkan saat Anda mengembangkan dan menerapkan kebijakan keamanan Anda sendiri. Praktik terbaik berikut adalah pedoman umum dan tidak mewakili solusi keamanan yang lengkap. Karena praktik terbaik ini mungkin tidak sesuai atau tidak memadai untuk lingkungan Anda, perlakukan itu sebagai pertimbangan yang bermanfaat, bukan sebagai resep.

Terapkan akses hak akses paling rendah

Saat memberikan izin, Anda memutuskan siapa yang mendapatkan izin apa untuk Timestream untuk sumber daya InfluxDB. Anda memungkinkan tindakan tertentu yang ingin Anda lakukan di sumber daya tersebut. Oleh karena itu, Anda harus memberikan hanya izin yang diperlukan untuk melaksanakan tugas. Menerapkan akses hak istimewa yang terkecil adalah hal mendasar dalam mengurangi risiko keamanan dan dampak yang dapat diakibatkan oleh kesalahan atau niat jahat.

Gunakan IAM peran

Aplikasi produsen dan klien harus memiliki kredensyal yang valid untuk mengakses Timestream untuk instans DB InfluxDB. Anda tidak boleh menyimpan AWS kredensyal secara langsung di aplikasi klien atau di bucket Amazon S3. Ini adalah kredensial jangka panjang yang tidak dirotasi secara otomatis dan dapat menimbulkan dampak bisnis yang signifikan jika dibobol.

Sebagai gantinya, Anda harus menggunakan IAM peran untuk mengelola kredensyal sementara untuk aplikasi produsen dan klien Anda untuk mengakses Timestream untuk instans DB InfluxDB. Saat Anda menggunakan peran, Anda tidak perlu menggunakan kredensial jangka panjang (seperti nama pengguna dan kata sandi atau access key) untuk mengakses sumber daya lainnya.

Untuk informasi selengkapnya, lihat topik berikut di Panduan IAM Pengguna:

- IAMPeran
- Skenario Umum untuk Peran: Pengguna, Aplikasi, dan Layanan

Gunakan AWS Identity and Access Management (IAM) akun untuk mengontrol akses ke Amazon Timestream untuk operasi InfluxDB, terutama API operasi yang membuat, memodifikasi, atau menghapus Amazon Timestream untuk sumber daya InfluxDB. Sumber daya tersebut termasuk instans DB, grup keamanan, dan grup parameter.

- Buat pengguna individual untuk setiap orang yang mengelola Amazon Timestream untuk sumber daya InfluxDB, termasuk Anda sendiri. Jangan gunakan kredensyal AWS root untuk mengelola Amazon Timestream untuk sumber daya InfluxDB.
- Beri setiap pengguna set izin minimum yang diperlukan untuk melakukan tugas-tugasnya.
- Gunakan IAM grup untuk mengelola izin secara efektif untuk beberapa pengguna.
- Putar IAM kredensyal Anda secara teratur.
- Konfigurasikan AWS Secrets Manager untuk secara otomatis memutar rahasia Amazon Timestream untuk InfluxDB. Untuk informasi selengkapnya, lihat Memutar AWS rahasia Secrets Manager Anda di Panduan Pengguna AWS Secrets Manager. Anda juga dapat mengambil kredensialnya dari AWS Secrets Manager secara terprogram. Untuk informasi selengkapnya, lihat Mengambil nilai rahasia di Panduan Pengguna AWS Secrets Manager.
- Amankan Timestream Anda untuk API token masuknya InfluxDB dengan menggunakan. APItoken

Terapkan Enkripsi Sisi Server di Sumber Daya Dependen

Data saat istirahat dan data dalam perjalanan dapat dienkripsi dalam Timestream untuk InfluxDB. Untuk informasi selengkapnya, lihat Enkripsi bergerak.

Gunakan CloudTrail untuk Memantau API Panggilan

Timestream untuk InfluxDB terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di Timestream untuk InfluxDB.

Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat ke Timestream untuk InfluxDB, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk informasi selengkapnya, lihat the section called “Logging Timestream untuk LiveAnalytics API panggilan dengan AWS CloudTrail”.

Amazon TimeStream untuk InfluxDB mendukung CloudTrail peristiwa bidang kontrol, tetapi bukan bidang data. Untuk informasi selengkapnya, lihat Mengontrol pesawat dan pesawat data.

