aws

Entwicklerhandbuch

Amazon Timestream

Amazon Timestream: Entwicklerhandbuch

Table of Contents

Wofür ist Amazon Timestream LiveAnalytics?

Amazon Timestream for LiveAnalytics ist eine schnelle, skalierbare, vollständig verwaltete, speziell entwickelte Zeitreihendatenbank, die es einfach macht, Billionen von Zeitreihendatenpunkten pro Tag zu speichern und zu analysieren. Timestream for LiveAnalytics spart Ihnen Zeit und Kosten bei der Verwaltung des Lebenszyklus von Zeitreihendaten, indem aktuelle Daten im Speicher aufbewahrt und historische Daten auf eine kostenoptimierte Speicherebene verschoben werden, die auf benutzerdefinierten Richtlinien basiert. Mit LiveAnalytics der speziell entwickelten Abfrage-Engine von Timestream for können Sie auf aktuelle und historische Daten zugreifen und diese gemeinsam analysieren, ohne deren Speicherort angeben zu müssen. Amazon Timestream for LiveAnalytics verfügt über integrierte Funktionen zur Zeitreihenanalyse, mit denen Sie Trends und Muster in Ihren Daten nahezu in Echtzeit erkennen können. Timestream for LiveAnalytics ist serverlos und wird automatisch nach oben oder unten skaliert, um Kapazität und Leistung anzupassen. Da Sie die zugrunde liegende Infrastruktur nicht verwalten müssen, können Sie sich auf die Optimierung und Entwicklung Ihrer Anwendungen konzentrieren.

Timestream for lässt sich LiveAnalytics auch in häufig verwendete Dienste für Datenerfassung, Visualisierung und maschinelles Lernen integrieren. Sie können Daten an Amazon Timestream senden, um Amazon Kinesis AWS IoT Core MSK, Amazon und Open Source Telegraf zu LiveAnalytics verwenden. Sie können Daten mithilfe von Amazon- QuickSight, Grafana- und Business Intelligence-Tools über JDBC visualisieren. Sie können Amazon auch SageMaker mit Timestream LiveAnalytics für maschinelles Lernen verwenden.

Timestream für wichtige Vorteile LiveAnalytics

Die wichtigsten Vorteile von Amazon Timestream für LiveAnalytics sind:

- Serverlos mit auto-scaling — Mit Amazon Timestream for LiveAnalytics müssen keine Server verwaltet und keine Kapazität bereitgestellt werden. Wenn sich die Anforderungen Ihrer Anwendung ändern, skaliert Timestream for LiveAnalytics automatisch, um die Kapazität anzupassen.
- Datenlebenszyklusmanagement — Amazon Timestream for LiveAnalytics vereinfacht den komplexen Prozess des Datenlebenszyklusmanagements. Es bietet Speicher-Tiering mit einem Speicherspeicher für aktuelle Daten und einem Magnetspeicher für historische Daten. Amazon Timestream automatisiert die Übertragung von Daten vom Speicherspeicher zum Magnetspeicher auf der Grundlage von vom Benutzer konfigurierbaren Richtlinien.

- Vereinfachter Datenzugriff — Mit Amazon Timestream for müssen Sie nicht mehr unterschiedliche Tools verwenden LiveAnalytics, um auf aktuelle und historische Daten zuzugreifen. LiveAnalyticsDie speziell entwickelte Abfrage-Engine von Amazon Timestream for greift transparent auf Daten auf verschiedenen Speicherebenen zu und kombiniert sie, ohne dass Sie den Datenstandort angeben müssen.
- Speziell für Zeitreihen konzipiert — Mithilfe der integrierten Zeitreihenfunktionen zur GlättungSQL, Approximation und Interpolation können Sie Zeitreihendaten schnell analysieren. Timestream for unterstützt LiveAnalytics auch erweiterte Aggregate, Fensterfunktionen und komplexe Datentypen wie Arrays und Zeilen.
- Immer verschlüsselt — Amazon Timestream for LiveAnalytics stellt sicher, dass Ihre Zeitreihendaten immer verschlüsselt sind, unabhängig davon, ob sie gespeichert sind oder übertragen werden. Mit Amazon Timestream for können Sie LiveAnalytics auch einen vom AWS KMS Kunden verwalteten Schlüssel (CMK) für die Verschlüsselung von Daten im Magnetspeicher angeben.
- Hohe Verfügbarkeit — Amazon Timestream gewährleistet eine hohe Verfügbarkeit Ihrer Schreib- und Leseanfragen, indem Daten automatisch repliziert und Ressourcen in mindestens 3 verschiedenen Availability Zones innerhalb einer einzigen Region zugewiesen werden. AWS Weitere Informationen finden Sie im Timestream Service Level Agreement.
- Haltbarkeit — Amazon Timestream gewährleistet die Beständigkeit Ihrer Daten, indem es Ihre Speicher- und Magnetspeicherdaten automatisch in verschiedenen Availability Zones innerhalb einer einzigen AWS Region repliziert. Alle Ihre Daten werden auf die Festplatte geschrieben, bevor Ihre Schreibanforderung als abgeschlossen bestätigt wird.

Timestream für Anwendungsfälle LiveAnalytics

Zu den Beispielen für eine wachsende Liste von Anwendungsfällen für Timestream gehören: LiveAnalytics

- Überwachung von Metriken zur Verbesserung der Leistung und Verfügbarkeit Ihrer Anwendungen.
- Speicherung und Analyse industrieller Telemetrie zur Optimierung der Geräteverwaltung und -wartung.
- Verfolgung der Benutzerinteraktion mit einer Anwendung im Laufe der Zeit.
- Speicherung und Analyse von IoT-Sensordaten.

Erste Schritte mit Timestream für LiveAnalytics

Wir empfehlen, zuerst die folgenden Abschnitte zu lesen:

- Tutorial- Um eine Datenbank mit Beispieldatensätzen zu erstellen und Beispielabfragen auszuführen.
- Amazon Timestream für Konzepte LiveAnalytics - Um grundlegende LiveAnalytics Timestream-Konzepte zu erlernen.
- Zugreifen auf Timestream für LiveAnalytics- Um zu erfahren, wie man auf Timestream zugreift, um die Konsole zu LiveAnalytics verwenden, AWS CLI, oder. API
- Kontingente- Um mehr über Kontingente für die Anzahl von Timestreams für LiveAnalytics Komponenten zu erfahren, die Sie bereitstellen können.

Im Folgenden erfahren Sie, wie Sie schnell mit der Entwicklung von Anwendungen für Timestream beginnen können: LiveAnalytics

- Mit dem AWS SDKs
- Referenz zur Abfragesprache

Funktionsweise

Die folgenden Abschnitte bieten einen Überblick über die Servicekomponenten von Amazon Timestream for Live Analytics und deren Zusammenspiel.

Nachdem Sie diese Einführung gelesen haben, erfahren Sie in den Zugreifen auf Timestream für LiveAnalytics Abschnitten, wie Sie über die Konsole,, AWS CLI oder auf Timestream for Live Analytics zugreifen können. SDKs

Themen

- Amazon Timestream für Konzepte LiveAnalytics
- Architektur
- Schreibt
- Speicher
- Abfragen

- Geplante Abfragen
- Timestream-Recheneinheit () TCU

Amazon Timestream für Konzepte LiveAnalytics

Zeitreihendaten sind eine Folge von Datenpunkten, die über ein Zeitintervall aufgezeichnet wurden. Diese Art von Daten wird zur Messung von Ereignissen verwendet, die sich im Laufe der Zeit ändern. Beispiele sind unter anderem:

- Aktienkurse im Laufe der Zeit
- Temperaturmessungen im Laufe der Zeit
- CPUNutzung einer EC2 Instanz im Laufe der Zeit

Bei Zeitreihendaten besteht jeder Datenpunkt aus einem Zeitstempel, einem oder mehreren Attributen und dem Ereignis, das sich im Laufe der Zeit ändert. Diese Daten können verwendet werden, um Einblicke in die Leistung und den Zustand einer Anwendung zu gewinnen, Anomalien zu erkennen und Optimierungsmöglichkeiten zu identifizieren. Beispielsweise möchten DevOps Ingenieure möglicherweise Daten einsehen, mit denen Änderungen an den Leistungskennzahlen der Infrastruktur gemessen werden. Hersteller möchten möglicherweise IoT-Sensordaten verfolgen, mit denen Änderungen an Geräten in einer Anlage gemessen werden. Online-Marketer möchten möglicherweise Clickstream-Daten analysieren, die erfassen, wie ein Nutzer im Laufe der Zeit auf einer Website navigiert. Da Zeitreihendaten aus mehreren Quellen in extrem großen Mengen generiert werden, müssen sie kostengünstig und nahezu in Echtzeit erfasst werden. Daher ist ein effizienter Speicher erforderlich, der die Organisation und Analyse der Daten unterstützt.

Im Folgenden sind die wichtigsten Konzepte von Timestream for aufgeführt. LiveAnalytics

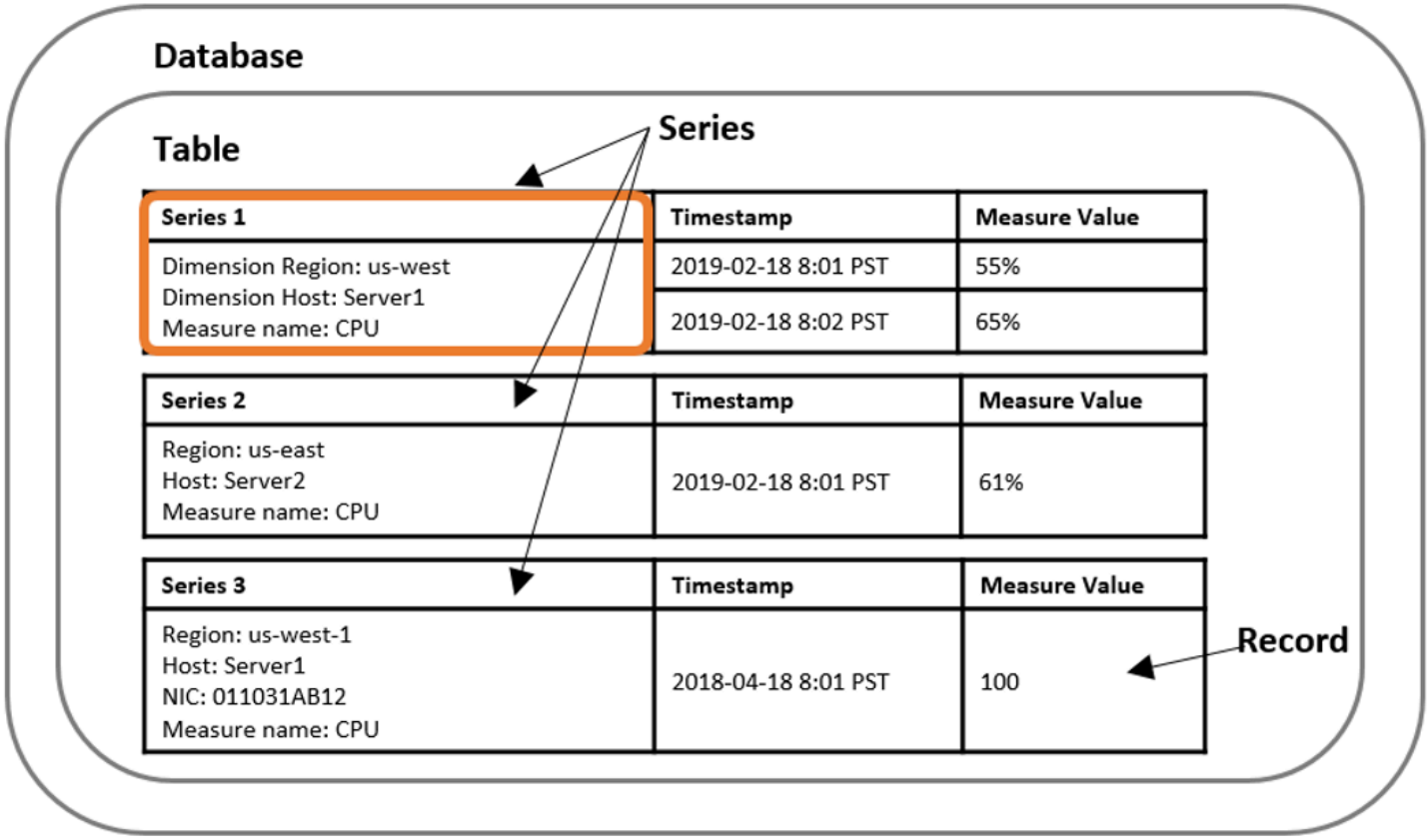
- Zeitreihe — Eine Folge von einem oder mehreren Datenpunkten (oder Datensätzen), die über ein Zeitintervall aufgezeichnet wurden. Beispiele hierfür sind der Kurs einer Aktie im Laufe der Zeit, die CPU Speicherauslastung einer EC2 Instanz im Laufe der Zeit und die Temperatur-/Druckmessung eines IoT-Sensors im Laufe der Zeit.
- Datensatz — Ein einzelner Datenpunkt in einer Zeitreihe.
- Dimension — Ein Attribut, das die Metadaten einer Zeitreihe beschreibt. Dimensionen bestehen aus einem Dimensionsnamen und einem Dimensionswert. Betrachten Sie die folgenden Beispiele:
  - Wenn eine Börse als Dimension betrachtet wird, lautet der Name der Dimension „Börse“ und der Dimensionswert „“ NYSE

  - Wenn eine AWS Region als Dimension betrachtet wird, lautet der Dimensionsname „Region“ und der Dimensionswert ist „us-east-1"
  - Bei einem IoT-Sensor lautet der Dimensionsname „Geräte-ID“ und der Dimensionswert „12345"
- Messen — Der tatsächliche Wert, der mit dem Datensatz gemessen wird. Beispiele hierfür sind der Aktienkurs, die CPU Speicherauslastung und die gemessene Temperatur oder Luftfeuchtigkeit. Kennzahlen bestehen aus Kennzahlnamen und Kennzahlwerten. Betrachten Sie die folgenden Beispiele:
  - Bei einem Aktienkurs lautet der Kennzahlname „Aktienkurs“ und der Kennzahlwert ist der tatsächliche Aktienkurs zu einem bestimmten Zeitpunkt.
  - Bei der CPU Auslastung lautet der Kennzahlname „CPUAuslastung“ und der Kennzahlwert entspricht der tatsächlichen CPU Auslastung.

  Kennzahlen können in Timestream LiveAnalytics als Datensätze mit mehreren oder einzelnen Messwerten modelliert werden. Weitere Informationen finden Sie unter Datensätze mit mehreren Kennzahlen im Vergleich zu Datensätzen mit nur einer Kennzahl.
- Zeitstempel — Gibt an, wann eine Kennzahl für einen bestimmten Datensatz erfasst wurde. Timestream for LiveAnalytics unterstützt Zeitstempel mit einer Granularität im Nanosekundenbereich.
- Tabelle — Ein Container für eine Reihe verwandter Zeitreihen.
- Datenbank — Ein Container auf oberster Ebene für Tabellen.

Eine Zusammenfassung von Timestream für Konzepte LiveAnalytics

Eine Datenbank enthält 0 oder mehr Tabellen. Jede Tabelle enthält 0 oder mehr Zeitreihen. Jede Zeitreihe besteht aus einer Folge von Datensätzen über ein bestimmtes Zeitintervall mit einer bestimmten Granularität. Jede Zeitreihe kann anhand ihrer Metadaten oder Dimensionen, ihrer Daten oder Kennzahlen und ihrer Zeitstempel beschrieben werden.

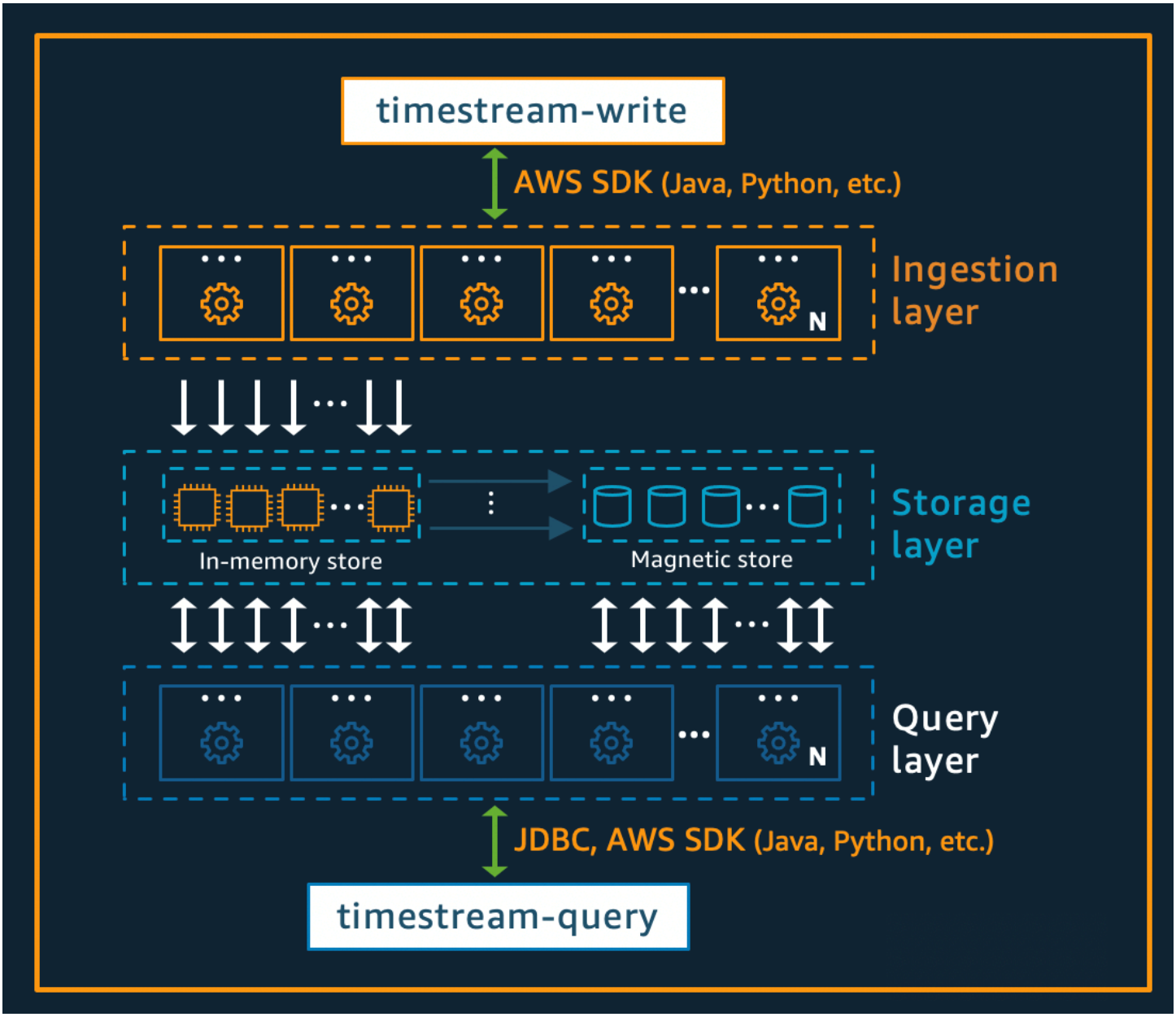


Architektur

Amazon Timestream for Live Analytics wurde von Grund auf für die Erfassung, Speicherung und Verarbeitung von Zeitreihendaten in großem Umfang konzipiert. Die serverlose Architektur unterstützt vollständig entkoppelte Datenaufnahme-, Speicher- und Abfrageverarbeitungssysteme, die unabhängig voneinander skaliert werden können. Dieses Design vereinfacht jedes Subsystem und macht es einfacher, unerschütterliche Zuverlässigkeit zu erreichen, Skalierungsengpässe zu beseitigen und die Wahrscheinlichkeit korrelierter Systemausfälle zu verringern. Jeder dieser Faktoren wird mit der Skalierung des Systems immer wichtiger.

Themen

- Schreiben Sie Architektur
- Speicherarchitektur
- Architektur abfragen
- Zellulare Architektur



Schreiben Sie Architektur

Beim Schreiben von Zeitreihendaten leitet Amazon Timestream for Live Analytics Schreibvorgänge für eine Tabelle oder Partition an eine fehlertolerante Speicherinstanz weiter, die Datenschreibvorgänge mit hohem Durchsatz verarbeitet. Der Speicherspeicher wiederum sorgt für Stabilität in einem separaten Speichersystem, das die Daten in drei Availability Zones repliziert (). AZs Die Replikation basiert auf einem Quorum, sodass der Verlust von Knoten oder einer gesamten AZ die Schreibverfügbarkeit nicht beeinträchtigt. Fast in Echtzeit werden andere In-Memory-Speicherknoten mit den Daten synchronisiert, um Abfragen zu bearbeiten. Die Reader-

Replikatknoten erstrecken sich AZs ebenfalls über mehrere Knoten, um eine hohe Leseverfügbarkeit zu gewährleisten.

Timestream for Live Analytics unterstützt das direkte Schreiben von Daten in den Magnetspeicher für Anwendungen, die spät eintreffende Daten mit geringerem Durchsatz generieren. Bei spät eintreffenden Daten handelt es sich um Daten, deren Zeitstempel vor der aktuellen Uhrzeit liegt. Ähnlich wie bei Schreibvorgängen mit hohem Durchsatz im Speicherspeicher werden die in den Magnetspeicher geschriebenen Daten dreifach repliziert, AZs und die Replikation erfolgt quorumbasiert.

Unabhängig davon, ob Daten in den Speicher oder den Magnetspeicher geschrieben werden, indexiert und partitioniert Timestream for Live Analytics Daten automatisch, bevor sie in den Speicher geschrieben werden. Eine einzelne Timestream for Live Analytics-Tabelle kann Hunderte, Tausende oder sogar Millionen von Partitionen enthalten. Einzelne Partitionen kommunizieren nicht direkt miteinander und teilen keine Daten (Shared-Nothing-Architektur). Stattdessen wird die Partitionierung einer Tabelle über einen hochverfügbaren Dienst zur Partitionsverfolgung und Indexierung nachverfolgt. Dies bietet eine weitere Trennung von Problemen, die speziell darauf ausgelegt sind, die Auswirkungen von Systemausfällen zu minimieren und die Wahrscheinlichkeit korrelierter Ausfälle erheblich zu verringern.

Speicherarchitektur

Wenn Daten in Timestream for Live Analytics gespeichert werden, werden die Daten anhand von Kontextattributen, die zusammen mit den Daten geschrieben werden, sowohl in zeitlicher Reihenfolge als auch im Zeitverlauf organisiert. Für die massive Skalierung eines Zeitreihensystems ist es wichtig, über ein Partitionierungsschema zu verfügen, das neben der Zeit auch „Raum“ unterteilt. Das liegt daran, dass die meisten Zeitreihendaten zur oder um die aktuelle Uhrzeit geschrieben werden. Daher eignet sich die Partitionierung nach Zeit allein nicht gut zur Verteilung des Schreibverkehrs oder zur effektiven Bereinigung von Daten zur Abfragezeit. Dies ist wichtig für die Verarbeitung extrem großer Zeitreihen und hat es Timestream for Live Analytics ermöglicht, serverlos um Größenordnungen höher zu skalieren als die anderen führenden Systeme, die es heute gibt. Die resultierenden Partitionen werden als „Kacheln“ bezeichnet, weil sie Teilungen eines zweidimensionalen Raums darstellen (die so konzipiert sind, dass sie eine ähnliche Größe haben). Timestream for Live Analytics-Tabellen bestehen zunächst aus einer einzelnen Partition (Kachel) und werden dann je nach Durchsatz in der räumlichen Dimension aufgeteilt. Wenn Kacheln eine bestimmte Größe erreichen, werden sie dann in der Zeitdimension aufgeteilt, um bei wachsender Datengröße eine bessere Lese-Parallelität zu erreichen.

Timestream for Live Analytics wurde entwickelt, um den Lebenszyklus von Zeitreihendaten automatisch zu verwalten. Timestream for Live Analytics bietet zwei Datenspeicher — einen In-Memory-Speicher und einen kostengünstigen Magnetspeicher. Es unterstützt auch die Konfiguration von Richtlinien auf Tabellenebene, um Daten automatisch zwischen Speichern zu übertragen. Eingehende Schreibvorgänge mit hohem Durchsatz landen im Speicherspeicher, wo die Daten für Schreibvorgänge optimiert sind. Das Gleiche gilt für Lesevorgänge, die etwa zur aktuellen Zeit ausgeführt werden, um Abfragen vom Typ Dashboard und Warnmeldungen zu starten. Wenn der Hauptzeitrahmen für Schreib-, Alarm- und Dashboard-Anforderungen abgelaufen ist, können die Daten automatisch vom Speicherspeicher zum Magnetspeicher fließen, um die Kosten zu optimieren. Timestream for Live Analytics ermöglicht zu diesem Zweck die Festlegung einer Datenaufbewahrungsrichtlinie für den Speicherspeicher. Datenschreibvorgänge für spät eintreffende Daten werden direkt in den Magnetspeicher geschrieben.

Sobald die Daten im Magnetspeicher verfügbar sind (aufgrund des Ablaufs der Aufbewahrungsfrist des Speicherspeichers oder aufgrund direkter Schreibvorgänge in den Magnetspeicher), werden sie in ein Format umorganisiert, das für das Lesen großer Datenmengen in hohem Maße optimiert ist. Der Magnetspeicher verfügt außerdem über eine Datenaufbewahrungsrichtlinie, die konfiguriert werden kann, wenn es einen bestimmten Zeitraum gibt, ab dem die Daten nicht mehr nützlich sind. Wenn die Daten den für die Aufbewahrungsrichtlinie für Magnetspeicher definierten Zeitraum überschreiten, werden sie automatisch entfernt. Daher erfolgt das Datenlebenszyklusmanagement bei Timestream for Live Analytics, abgesehen von einigen Konfigurationen, nahtlos hinter den Kulissen.

Architektur abfragen

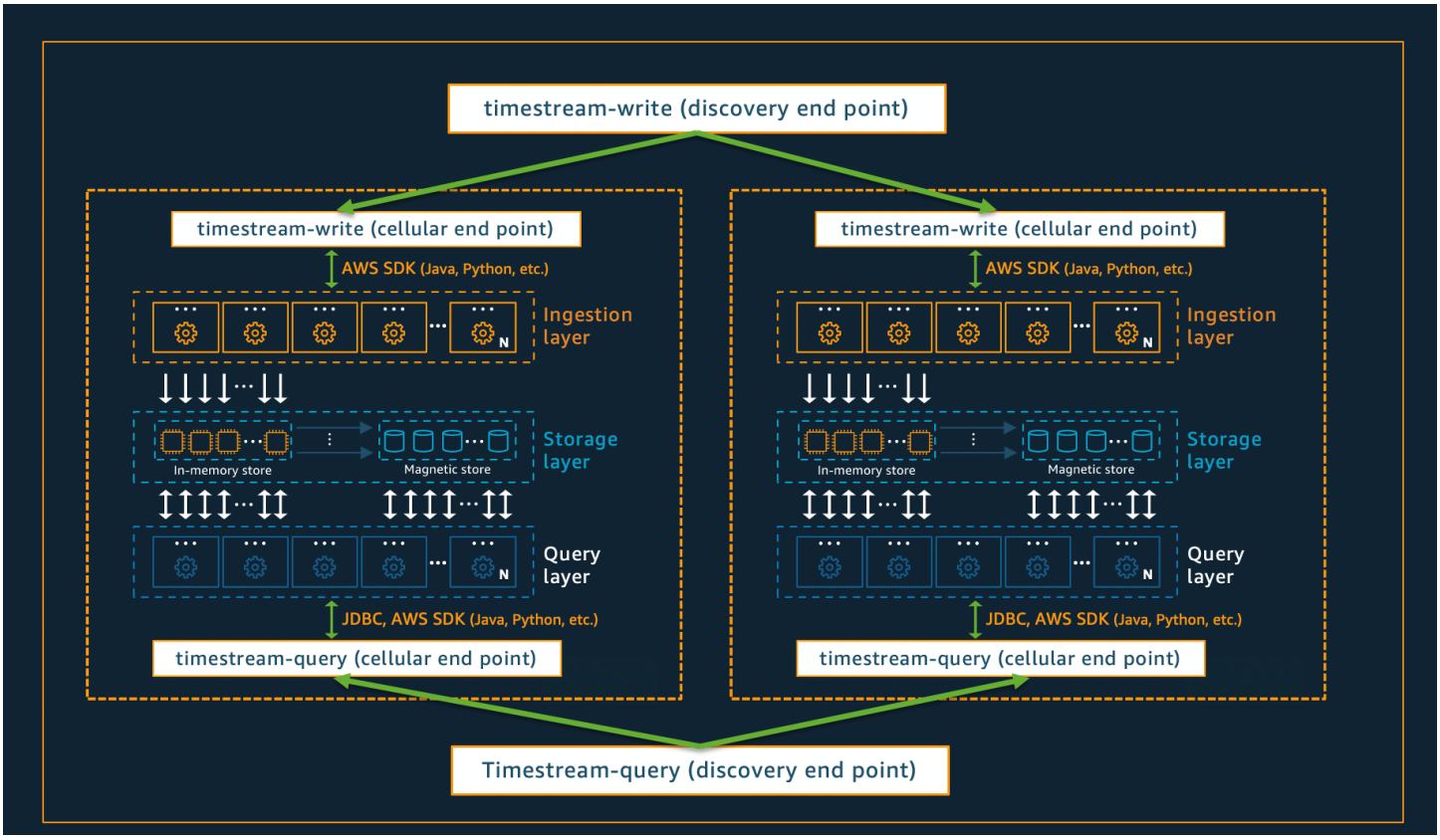
Timestream for Live Analytics-Abfragen werden in einer SQL Grammatik ausgedrückt, die Erweiterungen für zeitreihenspezifische Unterstützung bietet (zeitreihenspezifische Datentypen und Funktionen), sodass Entwickler, mit denen sie bereits vertraut sind, die Lernkurve leicht erlernen können. SQL Abfragen werden dann von einer adaptiven, verteilten Abfrage-Engine verarbeitet, die Metadaten aus dem Tile-Tracking- und Indexierungsservice verwendet, um zum Zeitpunkt der Ausgabe der Abfrage nahtlos auf Daten aus verschiedenen Datenspeichern zuzugreifen und diese zu kombinieren. Dies sorgt für ein Erlebnis, das bei den Kunden gut ankommt, da es viele der Rube-Goldberg-Komplexitäten in einer einfachen und vertrauten Datenbankabstraktion zusammenfasst.