Aksesibilitas publik

Saat meluncurkan instans DB di dalam virtual private cloud (VPC) berdasarkan VPC layanan Amazon, Anda dapat mengaktifkan atau menonaktifkan aksesibilitas publik untuk instans DB tersebut. Untuk menentukan apakah instans DB yang Anda buat memiliki DNS nama yang menyelesaikan ke alamat IP publik, Anda menggunakan parameter aksesibilitas Publik. Dengan menggunakan parameter ini, Anda dapat menentukan apakah ada akses publik ke instans DB

Jika instans DB Anda ada VPC tetapi tidak dapat diakses publik, Anda juga dapat menggunakan koneksi atau AWS Site-to-Site VPN koneksi AWS Direct Connect untuk mengaksesnya dari jaringan pribadi.

Jika instans DB Anda dapat diakses publik, pastikan untuk mengambil langkah-langkah untuk mencegah atau membantu mengurangi penolakan ancaman terkait layanan. Untuk informasi selengkapnya, lihat Pengantar serangan penolakan layanan dan Melindungi jaringan.

APIreferensi

Untuk daftar lengkap dan detail Amazon TimeStream untuk InfluxDB, lihat Amazon TimeStream untuk InfluxDBAPIs. APIs

Untuk kode kesalahan yang umum untuk semua AWS layanan, lihat bagian AWS Support.

Riwayat dokumen

Perubahan	Deskripsi	Tanggal
Pembaruan khusus dokumentasi	Memperbarui topik Kuota untuk memisahkan kuota default dan batas sistem.	Oktober 22, 2024
Amazon Timestream sekarang mendukung wawasan kueri	Timestream sekarang menyertakan dukungan untuk fitur wawasan kueri yang membantu Anda mengoptim alkan kueri, meningkatkan kinerjanya, dan mengurangi biaya.	Oktober 22, 2024

Amazon Timestream untuk InfluxDB memperbarui ke kebijakan yang ada.	Amazon Timestream untuk InfluxDB telah menambahkan `ec2:DescribeRouteTables` tindakan ke kebijakan `AmazonTimestreamInfluxDBFullAccess` terkelola yang ada untuk menjelaskan tabel rute Anda	Oktober 8, 2024
`AmazonTimestreamReadOnlyAccess` — Perbarui ke kebijakan yang ada	Timestream for LiveAnalytics telah menambahkan `DescribeAccountSettings` izin ke kebijakan `AmazonTimestreamReadOnlyAccess` terkelola untuk menjelaskan Akun AWS setelan.	3 Juni 2024
Amazon Timestream untuk LiveAnalytics saat ini mendukung Unit Komputasi Timestream () TCUs	Amazon Timestream untuk LiveAnalytics saat ini menyertakan dukungan untuk Timestream Compute Units (TCUs) untuk mengukur kapasitas komputasi yang dialokasikan untuk kebutuhan kueri Anda.	April 29, 2024

Kebijakan baru ditambahkan	Amazon Timestream untuk InfluxDB menambahkan dua kebijakan baru: Satu yang memungkinkan layanan untuk mengelola antarmuka jaringan dan grup keamanan di akun Anda. Untuk informasi lebih lanjut, lihat AmazonTimestreamInfluxDBServiceRolePolicy. Lain yang menyediakan akses administratif penuh untuk membuat, memperbarui, menghapus, dan mencantumkan instans Amazon TimeStream InfluxDB serta membuat dan membuat daftar grup parameter. Untuk informasi lebih lanjut, lihat AmazonTimestreamInfluxDBFullAccess.	Maret 14, 2024
Amazon Timestream untuk InfluxDB sekarang tersedia secara umum.	Dokumentasi ini mencakup rilis awal Amazon Timestream untuk InfluxDB.	Maret 14, 2024