Abfragen werden von einer eigenen Flotte von Mitarbeitern ausgeführt, wobei die Anzahl der Mitarbeiter, die für die Ausführung einer bestimmten Abfrage angeworben werden, von der Komplexität der Abfrage und der Datengröße abhängt. Die Leistung für komplexe Abfragen über große Datenmengen wird durch massive Parallelität erreicht, sowohl bei der Abfragelaufzeitflotte

als auch bei den Speicherflotten des Systems. Die Fähigkeit, riesige Datenmengen schnell und effizient zu analysieren, ist eine der größten Stärken von Timestream for Live Analytics. Eine einzelne Abfrage, die Terabyte oder sogar Petabyte an Daten umfasst, kann dazu führen, dass Tausende von Computern gleichzeitig daran arbeiten.

Zellulare Architektur

Um sicherzustellen, dass Timestream for Live Analytics eine praktisch unendliche Skalierbarkeit für Ihre Anwendungen bietet und gleichzeitig eine Verfügbarkeit von 99,99% gewährleistet, wurde das System auch unter Verwendung einer Mobilfunkarchitektur konzipiert. Anstatt das System als Ganzes zu skalieren, segmentiert Timestream for Live Analytics in mehrere kleinere Kopien von sich selbst, die als Zellen bezeichnet werden. Auf diese Weise können Zellen in vollem Umfang getestet werden, und es wird verhindert, dass ein Systemproblem in einer Zelle die Aktivität anderer Zellen in einer bestimmten Region beeinträchtigt. Timestream for Live Analytics ist zwar darauf ausgelegt, mehrere Zellen pro Region zu unterstützen, aber stellen Sie sich das folgende fiktive Szenario vor, in dem sich zwei Zellen in einer Region befinden.



In dem oben dargestellten Szenario werden die Datenaufnahme- und Abfrageanforderungen zunächst vom Erkennungsendpunkt für die Datenaufnahme bzw. Abfrage verarbeitet. Anschließend identifiziert der Erkennungsendpunkt die Zelle, die die Kundendaten enthält, und leitet die Anfrage

an den entsprechenden Aufnahme- oder Abfrageendpunkt für diese Zelle weiter. Wenn Sie den verwendenSDKs, werden diese Endpunktverwaltungsaufgaben transparent für Sie erledigt.

> **Note**
>
> Wenn Sie VPC Endgeräte mit Timestream for Live Analytics verwenden oder direkt auf REST API Vorgänge für Timestream for Live Analytics zugreifen, müssen Sie direkt mit den Mobilfunkendpunkten interagieren. Anleitungen dazu finden Sie unter VPCEndpoints für Anweisungen zum Einrichten VPC von Endpoints und unter Endpoint Discovery Pattern für Anweisungen zum direkten Aufrufen der Operationen. REST API

Schreibt

Sie können Zeitreihendaten von angeschlossenen Geräten, IT-Systemen und Industrieanlagen sammeln und sie in Timestream for Live Analytics schreiben. Timestream for Live Analytics ermöglicht es Ihnen, Datenpunkte aus einer einzelnen Zeitreihe und/oder Datenpunkte aus vielen Serien in einer einzigen Schreibanforderung zu schreiben, wenn die Zeitreihen zu derselben Tabelle gehören. Der Einfachheit halber bietet Ihnen Timestream for Live Analytics ein flexibles Schema, das die Spaltennamen und Datentypen für Ihre Timestream for Live Analytics-Tabellen auf der Grundlage der Dimensionsnamen und der Datentypen der Kennzahlwerte, die Sie beim Aufrufen von Schreibvorgängen in die Datenbank angeben, auto erkennt. Sie können auch Datenstapel in Timestream for Live Analytics schreiben.

> **Note**
>
> Timestream for Live Analytics unterstützt die Semantik für eventuelle Konsistenz bei Lesevorgängen. Wenn Sie also Daten unmittelbar nach dem Schreiben eines Datenstapels in Timestream for Live Analytics abfragen, spiegeln die Abfrageergebnisse möglicherweise nicht die Ergebnisse eines kürzlich abgeschlossenen Schreibvorgangs wider. Die Ergebnisse können auch einige veraltete Daten enthalten. In ähnlicher Weise kann eine Abfrage beim Schreiben von Zeitreihendaten mit einer oder mehreren neuen Dimensionen für einen kurzen Zeitraum eine Teilmenge von Spalten zurückgeben. Wenn Sie diese Abfrageanfragen nach kurzer Zeit wiederholen, sollten die Ergebnisse die neuesten Daten zurückgeben.

Sie können Daten mit AWS SDKs, AWS CLI, oder überAWS Lambda,, AWS IoT CoreAmazon Managed Service für Apache Flink, Amazon KinesisAmazon MSK, und schreibenOpen-Source-Telegraf.

Themen

Datentypen

Timestream for Live Analytics unterstützt die folgenden Datentypen für Schreibvorgänge.

Datentyp	Beschreibung
BIGINT	Stellt eine 64-Bit-Ganzzahl mit Vorzeichen dar.
BOOLEAN	Stellt die beiden Wahrheitswerte der Logik dar, nämlich wahr und falsch.
DOUBLE	64-Bit-Version mit variabler Genauigkeit, die den IEEE Standard 754 für binäre Gleitkomma-Arithmetik implementiert. **Note** Es gibt Funktionen für `Infinity` und `NaN` Double-We rte in Abfragesprache, die in Abfragen verwendet werden können. Sie können diese Werte jedoch nicht in Timestream schreiben.
VARCHAR	Zeichendaten mit variabler Länge und optionaler Maximallänge. Die maximale Grenze liegt bei 2 KB.

Datentyp	Beschreibung
MULTI	Datentyp für Datensätze mit mehreren Messwerten. Dieser Datentyp umfasst eine oder mehrere Kennzahlen des Typs`BIGINT`,`BOOLEAN`, `DOUBLE`,`VARCHAR`, und`TIMESTAMP` .
TIMESTAMP	Stellt eine Zeitinstanz mit einer Genauigkeit von Nanosekunden dar und verfolgt die Zeit seit der Unix-Zeit. UTC Dieser Datentyp wird derzeit nur für Datensätze mit mehreren Messwerten (d. h. innerhalb von Messwerten des Typs`MULTI`) unterstützt. *YYYY*-*MM*-*DD* *hh*:*mm*:*ss*.*sssssssss* Schreibt Unterstützungszeitstempel im Bereich bis`1970-01-01 00:00:00.000000000` . `2262-04-11 23:47:16.854775807`

Keine Schemadefinition im Voraus

Bevor Sie Daten an Amazon Timestream for Live Analytics senden, müssen Sie mithilfe der AWS Management Console Operationen Timestream for Live Analytics oder Timestream for Live Analytics SDKs eine Datenbank und eine Tabelle erstellen. API Weitere Informationen erhalten Sie unter Erstellen einer -Datenbank und Erstellen einer Tabelle. Beim Erstellen der Tabelle müssen Sie das Schema nicht im Voraus definieren. Amazon Timestream for Live Analytics erkennt das Schema automatisch anhand der Maße und Dimensionen der gesendeten Datenpunkte, sodass Sie Ihr Schema nicht mehr offline ändern müssen, um es an Ihre sich schnell ändernden Zeitreihendaten anzupassen.

Daten schreiben (Einfügungen und Upserts)

Der Schreibvorgang in Amazon Timestream for Live Analytics ermöglicht es Ihnen, Daten einzufügen und zu aktualisieren. Standardmäßig folgen Schreibvorgänge in Amazon Timestream for Live Analytics der Semantik „Der erste Writer gewinnt“. Dabei werden Daten nur als Anhängen gespeichert und doppelte Datensätze werden zurückgewiesen. Während die Semantik mit dem ersten Writer die Anforderungen vieler Zeitreihenanwendungen erfüllt, gibt es Szenarien, in denen Anwendungen bestehende Datensätze idempotent aktualisieren und/oder Daten mit der Semantik des letzten Schreibers schreiben müssen, wobei der Datensatz mit der höchsten Version im Service gespeichert wird. Um diesen Szenarien gerecht zu werden, bietet Amazon Timestream for Live

Analytics die Möglichkeit, Daten zu ändern. Upsert ist eine Operation, die einen Datensatz in das System einfügt, wenn der Datensatz nicht existiert, oder den Datensatz aktualisiert, wenn einer existiert. Wenn der Datensatz aktualisiert wird, wird er idempotent aktualisiert.

Es gibt keinen Vorgang zum Löschen auf Datensatzebene. Tabellen und Datenbanken können jedoch gelöscht werden.

Daten in den Speicher und den Magnetspeicher schreiben

Amazon Timestream for Live Analytics bietet die Möglichkeit, Daten direkt in den Speicher und den Magnetspeicher zu schreiben. Der Speicherspeicher ist für Datenschreibvorgänge mit hohem Durchsatz optimiert, und der Magnetspeicher ist für Schreibvorgänge von spät eintreffenden Daten mit niedrigerem Durchsatz optimiert.

Zu spät eintreffende Daten sind Daten, deren Zeitstempel vor der aktuellen Uhrzeit liegt und sich außerhalb der Aufbewahrungsfrist des Speicherspeichers befindet. Sie müssen die Möglichkeit, spät eintreffende Daten in den Magnetspeicher zu schreiben, ausdrücklich aktivieren, indem Sie Magnetspeicher-Schreibvorgänge für die Tabelle aktivieren. `MagneticStoreRejectedDataLocation`Wird auch definiert, wenn eine Tabelle erstellt wird. Um in den Magnetspeicher schreiben zu können, `WriteRecords` müssen Aufrufer von über `S3:PutObject` Berechtigungen für den S3-Bucket verfügen, der `MagneticStoreRejectedDataLocation` bei der Tabellenerstellung angegeben wurde. Weitere Informationen finden Sie unter CreateTableWriteRecords, und PutObject.

Schreiben von Daten mit Einzelmesswerten und Datensätzen mit mehreren Messwerten

Amazon Timestream for Live Analytics bietet die Möglichkeit, Daten mithilfe von zwei Arten von Datensätzen zu schreiben, nämlich Einzelmessdatensätzen und Datensätzen mit mehreren Messwerten.

Datensätze mit einer einzigen Messung

Einzelmessdatensätze ermöglichen es Ihnen, eine einzelne Messgröße pro Datensatz zu senden. Wenn Daten mit diesem Format an Timestream for Live Analytics gesendet werden, erstellt Timestream for Live Analytics eine Tabellenzeile pro Datensatz. Das heißt, wenn ein Gerät 4 Messwerte ausgibt und jede Metrik als Einzelmessdatensatz gesendet wird, erstellt Timestream for Live Analytics 4 Zeilen in der Tabelle, um diese Daten zu speichern, und die Geräteattribute werden für jede Zeile wiederholt. Dieses Format wird in Fällen empfohlen, in denen Sie eine einzelne Metrik von einer Anwendung aus überwachen möchten oder wenn Ihre Anwendung nicht mehrere Metriken gleichzeitig ausgibt.

Datensätze mit mehreren Messwerten

Bei Datensätzen mit mehreren Kennzahlen können Sie mehrere Kennzahlen in einer einzigen Tabellenzeile speichern, anstatt eine Kennzahl pro Tabellenzeile zu speichern. Multi-Measure-Aufzeichnungen ermöglichen es Ihnen daher, Ihre vorhandenen Daten mit minimalen Änderungen aus relationalen Datenbanken zu Amazon Timestream for Live Analytics zu migrieren.

Sie können auch mehr Daten in einer einzigen Schreibanforderung stapeln als Einzeldatensätze. Dies erhöht den Durchsatz und die Leistung beim Schreiben von Daten und senkt auch die Kosten für Datenschreibvorgänge. Das liegt daran, dass Amazon Timestream for Live Analytics durch die Bündelung von mehr Daten in einer einzigen Schreibanforderung (falls zutreffend) mehr wiederholbare Daten identifizieren kann und für wiederholte Daten nur einmal Gebühren berechnet werden.

Themen

Datensätze mit mehreren Messwerten

Bei Datensätzen mit mehreren Messwerten können Sie Ihre Zeitreihendaten in einem kompakteren Format im Speicher und Magnetspeicher speichern, wodurch die Kosten für die Datenspeicherung gesenkt werden. Außerdem eignet sich der kompakte Datenspeicher für das Schreiben einfacherer Abfragen für den Datenabruf, verbessert die Abfrageleistung und senkt die Kosten von Abfragen.

Darüber hinaus unterstützen Datensätze mit mehreren Messwerten auch den TIMESTAMP Datentyp zum Speichern von mehr als einem Zeitstempel in einem Zeitreihendatensatz. TIMESTAMPAttribute in einem Datensatz mit mehreren Kennzahlen unterstützen Zeitstempel in der future oder Vergangenheit. Datensätze mit mehreren Kennzahlen tragen somit zur Verbesserung der Leistung, der Kosten und der Einfachheit von Abfragen bei und bieten mehr Flexibilität beim Speichern verschiedener Arten von korrelierten Kennzahlen.

Vorteile

Im Folgenden sind die Vorteile der Verwendung von Datensätzen mit mehreren Kennzahlen aufgeführt.

- Leistung und Kosten — Mit Datensätzen mit mehreren Messwerten können Sie mehrere Zeitreihenwerte in einer einzigen Schreibanforderung schreiben. Dies erhöht den Schreibdurchsatz

und reduziert auch die Schreibkosten. Mit Datensätzen mit mehreren Messwerten können Sie Daten kompakter speichern, wodurch die Kosten für die Datenspeicherung gesenkt werden. Die kompakte Datenspeicherung von Datensätzen mit mehreren Messwerten führt dazu, dass weniger Daten durch Abfragen verarbeitet werden. Dadurch soll die allgemeine Abfrageleistung verbessert und die Abfragekosten gesenkt werden.

- Einfache Abfrage — Bei Datensätzen mit mehreren Kennzahlen müssen Sie keine komplexen allgemeinen Tabellenausdrücke (CTEs) in eine Abfrage schreiben, um mehrere Kennzahlen mit demselben Zeitstempel zu lesen. Das liegt daran, dass die Kennzahlen als Spalten in einer einzigen Tabellenzeile gespeichert werden. Datensätze mit mehreren Kennzahlen ermöglichen daher das Schreiben einfacherer Abfragen.
- Flexibilität bei der Datenmodellierung — Sie können future Zeitstempel in Timestream for Live Analytics schreiben, indem Sie den TIMESTAMP Datentyp und Datensätze mit mehreren Messwerten verwenden. Ein Datensatz mit mehreren Kennzahlen kann zusätzlich zum Zeitfeld in einem Datensatz mehrere Attribute des TIMESTAMP Datentyps haben. TIMESTAMPAttribute können in einem Datensatz mit mehreren Kennzahlen Zeitstempel in der future oder der Vergangenheit haben und sich wie das Zeitfeld verhalten, außer dass Timestream for Live Analytics die Werte vom Typ TIMESTAMP in einem Datensatz mit mehreren Kennzahlen nicht indexiert.

Anwendungsfälle

Sie können Datensätze mit mehreren Messwerten für jede Zeitreihenanwendung verwenden, die zu einem bestimmten Zeitpunkt mehr als eine Messung mit demselben Gerät generiert. Im Folgenden finden Sie einige Beispielanwendungen.

- Eine Video-Streaming-Plattform, die Hunderte von Metriken zu einem bestimmten Zeitpunkt generiert.
- Medizinische Geräte, die Messungen wie Blutsauerstoffgehalt, Herzfrequenz und Puls generieren.
- Industrieanlagen wie Ölplattformen, die Messwerte, Temperatur- und Wettersensoren erzeugen.
- Andere Anwendungen, die auf einem oder mehreren Microservices basieren.

Beispiel: Überwachung der Leistung und des Zustands einer Video-Streaming-Anwendung

Stellen Sie sich eine Videostreaming-Anwendung vor, die auf 200 EC2 Instanzen ausgeführt wird. Sie möchten Amazon Timestream for Live Analytics verwenden, um die von der Anwendung ausgegebenen Metriken zu speichern und zu analysieren, damit Sie die Leistung und den

Zustand Ihrer Anwendung verstehen, Anomalien schnell identifizieren, Probleme lösen und Optimierungsmöglichkeiten entdecken können.

Wir werden dieses Szenario mit Datensätzen mit Einzelkennzahlen und Datensätzen mit mehreren Messwerten modellieren und dann beide Ansätze vergleichen/gegenüberstellen. Für jeden Ansatz gehen wir von den folgenden Annahmen aus.

- Jede EC2 Instanz gibt vier Messwerte (video_startup_time, rebuffering_ratio, video_playback_failures und average_frame_rate) und vier Dimensionen (device_id, device_type, os_version und region) pro Sekunde aus.
- Sie möchten Daten für 6 Stunden im Speicher und 6 Monate für Daten im Magnetspeicher speichern.
- Um Anomalien zu identifizieren, haben Sie 10 Abfragen eingerichtet, die jede Minute ausgeführt werden, um ungewöhnliche Aktivitäten der letzten Minuten zu identifizieren. Sie haben außerdem ein Dashboard mit acht Widgets erstellt, die die Daten der letzten 6 Stunden anzeigen, sodass Sie Ihre Anwendung effektiv überwachen können. Auf dieses Dashboard können fünf Benutzer gleichzeitig zugreifen und es wird jede Stunde automatisch aktualisiert.

Verwendung von Datensätzen für einzelne Messwerte

Datenmodellierung: Bei Einzelmessdatensätzen erstellen wir einen Datensatz für jede der vier Messungen (Videostartzeit, Pufferrate, Fehler bei der Videowiedergabe und durchschnittliche Bildrate). Jeder Datensatz wird die vier Dimensionen (device_id, device_type, os_version und region) und einen Zeitstempel haben.

Schreibvorgänge: Wenn Sie Daten in Amazon Timestream for Live Analytics schreiben, werden die Datensätze wie folgt aufgebaut.

```
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);
    }
  }
```

Wenn Sie Einzelmessdatensätze speichern, werden die Daten logisch wie folgt dargestellt.

Zeit	Gerät_ID	Gerätetyp	os_version	Region	measure_name	Messwert::bigint	Messwert::doppelt
07.09.2021 21:48:44.000000000	12345678	iPhone 11	14,8	us-east-1	Startzeit des Videos	200	
07.09.2021 21:48:44,000000000	12345678	iPhone 11	14,8	us-east-1	Rebuffering_ratio		0.5
07.09.2021 21:48:44,000000000	12345678	iPhone 11	14,8	us-east-1	Fehler bei der Videowiedergabe	0	
07.09.2021 21:48:44,	12345678	iPhone 11	14,8	us-east-1	durchschnittliche Bildrate		0,85

Zeit	Gerät_ID	Gerätetyp	os_version	Region	measure_name	Messwert::bigint	Messwert::doppelt
000000000							
07.09.2021 21:53:44,000000000	12345678	iPhone 11	14,8	us-east-1	Startzeit des Videos	500	
07.09.2021 21:53:44,000000000	12345678	iPhone 11	14,8	us-east-1	Rebuffering_ratio		1.5
07.09.2021 21:53:44,000000000	12345678	iPhone 11	14,8	us-east-1	Fehler bei der Videowiedergabe	10	
07.09.2021 21:53:44,000000000	12345678	iPhone 11	14,8	us-east-1	durchschnittliche Bildrate		0.2

Abfragen: Sie können wie folgt eine Abfrage schreiben, die alle Datenpunkte mit demselben Zeitstempel abruft, die in den letzten 15 Minuten empfangen wurden.

```
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
```

Workload-Kosten: Die Kosten für diesen Workload werden auf 373,23$ pro Monat bei Datensätzen mit nur einer Kennzahl geschätzt

Verwendung von Datensätzen mit mehreren Kennzahlen

Datenmodellierung: Bei Datensätzen mit mehreren Messwerten erstellen wir einen Datensatz, der alle vier Kennzahlen (Videostartzeit, Pufferrate, Fehler bei der Videowiedergabe und durchschnittliche Bildrate), alle vier Dimensionen (device_id, device_type, os_version und Region) und einen Zeitstempel enthält.

Schreibvorgänge: Wenn Sie Daten in Amazon Timestream for Live Analytics schreiben, werden die Datensätze wie folgt aufgebaut.

```
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);
    }
  }
```

Wenn Sie Datensätze mit mehreren Kennzahlen speichern, werden die Daten logisch wie folgt dargestellt.

Zeit	Gerät_ID	Gerätety	os_versio n	Region	measure ame	Startzeit des Videos	Rebuffer ng_ratio	Fehler bei der Videowie ergabe	durchschn ittliche Bildrate
07.09.20 1 21:48:44 0000000	1234567	iPhone 11	14,8	us-east-1	Video-Met riken	200	0.5	0	0,85
07.09.20 1 21:53:44 0000000	1234567	iPhone 11	14,8	us-east-1	Video-Met riken	500	1.5	10	0.2

Abfragen: Sie können wie folgt eine Abfrage schreiben, die alle Datenpunkte mit demselben Zeitstempel abruft, die in den letzten 15 Minuten empfangen wurden.

```
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)
```

Workload-Kosten: Bei Datensätzen mit mehreren Kennzahlen werden die Workload-Kosten auf 127,43$ geschätzt.

> **Note**
>
> In diesem Fall werden durch die Verwendung von Datensätzen mit mehreren Kennzahlen die geschätzten monatlichen Gesamtausgaben um das 2,5-Fache reduziert, wobei die Kosten für das Schreiben von Daten um das 3,3-fache, die Speicherkosten um das 3,3-fache und die Abfragekosten um das 1,2-fache reduziert werden.

Daten mit einem Zeitstempel schreiben, der in der Vergangenheit oder in der future existiert

Timestream for Live Analytics bietet mithilfe verschiedener Mechanismen die Möglichkeit, Daten mit einem Zeitstempel zu schreiben, der außerhalb des Aufbewahrungszeitfensters für den Speicherspeicher liegt.

- Schreibvorgänge im Magnetspeicher — Sie können Daten, die spät eintreffen, mithilfe von Magnetspeicher-Schreibvorgängen direkt in den Magnetspeicher schreiben. Um Magnetspeicher-Schreibvorgänge verwenden zu können, müssen Sie zunächst Magnetspeicher-Schreibvorgänge für eine Tabelle aktivieren. Anschließend können Sie Daten mit demselben Mechanismus in die Tabelle aufnehmen, der für das Schreiben von Daten in den Speicherspeicher verwendet wurde. Amazon Timestream for Live Analytics schreibt die Daten anhand ihres Zeitstempels automatisch in den Magnetspeicher.

  > **Note**
  >
  > Die write-to-read Latenz für den Magnetspeicher kann bis zu 6 Stunden betragen, im Gegensatz zum Schreiben von Daten in den Speicherspeicher, wo die write-to-read Latenz im Bereich unter einer Sekunde liegt.

- TIMESTAMPDatentyp für Kennzahlen — Sie können den TIMESTAMP Datentyp verwenden, um Daten aus der Vergangenheit, Gegenwart oder future zu speichern. Ein Datensatz mit mehreren Kennzahlen kann zusätzlich zum Zeitfeld in einem Datensatz mehrere Attribute des TIMESTAMP Datentyps haben. TIMESTAMPAttribute können in einem Datensatz mit mehreren Kennzahlen Zeitstempel in der future oder der Vergangenheit haben und sich wie das Zeitfeld verhalten, außer dass Timestream for Live Analytics die Werte vom Typ TIMESTAMP in einem Datensatz mit mehreren Kennzahlen nicht indexiert.

> **Note**
>
> Der TIMESTAMP Datentyp wird nur für Datensätze mit mehreren Kennzahlen unterstützt.

Eventuelle Konsistenz bei Lesevorgängen

Timestream for Live Analytics unterstützt die Semantik der eventuellen Konsistenz bei Lesevorgängen. Wenn Sie also Daten unmittelbar nach dem Schreiben eines Datenstapels in Timestream for Live Analytics abfragen, spiegeln die Abfrageergebnisse möglicherweise nicht die Ergebnisse eines kürzlich abgeschlossenen Schreibvorgangs wider. Wenn Sie diese Abfrageanfragen nach kurzer Zeit wiederholen, sollten die Ergebnisse die neuesten Daten zurückgeben.

Batching schreibt mit WriteRecords API

Amazon Timestream for Live Analytics ermöglicht es Ihnen, Datenpunkte aus einer einzelnen Zeitreihe und/oder Datenpunkte aus vielen Serien in einer einzigen Schreibanforderung zu schreiben. Das Stapeln mehrerer Datenpunkte in einem einzigen Schreibvorgang ist aus Leistungs- und Kostensicht vorteilhaft. Weitere Informationen finden Sie Schreibt im Abschnitt Zählerfassung und Preisgestaltung.

> **Note**
>
> Ihre Schreibanfragen an Timestream for Live Analytics können gedrosselt werden, da Timestream for Live Analytics skaliert wird, um sich an die Datenaufnahmeanforderungen Ihrer Anwendung anzupassen. Wenn Ihre Anwendungen auf Drosselungsausnahmen stoßen, müssen Sie weiterhin Daten mit demselben (oder einem höheren) Durchsatz senden, damit Timestream for Live Analytics automatisch an die Anforderungen Ihrer Anwendung angepasst werden kann.

Batch-Laden

Mit Batch-Load für Amazon Timestream for LiveAnalytics können Sie in Amazon S3 gespeicherte CSV Dateien stapelweise in Timestream aufnehmen. Mit dieser neuen Funktion können Sie Ihre Daten in Timestream speichern, LiveAnalytics ohne sich auf andere Tools verlassen oder benutzerdefinierten Code schreiben zu müssen. Sie können Batch-Load verwenden, um Daten mit

flexiblen Wartezeiten aufzufüllen, z. B. Daten, die nicht sofort für Abfragen oder Analysen benötigt werden.

Sie können Batch-Load-Aufgaben erstellen, indem Sie die AWS Management Console AWS CLI, und die verwenden. AWS SDKs Weitere Informationen finden Sie unter Verwenden von Batch-Load mit der Konsole, Verwenden von Batch-Load mit AWS CLI und Verwenden von Batch-Load mit AWS SDKs.

Weitere Informationen zum Batch-Laden finden Sie unterVerwenden von Batch-Load in Timestream für LiveAnalytics.

Wählen Sie zwischen WriteRecords API Operation und Batch-Load

Mit diesem WriteRecords API Vorgang können Sie Ihre Streaming-Zeitreihendaten so in Timestream schreiben, LiveAnalytics wie sie von Ihrem System generiert werden. Auf diese WriteRecords Weise können Sie kontinuierlich einen einzelnen Datenpunkt oder kleinere Datenstapel in Echtzeit aufnehmen. Timestream for LiveAnalytics bietet Ihnen ein flexibles Schema, das die Spaltennamen und Datentypen für Ihren Timestream für LiveAnalytics Tabellen auto erkennt, basierend auf den Dimensionsnamen und Datentypen der Datenpunkte, die Sie beim Aufrufen von Schreibvorgängen in die Datenbank angeben.

Im Gegensatz dazu ermöglicht Batch-Load die robuste Aufnahme von Batch-Zeitreihendaten aus Quelldateien (CSVDateien) in Timestream for LiveAnalytics unter Verwendung eines von Ihnen definierten Datenmodells. Einige Beispiele für die Verwendung von Batch-Load mit einer Quelldatei sind der Import von Zeitreihendaten in großen Mengen für die Evaluierung von Timestream im LiveAnalytics Rahmen eines Machbarkeitsnachweises, der Import von Zeitreihendaten in großen Mengen von einem IoT-Gerät, das einige Zeit offline war, und die Migration historischer Zeitreihendaten von Amazon S3 zu Timestream for. LiveAnalytics Informationen zum Batch-Laden finden Sie unter. Verwenden von Batch-Load in Timestream für LiveAnalytics

Beide Lösungen sind sicher, zuverlässig und leistungsstark.