Amazon Timestream untuk acara LiveAnalytics Kueri tersedia di AWS CloudTrail	Amazon Timestream untuk LiveAnalytics saat ini menerbitkan peristiwa API data Kueri ke. AWS CloudTrai l Pelanggan dapat mengaudit semua API permintaan Kueri yang dibuat di AWS akun mereka, dan melihat informasi seperti IAM Pengguna/ Peran mana yang membuat permintaan, kapan permintaa n dibuat, database dan tabel mana yang ditanyakan, dan ID Kueri permintaan.	12 September 2023
Amazon Timestream untuk LiveAnalytics UNLOAD	Amazon Timestream untuk LiveAnalytics saat ini mendukung UNLOAD untuk mengekspor hasil kueri ke S3.	12 Mei 2023
Amazon Timestream untuk LiveAnalytics pembaruan ke kebijakan yang ada.	Izin pemuatan batch ditambahkan ke kebijakan terkelola.	Februari 24, 2023
Amazon Timestream untuk pemuatan LiveAnalytics batch.	Amazon Timestream untuk LiveAnalytics saat ini mendukung fungsionalitas pemuatan batch.	Februari 24, 2023
Amazon Timestream untuk LiveAnalytics saat ini mendukung. AWS Backup	Amazon Timestream untuk LiveAnalytics saat ini mendukung. AWS Backup	14 Desember 2022

Amazon Timestream untuk LiveAnalytics pembaruan kebijakan terkelola AWS	Informasi baru tentang kebijakan AWS terkelola dan Amazon Timestream untuk LiveAnalytics, termasuk pembaruan kebijakan terkelola yang ada.	29 November 2021
Amazon Timestream untuk LiveAnalytics mendukung kueri terjadwal	Amazon Timestream untuk LiveAnalytics saat ini mendukung menjalankan kueri atas nama Anda, berdasarkan jadwal.	29 November 2021
Amazon Timestream untuk LiveAnalytics mendukung penyimpanan magnetik.	Amazon Timestream untuk LiveAnalytics saat ini mendukung penggunaan penyimpanan magnetik untuk penulisan tabel Anda.	29 November 2021
Amazon Timestream untuk catatan LiveAnalytics multi-ukuran.	Amazon Timestream untuk LiveAnalytics saat ini mendukung format yang lebih ringkas untuk menyimpan data deret waktu Anda.	29 November 2021
Amazon Timestream untuk LiveAnalytics pembaruan kebijakan terkelola AWS	Informasi baru tentang kebijakan AWS terkelola dan Amazon Timestream untuk LiveAnalytics, termasuk pembaruan kebijakan terkelola yang ada.	24 Mei 2021
Amazon Timestream untuk sekarang LiveAnalytics tersedia di wilayah eropa (frankfurt).	Amazon Timestream untuk sekarang LiveAnalytics umumnya tersedia di wilayah Eropa (Frankfurt) (). `eu-central-1`	23 April 2021

Amazon Timestream for LiveAnalytics is sekarang mendukung VPC endpoints ().AWS PrivateLink	Amazon Timestream untuk LiveAnalytics saat ini mendukung penggunaan VPC endpoints ().AWS PrivateLink	23 Maret 2021
Amazon Timestream sekarang mendukung kueri lintas tabel.	Anda dapat menggunak an Amazon Timestream LiveAnalytics untuk menjalank an kueri lintas tabel.	10 Februari 2021
Amazon Timestream untuk LiveAnalytics saat ini mendukung statistik eksekusi kueri yang disempurnakan.	Amazon Timestream untuk LiveAnalytics saat ini mendukung peningkatan statistik eksekusi kueri, seperti jumlah data yang dipindai.	10 Februari 2021
Amazon Timestream untuk LiveAnalytics saat ini mendukung fungsi deret waktu lanjutan.	Anda dapat menggunak an Amazon Timestream LiveAnalytics untuk menjalank an SQL kueri dengan fungsi deret waktu lanjutan, seperti turunan, integral, dan korelasi.	10 Februari 2021
Amazon Timestream untuk saat LiveAnalytics iniHIPAA, ISO, dan PCI sesuai.	Sekarang Anda dapat menggunakan Amazon Timestream LiveAnalytics untuk beban kerja yang memerlukan HIPAAISO, dan PCI infrastruktur yang sesuai.	27 Januari 2021

Amazon Timestream untuk LiveAnalytics saat ini mendukung telegraf dan grafana sumber terbuka.	Anda sekarang dapat menggunakan Telegraf, agen server open-source, berbasis plugin untuk mengumpulkan dan melaporkan metrik, dan Grafana, platform analitik dan pemantauan sumber terbuka untuk database, dengan Amazon Timestream for. LiveAnalytics	25 November 2020
Amazon Timestream untuk sekarang LiveAnalytics tersedia secara umum.	Dokumentasi ini mencakup rilis awal Amazon Timestream untuk. LiveAnalytics	30 September 2020

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.