Verwenden Sie, WriteRecords wenn:

- Streaming kleinerer Datenmengen (weniger als 10 MB) pro Anfrage.
- Bestehende Tabellen werden aufgefüllt.
- Daten aus einem Protokollstream aufnehmen.
- Durchführung von Analysen in Echtzeit.
- Geringere Latenz erforderlich.

Verwenden Sie Batch-Load in folgenden Fällen:

- Aufnahme größerer Datenmengen, die ihren Ursprung in Amazon S3 haben, in CSV Dateien. Weitere Informationen zu Limits finden Sie unter Kontingente.
- Auffüllen neuer Tabellen, z. B. im Fall einer Datenmigration.
- Anreicherung von Datenbanken mit historischen Daten (Aufnahme in neue Tabellen).
- Sie haben Quelldaten, die sich langsam oder gar nicht ändern.
- Sie haben flexible Wartezeiten, da eine Batch-Ladeaufgabe möglicherweise den Status „Ausstehend“ hat, bis Ressourcen verfügbar sind, insbesondere wenn Sie eine sehr große Datenmenge laden. Das Batch-Load eignet sich für Daten, die nicht sofort für Abfragen oder Analysen verfügbar sein müssen, um mehr Klarheit zu schaffen.

Speicher

Timestream for Live Analytics speichert und organisiert Ihre Zeitreihendaten, um die Bearbeitungszeit von Abfragen zu optimieren und die Speicherkosten zu senken. Es bietet Datenspeicher-Tiering und unterstützt zwei Speicherebenen: einen Speicherspeicher und einen Magnetspeicher. Der Speicherspeicher ist für Datenschreibvorgänge mit hohem Durchsatz und schnelle point-in-time Abfragen optimiert. Der Magnetspeicher ist für einen geringeren Durchsatz, spät eintreffende Datenschreibvorgänge, langfristige Datenspeicherung und schnelle analytische Abfragen optimiert.

Timestream for Live Analytics gewährleistet die Haltbarkeit Ihrer Daten, indem es Ihre Speicher- und Magnetspeicherdaten automatisch in verschiedenen Availability Zones innerhalb einer einzigen repliziert. AWS-Region Alle Ihre Daten werden auf die Festplatte geschrieben, bevor Ihre Schreibanforderung als abgeschlossen bestätigt wird.

Mit Timestream for Live Analytics können Sie Aufbewahrungsrichtlinien konfigurieren, um Daten vom Speicherspeicher in den Magnetspeicher zu verschieben. Wenn die Daten den konfigurierten Wert erreichen, verschiebt Timestream for Live Analytics die Daten automatisch in den Magnetspeicher. Sie können auch einen Aufbewahrungswert für den Magnetspeicher festlegen. Wenn Daten außerhalb des Magnetspeichers ablaufen, werden sie dauerhaft gelöscht.

Stellen Sie sich beispielsweise ein Szenario vor, in dem Sie den Speicherspeicher für Daten für eine Woche und den Magnetspeicher für Daten eines Jahres konfigurieren. Das Alter der Daten wird anhand des Zeitstempels berechnet, der dem Datenpunkt zugeordnet ist. Wenn die Daten im Speicher eine Woche alt werden, werden sie automatisch in den Magnetspeicher verschoben. Sie werden dann für ein Jahr im Magnetspeicher aufbewahrt. Wenn die Daten ein Jahr

alt werden, werden sie aus Timestream for Live Analytics gelöscht. Die Aufbewahrungswerte des Speicherspeichers und des Magnetspeichers definieren kumulativ die Zeitspanne, für die Ihre Daten in Timestream for Live Analytics gespeichert werden. Das bedeutet, dass für das obige Szenario die Daten ab dem Zeitpunkt des Eintreffens der Daten für einen Gesamtzeitraum von 1 Jahr und 1 Woche in Timestream for Live Analytics gespeichert werden.

> **Note**
>
> Wenn Sie die Aufbewahrungsdauer des Speichers oder Magnetspeichers aktualisieren, wird die Änderung der Aufbewahrungsdauer ab diesem Zeitpunkt wirksam. Wenn beispielsweise die Aufbewahrungsdauer des Speicherspeichers auf 2 Stunden festgelegt und dann durch Aktualisierung der Tabellenaufbewahrungsrichtlinien auf 24 Stunden geändert wurde, kann der Speicherspeicher Daten für 24 Stunden speichern, wird aber 22 Stunden nach dieser Änderung mit Daten für 24 Stunden gefüllt. Timestream for Live Analytics ruft keine Daten aus dem Magnetspeicher ab, um den Speicherspeicher zu füllen.

Um die Sicherheit Ihrer Zeitreihendaten zu gewährleisten, werden Ihre Daten in Timestream for Live Analytics standardmäßig immer verschlüsselt. Dies gilt für Daten während der Übertragung und im Ruhezustand. Darüber hinaus können Sie mit Timestream for Live Analytics vom Kunden verwaltete Schlüssel verwenden, um Ihre Daten im Magnetspeicher zu sichern. Weitere Informationen zu vom Kunden verwalteten Schlüsseln finden Sie unter AWS KMS keys.

Abfragen

Mit Timestream for Live Analytics können Sie auf einfache Weise Metriken für DevOps Sensordaten für IoT-Anwendungen und industrielle Telemetriedaten für die Gerätewartung sowie viele andere Anwendungsfälle speichern und analysieren. Die speziell entwickelte, adaptive Abfrage-Engine in Timestream for Live Analytics ermöglicht Ihnen den Zugriff auf Daten über mehrere Speicherstufen hinweg mit einer einzigen Anweisung. SQL Sie greift transparent auf Daten auf verschiedenen Speicherebenen zu und kombiniert sie, ohne dass Sie den Speicherort der Daten angeben müssen. Sie können SQL Daten in Timestream for Live Analytics abfragen, um Zeitreihendaten aus einer oder mehreren Tabellen abzurufen. Sie können auf die Metadateninformationen für Datenbanken und Tabellen zugreifen. Timestream for Live Analytics unterstützt SQL auch integrierte Funktionen für Zeitreihenanalysen. Weitere Informationen finden Sie in der Referenz zur Abfragesprache Referenz.

Timestream for Live Analytics ist so konzipiert, dass es über eine vollständig entkoppelte Architektur für Datenaufnahme, Speicherung und Abfrage verfügt, bei der jede Komponente unabhängig von

anderen Komponenten skaliert werden kann (sodass sie praktisch unendlich skaliert werden kann, um den Anforderungen einer Anwendung gerecht zu werden). Das bedeutet, dass Timestream for Live Analytics nicht „umkippt“, wenn Ihre Anwendungen Hunderte von Terabyte an Daten pro Tag senden oder Millionen von Abfragen ausführen, die kleine oder große Datenmengen verarbeiten. Da Ihre Daten im Laufe der Zeit wachsen, bleibt die Abfragelatenz in Timestream for Live Analytics weitgehend unverändert. Dies liegt daran, dass die Timestream for Live Analytics-Abfragearchitektur enorme Mengen an Parallelität nutzen kann, um größere Datenmengen zu verarbeiten und automatisch zu skalieren, um den Anforderungen einer Anwendung an den Abfragedurchsatz gerecht zu werden.

Datenmodell

Timestream unterstützt zwei Datenmodelle für Abfragen: das flache Modell und das Zeitreihenmodell.

> **Note**
>
> Daten in Timestream werden mithilfe des flachen Modells gespeichert und es ist das Standardmodell für die Abfrage von Daten. Das Zeitreihenmodell ist ein Abfragezeitkonzept und wird für Zeitreihenanalysen verwendet.

- Flaches Modell
- Zeitreihenmodell

Flaches Modell

Das flache Modell ist das Standarddatenmodell von Timestream für Abfragen. Es stellt Zeitreihendaten in einem tabellarischen Format dar. Die Dimensionsnamen, die Zeit, die Kennzahlnamen und die Kennzahlwerte werden als Spalten angezeigt. Jede Zeile in der Tabelle ist ein atomarer Datenpunkt, der einer Messung zu einem bestimmten Zeitpunkt innerhalb einer Zeitreihe entspricht. Für Timestream-Datenbanken, -Tabellen und -Spalten gelten einige Einschränkungen bei der Benennung. Diese werden unter beschrieben. Service Limits

Die folgende Tabelle zeigt ein anschauliches Beispiel dafür, wie Timestream Daten speichert, die die CPU Auslastung, Speicherauslastung und Netzwerkaktivität von EC2 Instances darstellen, wenn die Daten als Einzeldatensatz gesendet werden. In diesem Fall handelt es sich bei den Dimensionen um die Region, die Verfügbarkeitszone, die virtuelle private Cloud und die Instanz IDs der Instanzen. EC2 Bei den Kennzahlen handelt es sich um die CPU Auslastung, die Speicherauslastung

und die eingehenden Netzwerkdaten für die EC2 Instanzen. Die Spalten region, az, vpc und instance_id enthalten die Dimensionswerte. Die Spalte time enthält den Zeitstempel für jeden Datensatz. Die Spalte measure_name enthält die Namen der Kennzahlen, die durch cpu-utilization, memory_utilization und network_bytes_in dargestellt werden. Die Spalten measure_value: :double enthalten Messwerte, die doppelt ausgegeben werden (z. B. Auslastung und Speicherauslastung). CPU Die Spalte measure_value: :bigint enthält Messungen, die als ganze Zahlen ausgegeben werden, z. B. die eingehenden Netzwerkdaten.

Zeit	Region	az	vpc	instance_id	measure_name	Messwert::doppelt	Messwert::bigint
2019-12-04 19:00:00.000 000000	us-east-1	us-ost-1d	vpc-1a2b3c4d	i-1234567890abcdef0	CPU-Auslastung	35,0	Null
2019-12-04 19:00:01.000 000000	us-east-1	us-ost-1d	vpc-1a2b3c4d	i-1234567890abcdef0	CPU-Auslastung	38,2	Null
2019-12-04 19:00:02.000 000000	us-east-1	us-ost-1d	vpc-1a2b3c4d	i-1234567890abcdef0	CPU-Auslastung	45,3	Null
2019-12-04 19:00:00.000 000000	us-east-1	us-ost-1d	vpc-1a2b3c4d	i-1234567890abcdef0	memory_utilization	54,9	Null
2019-12-04 19:00:01.	us-east-1	us-ost-1d	vpc-1a2b3c4d	i-1234567890abcdef0	memory_utilization	42,6	Null

Zeit	Region	az	vpc	instance_id	measure_name	Messwert::doppelt	Messwert::bigint
000 000000							
2019-12-04 19:00:02.000 000000	us-east-1	us-ost-1d	vpc-1a2b3c4d	i-1234567890abcdef0	memory_utilization	33,3	Null
2019-12-04 19:00:00.000 000000	us-east-1	us-ost-1d	vpc-1a2b3c4d	i-1234567890abcdef0	Netzwerk-Bytes	34.400	Null
2019-12-04 19:00:01.000 000000	us-east-1	us-ost-1d	vpc-1a2b3c4d	i-1234567890abcdef0	Netzwerk-Bytes	1.500	Null
2019-12-04 19:00:02.000 000000	us-east-1	us-ost-1d	vpc-1a2b3c4d	i-1234567890abcdef0	Netzwerk-Bytes	6 000	Null

Die folgende Tabelle zeigt ein anschauliches Beispiel dafür, wie Timestream Daten speichert, die die CPU Auslastung, Speicherauslastung und Netzwerkaktivität von EC2 Instances darstellen, wenn die Daten als Datensatz mit mehreren Messwerten gesendet werden.

Zeit	Region	az	vpc	instance_id	measure_name	cpu_utilization	memory_utilization	Netzwerk-Bytes
2019-12-04 19:00:00.000 000000	us-east-1	us-ost-1d	vpc-1a2b3c4d	i-1234567890abcde0	Kennzahlen	35,0	54,9	34.400
2019-12-04 19:00:01.000 000000	us-east-1	us-ost-1d	vpc-1a2b3c4d	i-1234567890abcde0	Kennzahlen	38,2	42,6	1.500
2019-12-04 19:00:02.000 000000	us-east-1	us-ost-1d	vpc-1a2b3c4d	i-1234567890abcde0	Kennzahlen	45,3	33,3	6.600

Zeitreihenmodell

Das Zeitreihenmodell ist ein Abfragezeitkonstrukt, das für Zeitreihenanalysen verwendet wird. Es stellt Daten als eine geordnete Folge von Paaren (Zeit, Messwert) dar. Timestream unterstützt Zeitreihenfunktionen wie Interpolation, damit Sie die Lücken in Ihren Daten schließen können. Um diese Funktionen verwenden zu können, müssen Sie Ihre Daten mithilfe von Funktionen wie create_time_series in das Zeitreihenmodell konvertieren. Weitere Einzelheiten finden Sie unter. Referenz zur Abfragesprache

Anhand des vorherigen Beispiels der EC2 Instanz werden hier die CPU Nutzungsdaten als Zeitreihe ausgedrückt.

Region	az	vpc	instance_id	cpu_utilization
us-east-1	us-ost-1d	vpc-1a2b3c4d	i-1234567890abcdef0	[{Zeit: 2019-12-04 19:00:00.

Region	az	vpc	instance_id	cpu_utilization
				000 000000, Wert: 35}, {Zeit: 2019-12-04 19:00:01.000 000000, Wert: 38,2}, {Zeit: 2019-12-04 19:00:02.000 000000, Wert: 45,3}]

Geplante Abfragen

Die Funktion für geplante Abfragen in Amazon Timestream for Live Analytics ist eine vollständig verwaltete, serverlose und skalierbare Lösung für die Berechnung und Speicherung von Aggregaten, Rollups und anderen Formen vorverarbeiteter Daten, die normalerweise für operative Dashboards, Geschäftsberichte, Ad-hoc-Analysen und andere Anwendungen verwendet werden. Geplante Abfragen machen Echtzeitanalysen leistungsfähiger und kostengünstiger, sodass Sie zusätzliche Erkenntnisse aus Ihren Daten gewinnen und weiterhin bessere Geschäftsentscheidungen treffen können.

Weitere Informationen zu geplanten Abfragen finden Sie unter. Verwenden von geplanten Abfragen in Timestream für LiveAnalytics

Timestream-Recheneinheit () TCU

Amazon Timestream for Live Analytics misst die Rechenkapazität, die Ihnen für Ihre Abfrageanforderungen zugewiesen wurde, in der Timestream-Recheneinheit ()TCU. Eine TCU besteht aus 4 vCPUs und 16 GB Arbeitsspeicher. Wenn Sie Abfragen in Timestream for Live Analytics ausführen, ordnet der Service auf der Grundlage der Komplexität Ihrer Abfragen und der Menge der verarbeiteten Daten eine Zuweisung nach TCUs Bedarf zu. Die Anzahl der DatenTCUs, die eine Abfrage verbraucht, bestimmt die damit verbundenen Kosten.

> **Note**
>
> Alle AWS-Konten , die nach dem 29. April 2024 in den Service integriert sind, werden standardmäßig TCUs für die Preisgestaltung von Abfragen verwendet.

In diesem Thema

MaxQuery TCU

Diese Einstellung gibt die maximale Anzahl von Recheneinheiten an, die der Service zu einem beliebigen Zeitpunkt für die Bearbeitung Ihrer Anfragen verwendet. Um Abfragen auszuführen, müssen Sie die Mindestkapazität auf 4 festlegenTCUs. Sie können die maximale Anzahl von TCUs in Vielfachen von 4 festlegen, z. B. 4, 8, 16, 32 usw. Ihnen werden nur die Rechenressourcen in Rechnung gestellt, die Sie für Ihren Workload verwenden. Zum Beispiel, wenn Sie das Maximum TCUs auf 128 setzen, aber durchweg nur 8 verwendenTCUs. Ihnen wird nur die Dauer in Rechnung gestellt, in der Sie die 8 genutzt habenTCUs. Die Standardeinstellung `MaxQueryTCU` in Ihrem Konto ist auf 200 festgelegt. Sie können einen Wert `MaxQueryTCU` zwischen 4 und 1000 einstellen, indem Sie die UpdateAccountSettingsAPIOperation AWS Management Console oder zusammen mit der Option AWS SDK oder verwenden AWS CLI.

Wir empfehlen die Einstellung `MaxQueryTCU` für Ihr Konto. Die Festlegung eines TCU Höchstlimits trägt zur Kostenkontrolle bei, da die Anzahl der Recheneinheiten begrenzt wird, die der Service für Ihre Abfrage-Workloads verwenden kann. Auf diese Weise können Sie Ihre Ausgaben für Abfragen besser vorhersagen und verwalten.

Abrechnung für TCU

Jede Abrechnung TCU erfolgt auf Stundenbasis mit einer Genauigkeit pro Sekunde und für mindestens 30 Sekunden. Die Nutzungseinheit dieser Recheneinheiten ist -Stunde. TCU

Wenn Sie Abfragen ausführen, wird Ihnen die während der Ausführung der Abfrage TCUs genutzte Zeit in Rechnung gestellt, gemessen in TCU Stunden. Beispielsweise:

- Ihre Arbeitslast beläuft sich auf 20 TCUs für 3 Stunden. Ihnen werden 60 Stunden (20 TCUs x TCU 3 Stunden) in Rechnung gestellt.
- Ihr Workload umfasst 10 TCUs für 30 Minuten und dann 20 TCUs für die nächsten 30 Minuten. Ihnen werden 15 Stunden (10 TCUs x 0,5 TCU Stunden + 20 TCUs x 0,5 Stunden) in Rechnung gestellt.

Die Preise pro TCU Stunde variieren je nach. AWS-Region Weitere Informationen finden Sie in den Amazon Timestream-Preisen. Wenn Ihre Arbeitslast wächst, skaliert der Service die Rechenkapazität automatisch bis zur angegebenen TCU Höchstgrenze (`MaxQueryTCU`), um eine gleichbleibende Leistung zu gewährleisten. Diese `MaxQueryTCU` Einstellung dient als Obergrenze für die Rechenkapazität, auf die der Service skaliert werden kann. Diese Einstellung hilft Ihnen, die Anzahl der Rechenressourcen und damit deren Kosten zu kontrollieren.

Konfiguration TCU

Wenn Sie den Dienst einbinden, gilt für jeden AWS-Konto Dienst ein `MaxQueryTCU` Standardlimit von 200. Sie können dieses Limit jederzeit nach Bedarf aktualisieren, indem Sie die UpdateAccountSettingsAPIOperation AWS Management Console oder zusammen mit dem AWS SDK oder verwenden AWS CLI.

Wenn Sie sich nicht sicher sind, welche Werte Sie konfigurieren sollen, überprüfen Sie die `QueryTCU` Metrik für Ihr Konto. Diese Metrik ist in Amazon AWS Management Console und Amazon verfügbar CloudWatch. Diese Metrik bietet Einblick in die maximale Anzahl der TCUs verwendeten Daten mit einer Genauigkeit von Minute zu Minute. Stellen Sie den auf der Grundlage historischer Daten und Ihrer Einschätzung des future Wachstums so ein, `MaxQueryTCU` dass er Ihren Nutzungsspitzen Rechnung trägt. Wir empfehlen, einen Spielraum von mindestens 4 bis 16% TCUs über Ihrer Spitzennutzung zu haben. Wenn Ihr Spitzenwert `QueryTCU` in den letzten 30 Tagen beispielsweise 128 betrug, empfehlen wir einen Wert `MaxQueryTCU` zwischen 132 und 144.

Schätzung der benötigten Recheneinheiten

Recheneinheiten können Abfragen gleichzeitig verarbeiten. Beachten Sie die allgemeinen Richtlinien in der folgenden Tabelle, um die Anzahl der erforderlichen Recheneinheiten zu ermitteln:

Gleichzeitige Abfragen	TCUs
7	4
14	8
21	12

> **Note**
>
> - Dies sind allgemeine Richtlinien, und die tatsächliche Anzahl der benötigten Recheneinheiten hängt von mehreren Faktoren ab, wie z. B.:
> - Die effektive Parallelität von Abfragen.
> - Abfragemuster.
> - Die Anzahl der gescannten Partitionen.
> - Andere Workload-spezifische Merkmale.
> - Diese Richtlinie bezieht sich auf Abfragen, bei denen Daten der letzten Minuten bis zu einer Stunde gescannt werden und die bewährten Methoden für Abfragen und die Datenmodellierung von Timestream eingehalten werden.
> - Überwachen Sie die Leistung Ihrer Anwendung und die `QueryTCU` Metrik, um die Recheneinheiten nach Bedarf anzupassen.

Wann sollte erhöht werden MaxQuery TCU

In den folgenden Szenarien sollten Sie eine Erhöhung der Werte `MaxQueryTCU` in Betracht ziehen:

- Ihr maximaler Abfrageverbrauch nähert sich oder erreicht den aktuell konfigurierten Höchstwert für AbfragenTCU. Wir empfehlen, den maximalen Abfragewert TCU mindestens 4-16 TCUs höher einzustellen als der Spitzenverbrauch.

- Ihre Abfragen geben einen 4xx-Fehler zurück, bei dem die Meldung MaxQuery TCU überschritten wurde. Wenn Sie mit einer geplanten Erhöhung Ihrer Arbeitslast rechnen, sollten Sie die konfigurierte maximale Abfrage erneut aufrufen und entsprechend anpassen. TCU

Wann sollten Sie verringern MaxQuery TCU

In den folgenden Szenarien sollten Sie eine Verringerung der Werte `MaxQueryTCU` in Betracht ziehen:

- Ihr Workload weist ein vorhersehbares und stabiles Nutzungsmuster auf, und Sie haben ein gutes Verständnis für Ihre Anforderungen an die Computernutzung. Wenn Sie die maximale Anzahl TCU an Abfragen auf einen Wert zwischen 4 und 16% TCU über Ihrem Spitzenverbrauch senken, können Sie eine unbeabsichtigte Nutzung und damit verbundene Kosten vermeiden. Sie können den Wert mithilfe der Operation ändern. UpdateAccountSettingsAPI
- Die Spitzenauslastung Ihres Workloads hat im Laufe der Zeit abgenommen, entweder aufgrund von Änderungen in Ihrer Anwendung oder aufgrund von Benutzerverhaltensmustern. Eine Senkung des Maximums TCU kann dazu beitragen, unbeabsichtigte Kosten zu minimieren.

> **Note**
>
> Je nach Ihrer aktuellen Nutzung kann es bis zu 24 Stunden dauern, bis die Änderung des TCU Höchstlimits wirksam wird. Ihnen wird nur der Betrag in Rechnung gestelltTCUs, den Ihre Abfragen tatsächlich verbrauchen. Ein höheres maximales TCU Abfragelimit wirkt sich nicht auf Ihre Kosten aus, es sei denn, diese TCUs werden von Ihrem Workload aufgebraucht.

Überwachung der Nutzung mithilfe von CloudWatch Metriken

Um Ihre TCU Nutzung zu überwachen, bietet Timestream for Live Analytics die folgende CloudWatch Metrik:`QueryTCU`. Diese Metrik gibt die Anzahl der pro Minute verwendeten Recheneinheiten an und wird jede Minute ausgegeben. Sie können wählen, ob der maximale und der minimale Verbrauch pro Minute überwacht TCUs werden sollen. Sie können auch Alarme für diese Metrik einrichten, um Ihre Abfragekosten in Echtzeit zu verfolgen.

Grundlegendes zu Variationen bei der Nutzung von Recheneinheiten

Die Anzahl der Rechenressourcen, die für Ihre Abfragen benötigt werden, kann sich auf der Grundlage mehrerer Parameter entweder erhöhen oder verringern. Zum Beispiel Datenvolumen, Datenaufnahmemuster, Abfragelatenz, Abfrageform, Abfrageeffizienz und Abfragekombinationen, die Echtzeit- und Analyseabfragen verwenden. Diese Parameter können dazu führen, dass für Ihre Arbeitslast entweder höhere oder niedrigere TCU Einheiten erforderlich sind. In einem stabilen Zustand, in dem sich diese Parameter nicht ändern, stellen Sie möglicherweise fest, dass die Anzahl der für Ihre Arbeitslast erforderlichen Recheneinheiten abnimmt. Folglich kann dies Ihre monatlichen Kosten senken.

Wenn sich einer dieser Parameter in Ihrer Arbeitslast oder Ihren Daten ändert, kann sich außerdem die Anzahl der benötigten Recheneinheiten erhöhen. Wenn Timestream eine Abfrage empfängt, entscheidet Timestream je nach den Datenpartitionen, auf die die Abfrage zugreift, über die Anzahl der Rechenressourcen, um die Abfrage performant zu bearbeiten.

In regelmäßigen Abständen optimiert Timestream auf der Grundlage Ihrer Eingabe- und Abfragezugriffsmuster das Datenlayout. Timestream führt die Optimierung durch, indem Partitionen, auf die weniger zugegriffen wird, in einer einzigen Partition zusammengefasst oder eine Hotpartition aus Performance-Gründen in mehrere Partitionen aufgeteilt wird. Folglich kann die Rechenkapazität, die von derselben Abfrage verwendet wird, zu verschiedenen Zeitpunkten leicht variieren.

> Sie entscheiden sich dafür, die TCU Preisgestaltung für Ihre Anfragen zu verwenden
>
> Als bestehender Benutzer können Sie sich einmalig anmelden, um das Kostenmanagement zu verbessern und TCUs die Mindestanzahl der pro Abfrage gemessenen Byte zu entfernen. Sie können sich anmelden, indem Sie die UpdateAccountSettingsAPIOperation AWS Management Console oder zusammen mit der Option oder verwenden. AWS SDK AWS CLI Stellen Sie in der API Operation den `QueryPricingModel` Parameter auf ein`COMPUTE_UNITS`.
> Die Entscheidung für das rechnergestützte Preismodell ist eine unumkehrbare Änderung.

Zugreifen auf Timestream für LiveAnalytics

Sie können auf Timestream zugreifen, um die Konsole zu LiveAnalytics verwenden, CLI oder API Informationen zum Zugriff auf Timestream für LiveAnalytics finden Sie in den folgenden Abschnitten:

Themen

- Melden Sie sich an für eine AWS-Konto
- Erstellen eines Benutzers mit Administratorzugriff
- Stellen Sie Timestream für den Zugriff bereit LiveAnalytics
- Erteilen programmgesteuerten Zugriffs

Melden Sie sich an für eine AWS-Konto

Wenn Sie noch keine haben AWS-Konto, führen Sie die folgenden Schritte aus, um eine zu erstellen.

Um sich für eine anzumelden AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/die Anmeldung.
2. Folgen Sie den Online-Anweisungen.

   Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für eine anmelden AWS-Konto, Root-Benutzer des AWS-Kontoswird eine erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Administratorbenutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

AWS sendet Ihnen nach Abschluss des Anmeldevorgangs eine Bestätigungs-E-Mail. Du kannst jederzeit deine aktuellen Kontoaktivitäten einsehen und dein Konto verwalten, indem du zu https://aws.amazon.com/gehst und Mein Konto auswählst.

Erstellen eines Benutzers mit Administratorzugriff

Nachdem Sie sich für einen angemeldet haben AWS-Konto, sichern Sie Ihren Root-Benutzer des AWS-Kontos AWS IAM Identity Center, aktivieren und erstellen Sie einen Administratorbenutzer, sodass Sie den Root-Benutzer nicht für alltägliche Aufgaben verwenden.

Sichern Sie Ihre Root-Benutzer des AWS-Kontos

1. Melden Sie sich AWS Management Consoleals Kontoinhaber an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-Anmeldung Benutzerhandbuch zu.

2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für Ihren Root-Benutzer.

   Anweisungen finden Sie im Benutzerhandbuch unter Aktivieren eines virtuellen MFA Geräts für Ihren AWS-Konto IAM Root-Benutzer (Konsole).

Erstellen eines Benutzers mit Administratorzugriff

1. Aktivieren Sie IAM Identity Center.

   Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center Benutzerhandbuch.

2. Gewähren Sie einem Benutzer in IAM Identity Center Administratorzugriff.

   Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie unter Benutzerzugriff mit der Standardeinstellung konfigurieren IAM-Identity-Center-Verzeichnis im AWS IAM Identity Center Benutzerhandbuch.

Anmelden als Administratorbenutzer

- Um sich mit Ihrem IAM Identity Center-Benutzer anzumelden, verwenden Sie die Anmeldung, URL die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM Identity Center-Benutzer erstellt haben.

  Hilfe bei der Anmeldung mit einem IAM Identity Center-Benutzer finden Sie im AWS-Anmeldung Benutzerhandbuch unter Anmeldung beim AWS Zugangsportal.

Weiteren Benutzern Zugriff zuweisen

1. Erstellen Sie in IAM Identity Center einen Berechtigungssatz, der der bewährten Methode zur Anwendung von Berechtigungen mit den geringsten Rechten folgt.

   Anweisungen hierzu finden Sie unter Berechtigungssatz erstellen im AWS IAM Identity Center Benutzerhandbuch.

2. Weisen Sie Benutzer einer Gruppe zu und weisen Sie der Gruppe dann Single Sign-On-Zugriff zu.

Eine genaue Anleitung finden Sie unter Gruppen hinzufügen im AWS IAM Identity Center Benutzerhandbuch.

Stellen Sie Timestream für den Zugriff bereit LiveAnalytics

Die Berechtigungen, die für den Zugriff auf Timestream für erforderlich LiveAnalytics sind, wurden dem Administrator bereits erteilt. Anderen Benutzern sollten Sie mithilfe der folgenden Richtlinie Timestream für den LiveAnalytics Zugriff gewähren:

```
{
  "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**
>
> Weitere Informationen dazu dbqms finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für den Database Query Metadata Service. Informationen dazu

kms finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für den AWS Key Management Service.

Erteilen programmgesteuerten Zugriffs

Benutzer benötigen programmgesteuerten Zugriff, wenn sie mit AWS außerhalb des AWS Management Console interagieren möchten. Die Art und Weise, wie programmatischer Zugriff gewährt wird, hängt von der Art des Benutzers ab, der zugreift. AWS

Um Benutzern programmgesteuerten Zugriff zu gewähren, wählen Sie eine der folgenden Optionen.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
Mitarbeiteridentität (In IAM Identity Center verwaltete Benutzer)	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an das AWS CLI AWS SDKs, oder AWS APIs zu signieren.	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zu den AWS CLI finden Sie unter Konfiguration der AWS CLI zur Verwendung AWS IAM Identity Center im AWS Command Line Interface Benutzerhandbuch. • Informationen zu AWS SDKs Tools und AWS APIs finden Sie unter IAMIdentity Center-Authentifizierung im Referenzhandbuch AWS SDKsund im Tools-Referenzhandbuch.
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen	Folgen Sie den Anweisungen unter Verwenden temporärer Anmeldeinformationen

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
	an das AWS CLI AWS SDKs, oder AWS APIs zu signieren.	mit AWS Ressourcen im IAMBenutzerhandbuch.
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um programmatische Anfragen an das AWS CLI AWS SDKs, oder AWS APIs zu signieren.	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen dazu AWS CLI finden Sie unter Authentifizierung mithilfe von IAM Benutzeranmeldedaten im AWS Command Line Interface Benutzerhandbuch. • Informationen zu AWS SDKs und Tools finden Sie unter Authentifizieren mit langfristigen Anmeldeinformationen im Referenzhandbuch AWS SDKsund im Tools-Referenzhandbuch. • Weitere Informationen finden Sie unter Verwaltung von Zugriffsschlüsseln für IAM Benutzer im IAMBenutzerhandbuch. AWS APIs

Verwenden der Konsole

Sie können die AWS Management Console für Timestream Live Analytics verwenden, um Datenbanken und Tabellen zu erstellen, zu bearbeiten, zu löschen, zu beschreiben und aufzulisten. Sie können die Konsole auch verwenden, um Abfragen auszuführen.

Themen

Tutorial

In diesem Tutorial erfahren Sie, wie Sie eine Datenbank mit Beispieldatensätzen erstellen und Beispielabfragen ausführen. Die in diesem Tutorial verwendeten Beispieldatensätze werden häufig in IoT und DevOps Szenarien verwendet. Der IoT-Datensatz enthält Zeitreihendaten wie Geschwindigkeit, Standort und Ladung eines Lkw, um das Flottenmanagement zu rationalisieren und Optimierungsmöglichkeiten zu identifizieren. Der DevOps Datensatz enthält EC2 Instanzmetriken wie CPU Netzwerk- und Speicherauslastung, um die Leistung und Verfügbarkeit von Anwendungen zu verbessern. Hier finden Sie ein Video-Tutorial für die in diesem Abschnitt beschriebenen Anweisungen

Gehen Sie wie folgt vor, um eine Datenbank mit den Beispieldatensätzen zu erstellen und Beispielabfragen mithilfe der AWS Konsole auszuführen.

1. Öffnen Sie die AWS Konsole.
2. Wählen Sie im Navigationsbereich Datenbanken
3. Klicken Sie auf Datenbank erstellen.
4. Geben Sie auf der Seite „Datenbank erstellen“ Folgendes ein:
    - Konfiguration wählen — Wählen Sie Beispieldatenbank aus.
    - Name — Geben Sie einen Datenbanknamen Ihrer Wahl ein.
    - Wählen Sie Beispieldatensätze aus — Wählen Sie IoT und. DevOps
    - Klicken Sie auf Datenbank erstellen, um eine Datenbank zu erstellen, die zwei IoT-Tabellen enthält und mit DevOps Beispieldaten gefüllt ist.

5. Wählen Sie im Navigationsbereich den Abfrage-Editor
6. Wählen Sie im oberen Menü Beispielabfragen aus.
7. Klicken Sie auf eine der Beispielabfragen. Dadurch gelangen Sie zurück zum Abfrage-Editor, in dem der Editor die Beispielabfrage enthält.
8. Klicken Sie auf Ausführen, um die Abfrage auszuführen und die Abfrageergebnisse anzuzeigen.

Erstellen einer -Datenbank

Gehen Sie wie folgt vor, um mit der AWS Konsole eine Datenbank zu erstellen.

1. Öffnen Sie die AWS Konsole.
2. Wählen Sie im Navigationsbereich Datenbanken
3. Klicken Sie auf Datenbank erstellen.
4. Geben Sie auf der Seite „Datenbank erstellen“ Folgendes ein.
   - Konfiguration wählen — Wählen Sie Standarddatenbank aus.
   - Name — Geben Sie einen Datenbanknamen Ihrer Wahl ein.
   - Verschlüsselung — Wählen Sie einen KMS Schlüssel oder verwenden Sie die Standardoption, bei der Timestream Live Analytics einen KMS Schlüssel in Ihrem Konto erstellt, falls noch keiner vorhanden ist.
5. Klicken Sie auf Datenbank erstellen, um eine Datenbank zu erstellen.

Erstellen einer Tabelle

Gehen Sie wie folgt vor, um mit der AWS Konsole eine Tabelle zu erstellen.

1. Öffnen Sie die AWS Konsole.
2. Wählen Sie im Navigationsbereich Tabellen
3. Klicken Sie auf Tabelle erstellen.
4. Geben Sie auf der Seite „Tabelle erstellen“ Folgendes ein.
   - Datenbankname — Wählen Sie den Namen der Datenbank aus, die in Erstellen einer -Datenbank erstellt wurde.
   - Tabellenname — Geben Sie einen Tabellennamen Ihrer Wahl ein.

- Aufbewahrung im Speicher — Geben Sie an, wie lange Sie Daten im Speicherspeicher aufbewahren möchten. Der Speicherspeicher verarbeitet eingehende Daten, einschließlich spät eingehender Daten (Daten, deren Zeitstempel vor der aktuellen Uhrzeit liegt) und ist für schnelle point-in-time Abfragen optimiert.
- Aufbewahrung im Magnetspeicher — Geben Sie an, wie lange Sie Daten im Magnetspeicher aufbewahren möchten. Der Magnetspeicher ist für die Langzeitspeicherung vorgesehen und für schnelle analytische Abfragen optimiert.

5. Klicken Sie auf Tabelle erstellen.

Ausführen einer -Abfrage

Gehen Sie wie folgt vor, um Abfragen mit der AWS Konsole auszuführen.

1. Öffnen Sie die AWS Konsole.
2. Wählen Sie im Navigationsbereich den Abfrage-Editor
3. Wählen Sie im linken Bereich die Datenbank aus, die in erstellt wurdeErstellen einer -Datenbank.
4. Wählen Sie im linken Bereich die Datenbank aus, die in erstellt wurdeErstellen einer Tabelle.
5. Im Abfrage-Editor können Sie eine Abfrage ausführen. Um die letzten 10 Zeilen in der Tabelle zu sehen, führen Sie folgenden Befehl aus:

```
SELECT * FROM <database_name>.<table_name> ORDER BY time DESC LIMIT 10
```

6. (Optional) Aktivieren Sie die Option „Einblicke aktivieren“, um Einblicke in die Effizienz Ihrer Abfragen zu erhalten.

Erstellen Sie eine geplante Abfrage

Gehen Sie wie folgt vor, um mithilfe der AWS Konsole eine geplante Abfrage zu erstellen.

1. Öffnen Sie die AWS Konsole.
2. Wählen Sie im Navigationsbereich die Option Geplante Abfragen aus.
3. Klicken Sie auf Geplante Abfrage erstellen.
4. Geben Sie in den Abschnitten Abfragename und Zieltabelle Folgendes ein.

   - Name — Geben Sie einen Abfragenamen ein.

- Datenbankname — Wählen Sie den Namen der Datenbank aus, die in erstellt wurde. Erstellen einer -Datenbank
- Tabellenname — Wählen Sie den Namen der Tabelle aus, die in erstellt wurde. Erstellen einer Tabelle

5. Geben Sie im Abschnitt Abfrageanweisung eine gültige Abfrageanweisung ein. Klicken Sie dann auf Abfrage validieren.
6. Definieren Sie im Zieltabellenmodell das Modell für alle undefinierten Attribute. Sie können Visual Builder oder JSON verwenden.
7. Wählen Sie im Abschnitt Zeitplan ausführen die Option Feste Rate oder Chron-Ausdruck aus. Weitere Informationen zu Zeitplanausdrücken finden Sie unter Zeitplanausdrücke für geplante Abfragen.
8. Geben Sie im SNSThemenbereich das SNS Thema ein, das für Benachrichtigungen verwendet werden soll.
9. Geben Sie im Abschnitt Fehlerprotokollbericht den S3-Speicherort ein, der für die Meldung von Fehlern verwendet werden soll.

   Wählen Sie den Encryption key type (Verschlüsselungsschlüssel-Typ).
10. Wählen Sie im Abschnitt Sicherheitseinstellungen unter AWS KMSSchlüssel den AWS KMS Schlüsseltyp aus.

    Geben Sie die IAMRolle ein, für LiveAnalytics die Timestream die geplante Abfrage ausführen soll. Einzelheiten zu den erforderlichen Berechtigungen und der Vertrauensstellung für die Rolle finden Sie in den IAM Richtlinienbeispielen für geplante Abfragen.
11. Klicken Sie auf Geplante Abfrage erstellen.

Löschen Sie eine geplante Abfrage

Gehen Sie wie folgt vor, um eine geplante Abfrage mithilfe der AWS Konsole zu löschen oder zu deaktivieren.

1. Öffnen Sie die AWS Konsole.
2. Wählen Sie im Navigationsbereich die Option Geplante Abfragen
3. Wählen Sie die geplante Abfrage aus, die in erstellt wurdeErstellen Sie eine geplante Abfrage.
4. Wählen Sie Aktionen aus.
5. Wählen Sie Deaktivieren oder Löschen.

6. Wenn Sie Löschen ausgewählt haben, bestätigen Sie die Aktion und wählen Sie Löschen aus.

Löschen einer Tabelle

Gehen Sie wie folgt vor, um eine Datenbank mithilfe der AWS Konsole zu löschen.

1. Öffnen Sie die AWS Konsole.
2. Wählen Sie im Navigationsbereich Tabellen
3. Wählen Sie die Tabelle aus, in der Sie erstellt habenErstellen einer Tabelle.
4. Klicken Sie auf Delete.
5. Geben Sie Löschen in das Bestätigungsfeld ein.

Löschen einer Datenbank

Gehen Sie wie folgt vor, um eine Datenbank mit der AWS Konsole zu löschen:

1. Öffnen Sie die AWS Konsole.
2. Wählen Sie im Navigationsbereich Datenbanken
3. Wählen Sie die Datenbank aus, die Sie unter Datenbank erstellen erstellt haben.
4. Klicken Sie auf Delete.
5. Geben Sie Löschen in das Bestätigungsfeld ein.

Bearbeiten Sie eine Tabelle

Gehen Sie wie folgt vor, um eine Tabelle mit der AWS Konsole zu bearbeiten.

1. Öffnen Sie die AWS Konsole.
2. Wählen Sie im Navigationsbereich Tabellen
3. Wählen Sie die Tabelle aus, in der Sie erstellt habenErstellen einer Tabelle.
4. Klicken Sie auf Bearbeiten
5. Bearbeiten Sie die Tabellendetails und speichern Sie sie.

    - Aufbewahrung im Speicher — Geben Sie an, wie lange Sie Daten im Speicherspeicher aufbewahren möchten. Der Speicherspeicher verarbeitet eingehende Daten, einschließlich

spät eingehender Daten (Daten, deren Zeitstempel vor der aktuellen Uhrzeit liegt) und ist für schnelle point-in-time Abfragen optimiert.

- Aufbewahrung im Magnetspeicher — Geben Sie an, wie lange Sie Daten im Magnetspeicher aufbewahren möchten. Der Magnetspeicher ist für die Langzeitspeicherung vorgesehen und für schnelle analytische Abfragen optimiert.

Bearbeiten Sie eine Datenbank

Gehen Sie wie folgt vor, um eine Datenbank mit der AWS Konsole zu bearbeiten.

1. Öffnen Sie die AWS Konsole.
2. Wählen Sie im Navigationsbereich Datenbanken
3. Wählen Sie die Datenbank aus, die Sie unter Datenbank erstellen erstellt haben.
4. Klicken Sie auf Bearbeiten
5. Bearbeiten Sie die Datenbankdetails und speichern Sie sie.

Zugreifen auf Amazon Timestream für die LiveAnalytics Nutzung des AWS CLI

Sie können die AWS Command Line Interface (AWS CLI) verwenden, um mehrere AWS Dienste von der Befehlszeile aus zu steuern und sie mithilfe von Skripten zu automatisieren. Sie können das AWS CLI für Ad-hoc-Operationen verwenden. Sie können es auch verwenden, um Amazon Timestream für LiveAnalytics Operationen in Utility-Skripten einzubetten.

Bevor Sie das AWS CLI mit Timestream for verwenden können LiveAnalytics, müssen Sie den programmatischen Zugriff einrichten. Weitere Informationen finden Sie unter Erteilen programmgesteuerten Zugriffs.

Eine vollständige Liste aller Befehle, die für Timestream for LiveAnalytics Query API in verfügbar sind AWS CLI, finden Sie in der Befehlsreferenz.AWS CLI

Eine vollständige Liste aller Befehle, die für Timestream for LiveAnalytics Write API in der verfügbar sind AWS CLI, finden Sie in der AWS CLI Befehlsreferenz.

Themen

- Herunterladen und Konfigurieren des AWS CLI

- Verwenden von AWS CLI mit Timestream für LiveAnalytics

Herunterladen und Konfigurieren des AWS CLI

Das AWS CLI läuft unter Windows, MacOS oder Linux. Zum Herunterladen, Installieren undKonfigurieren führen Sie die folgenden Schritte aus:

1. Laden Sie das AWS CLI unter http://aws.amazon.com/cli herunter.
2. Folgen Sie den Anweisungen zur Installation AWS CLI und Konfiguration von AWS CLI im AWS Command Line Interface Benutzerhandbuch.

Verwenden von AWS CLI mit Timestream für LiveAnalytics

Das Befehlszeilenformat besteht aus einem Amazon Timestream für den LiveAnalytics Operationsnamen, gefolgt von den Parametern für diesen Vorgang. Das AWS CLI unterstützt zusätzlich zu eine Kurzsyntax für die Parameterwerte. JSON

Wird verwendet`help`, um alle verfügbaren Befehle in Timestream für aufzulisten. LiveAnalytics Beispielsweise:

```
aws timestream-write help
```

```
aws timestream-query help
```

Sie können auch `help` verwenden, um einen bestimmten Befehl zu beschreiben und mehr über seine Nutzung zu erfahren:

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

Um beispielsweise eine Datenbank zu erstellen:

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

Um eine Tabelle mit aktivierten Magnetspeicher-Schreibvorgängen zu erstellen:

```
aws timestream-write create-table \
```

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

So schreiben Sie Daten mithilfe von Einzelmessdatensätzen:

```
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\"}]"
```

Um Daten mithilfe von Datensätzen mit mehreren Messwerten zu schreiben:

```
# 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
```

So fragen Sie eine Tabelle ab:

```
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)"
```

Um eine geplante Abfrage zu erstellen:

```
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\"\
       }\
     }"
```

Mit dem API

Darüber hinaus LiveAnalytics bietet Amazon Timestream for direkten REST API Zugriff über das Endpoint Discovery Pattern. SDKs Das Endpoint Discovery Pattern wird im Folgenden zusammen mit seinen Anwendungsfällen beschrieben.

Das Endpoint Discovery-Muster

Da Timestream Live Analytics so konzipiert SDKs sind, dass sie transparent mit der Architektur des Dienstes zusammenarbeiten, einschließlich der Verwaltung und Zuordnung der Dienstendpunkte, wird empfohlen, sie SDKs für die meisten Anwendungen zu verwenden. Es gibt jedoch einige Fälle, in denen die Verwendung des Musters Timestream for LiveAnalytics REST API Endpoint Discovery erforderlich ist:

- Sie verwenden VPCendpoints (AWS PrivateLink) mit Timestream für LiveAnalytics
- Ihre Anwendung verwendet eine Programmiersprache, die noch nicht unterstützt wird SDK
- Sie benötigen eine bessere Kontrolle über die clientseitige Implementierung

Dieser Abschnitt enthält Informationen darüber, wie das Endpoint Discovery Pattern funktioniert, wie das Endpoint Discovery Pattern implementiert wird, sowie Nutzungshinweise. Wählen Sie unten ein Thema aus, um mehr zu erfahren.

Themen

- So funktioniert das Endpoint Discovery Pattern
- Implementierung des Endpoint Discovery Patterns

So funktioniert das Endpoint Discovery Pattern

Timestream basiert auf einer Mobilfunkarchitektur, um eine bessere Skalierung und Isolierung des Datenverkehrs zu gewährleisten. Da jedes Kundenkonto einer bestimmten Zelle in einer Region zugeordnet ist, muss Ihre Anwendung die richtigen zellenspezifischen Endpunkte verwenden, denen Ihr Konto zugeordnet wurde. Wenn Sie den verwendenSDKs, wird diese Zuordnung transparent für Sie vorgenommen, und Sie müssen die zellenspezifischen Endpunkte nicht verwalten. Wenn Sie jedoch direkt auf die zugreifen RESTAPI, müssen Sie die richtigen Endpunkte selbst verwalten und zuordnen. Dieser Prozess, das Endpoint Discovery Pattern, wird im Folgenden beschrieben:

1. Das Endpoint Discovery Pattern beginnt mit einem Aufruf zur `DescribeEndpoints` Aktion (im `DescribeEndpoints`Abschnitt beschrieben).
2. Der Endpunkt sollte zwischengespeichert und für den Zeitraum wiederverwendet werden, der durch den zurückgegebenen Wert time-to-live (TTL) (der `CachePeriodInMinutes`) angegeben ist. Aufrufe von Timestream Live Analytics API können dann für die Dauer von getätigt werden. TTL
3. Nach TTL Ablauf DescribeEndpoints sollte ein neuer Aufruf an erfolgen, um den Endpunkt zu aktualisieren (mit anderen Worten, beginnen Sie bei Schritt 1 von vorne).

> **Note**
>
> Syntax, Parameter und andere Nutzungsinformationen für die `DescribeEndpoints` Aktion werden in der APIReferenz beschrieben. Beachten Sie, dass die `DescribeEndpoints` Aktion über beide SDKs verfügbar ist und für beide identisch ist.

Informationen zur Implementierung des Endpoint Discovery Patterns finden Sie unterImplementierung des Endpoint Discovery Patterns.

Implementierung des Endpoint Discovery Patterns

Um das Endpoint Discovery-Muster zu implementieren, wählen Sie ein API (Schreiben oder Abfragen), erstellen Sie eine DescribeEndpointsAnfrage und verwenden Sie die zurückgegebenen Endpunkte für die Dauer der zurückgegebenen TTL Werte. Das Implementierungsverfahren wird unten beschrieben.

> **Note**
>
> Stellen Sie sicher, dass Sie mit den Nutzungshinweisen vertraut sind.

Verfahren zur Implementierung

1. Ermitteln API Sie mithilfe der Anfrage den Endpunkt für den, gegen den Sie Anrufe tätigen möchten (Write oder Query). `DescribeEndpoints`

   a. Erstellen Sie mithilfe eines API der beiden unten beschriebenen Endpunkte eine Anfrage, die der gewünschten Anfrage entspricht (Write oder Query). `DescribeEndpoints` Es gibt keine Eingabeparameter für die Anfrage. Stellen Sie sicher, dass Sie die folgenden Hinweise gelesen haben.

   Schreiben SieSDK:

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

   AnfrageSDK:

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

   Es `us-east-1` folgt ein Beispiel für einen CLI Aufruf für die Region.

   ```
   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**
   >
   > Der Header HTTP „Host“ muss auch den API Endpunkt enthalten. Die Anfrage schlägt fehl, wenn der Header nicht gefüllt ist. Dies ist eine Standardanforderung für alle HTTP /1.1-Anfragen. Wenn Sie eine HTTP Bibliothek verwenden, die 1.1

oder höher unterstützt, sollte die HTTP Bibliothek den Header automatisch für Sie ausfüllen.

> **Note**
>
> Ersetze *<region>* mit der Regionskennung für die Region, in der die Anfrage gestellt wird, z. B. `us-east-1`

b. Analysieren Sie die Antwort, um die Endpunkte und die TTL Cache-Werte zu extrahieren. Die Antwort ist ein Array von einem oder mehreren `EndpointObjekten`. Jedes `Endpoint` Objekt enthält eine Endpunktadresse (`Address`) und die TTL für diesen Endpunkt (`CachePeriodInMinutes`).

2. Zwischenspeichern Sie den Endpunkt bis zum angegebenen WertTTL.
3. Wenn der TTL abläuft, rufen Sie einen neuen Endpunkt ab, indem Sie bei Schritt 1 der Implementierung von vorne beginnen.

Nutzungshinweise für das Endpoint Discovery Pattern

- Die DescribeEndpointsAktion ist die einzige Aktion, die von den regionalen Endpunkten von Timestream Live Analytics erkannt wird.
- Die Antwort enthält eine Liste von Endpunkten, gegen die Timestream Live Analytics-Aufrufe getätigt werden sollen. API
- Bei erfolgreicher Antwort sollte die Liste mindestens einen Endpunkt enthalten. Wenn die Liste mehr als einen Endpunkt enthält, kann jeder von ihnen gleichermaßen für die API Anrufe verwendet werden, und der Anrufer kann den zu verwendenden Endpunkt nach dem Zufallsprinzip auswählen.
- Zusätzlich zur DNS Adresse des Endpunkts gibt jeder Endpunkt in der Liste eine Gültigkeitsdauer (TTL) an, die für die Nutzung des angegebenen Endpunkts in Minuten zulässig ist.
- Der Endpunkt sollte zwischengespeichert und für die durch den zurückgegebenen TTL Wert angegebene Zeit (in Minuten) wiederverwendet werden. Nach TTL Ablauf DescribeEndpointssollte ein neuer Aufruf an erfolgen, um den zu verwendenden Endpunkt zu aktualisieren, da der Endpunkt nach TTL Ablauf des nicht mehr funktioniert.

Mit dem AWS SDKs

Sie können mit dem auf Amazon Timestream zugreifen. AWS SDKs Timestream unterstützt zwei SDKs pro Sprache, nämlich Write SDK und Query. SDK Write SDK wird verwendet, um CRUD Operationen auszuführen und Ihre Zeitreihendaten in Timestream einzufügen. Die Abfrage SDK wird verwendet, um Ihre vorhandenen Zeitreihendaten abzufragen, die in Timestream gespeichert sind.

Sobald Sie die erforderlichen Voraussetzungen für Ihre SDK Wahl erfüllt haben, können Sie mit dem Codebeispiele beginnen.

Themen

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

Java

Um mit Java 1.0 SDK und Amazon Timestream zu beginnen, müssen Sie die unten beschriebenen Voraussetzungen erfüllen.

Sobald Sie die erforderlichen Voraussetzungen für Java erfüllt habenSDK, können Sie mit dem Codebeispiele beginnen.

Voraussetzungen

Bevor Sie mit Java beginnen, müssen Sie Folgendes tun:

1. Folgen Sie den Anweisungen zur AWS Einrichtung unterZugreifen auf Timestream für LiveAnalytics.
2. Richten Sie eine Java-Entwicklungsumgebung ein, indem Sie Folgendes herunterladen und installieren:
   - Java SE Development Kit 8 (z. B. Amazon Corretto 8).
   - Java IDE (wie Eclipse oder IntelliJ).

Weitere Informationen finden Sie unter Erste Schritte mit dem AWS SDK for Java

3. Konfigurieren Sie Ihre AWS Anmeldeinformationen und Ihre Region für die Entwicklung:

   - Richten Sie Ihre AWS Sicherheitsanmeldedaten für die Verwendung mit dem ein AWS SDK for Java.
   - Stellen Sie Ihre AWS Region ein, um Ihren Standard-Timestream für den LiveAnalytics Endpunkt zu bestimmen.

Verwenden von Apache Maven

Sie können Apache Maven verwenden, um Projekte zu konfigurieren und zu erstellen AWS SDK for Java .

> **Note**
>
> Um Apache Maven verwenden zu können, stellen Sie sicher, dass Java SDK und Runtime 1.8 oder höher sind.

Sie können die AWS SDK als Maven-Abhängigkeit konfigurieren, wie unter Verwenden von SDK mit Apache Maven beschrieben.

Sie können Compile ausführen und Ihren Quellcode mit dem folgenden Befehl ausführen:

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

> **Note**
>
> `<your source code Main class>`ist der Pfad zur Hauptklasse Ihres Java-Quellcodes.

Einstellung Ihrer AWS Anmeldedaten

Das AWS SDK for Javaerfordert, dass Sie zur Laufzeit AWS Anmeldeinformationen für Ihre Anwendung angeben. Bei den Codebeispielen in diesem Handbuch wird davon ausgegangen, dass Sie eine AWS Anmeldeinformationsdatei verwenden, wie unter Einrichten von AWS

Anmeldeinformationen und Region für die Entwicklung im AWS SDK for Java Entwicklerhandbuch beschrieben.

Im Folgenden finden Sie ein Beispiel für eine AWS Anmeldeinformationsdatei mit dem Namen`~/.aws/credentials`, wobei die Tilde (~) für Ihr Home-Verzeichnis steht.

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

Java v2

Um mit Java 2.0 SDK und Amazon Timestream zu beginnen, müssen Sie die unten beschriebenen Voraussetzungen erfüllen.

Sobald Sie die erforderlichen Voraussetzungen für Java 2.0 erfüllt habenSDK, können Sie mit dem Codebeispiele beginnen.

Voraussetzungen

Bevor Sie mit Java beginnen, müssen Sie Folgendes tun:

1. Folgen Sie den Anweisungen zur AWS Einrichtung unterZugreifen auf Timestream für LiveAnalytics.
2. Sie können die AWS SDK als Maven-Abhängigkeit konfigurieren, wie unter Verwenden von SDK mit Apache Maven beschrieben.
3. Richten Sie eine Java-Entwicklungsumgebung ein, indem Sie Folgendes herunterladen und installieren:

   - Java SE Development Kit 8 (z. B. Amazon Corretto 8).
   - Java IDE (wie Eclipse oder IntelliJ).

     Weitere Informationen finden Sie unter Erste Schritte mit dem AWS SDK for Java

Verwenden von Apache Maven

Sie können Apache Maven verwenden, um AWS SDK for Java Projekte zu konfigurieren und zu erstellen.

> **Note**
>
> Um Apache Maven verwenden zu können, stellen Sie sicher, dass Java SDK und Runtime 1.8 oder höher sind.

Sie können die AWS SDK als Maven-Abhängigkeit konfigurieren, wie unter Verwenden von SDK mit Apache Maven beschrieben. Die an der Datei pom.xml erforderlichen Änderungen werden hier beschrieben.

Sie können Compile ausführen und Ihren Quellcode mit dem folgenden Befehl ausführen:

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

> **Note**
>
> `<your source code Main class>`ist der Pfad zur Hauptklasse Ihres Java-Quellcodes.

Go

Um mit Go SDK und Amazon Timestream zu beginnen, müssen Sie die unten beschriebenen Voraussetzungen erfüllen.

Sobald Sie die erforderlichen Voraussetzungen für Go erfüllt habenSDK, können Sie mit dem Codebeispiele beginnen.

Voraussetzungen

1. Laden Sie das GO SDK 1.14 herunter.
2. Konfiguriere das GO SDK.
3. Konstruieren Sie Ihren Client.

Python

Um mit Python SDK und Amazon Timestream zu beginnen, müssen Sie die unten beschriebenen Voraussetzungen erfüllen.

Sobald Sie die erforderlichen Voraussetzungen für Python erfüllt habenSDK, können Sie mit dem beginnenCodebeispiele.

Voraussetzungen

Um Python zu verwenden, installieren und konfigurieren Sie Boto3, indem Sie den Anweisungen hier folgen.

Node.js

Um mit Node.js SDK und Amazon Timestream zu beginnen, müssen Sie die unten beschriebenen Voraussetzungen erfüllen.

Sobald Sie die erforderlichen Voraussetzungen für die Datei Node.js erfüllt habenSDK, können Sie mit dem Codebeispiele beginnen.

Voraussetzungen

Bevor Sie mit Node.js beginnen, müssen Sie Folgendes tun:

1. Installieren Sie Node.js.
2. Installieren Sie das AWS SDK für JavaScript.

.NET

Um mit dem zu beginnen. NETSDKund Amazon Timestream, erfüllen Sie die unten beschriebenen Voraussetzungen.

Sobald Sie die erforderlichen Voraussetzungen für die erfüllt haben. NETSDK, können Sie mit dem beginnenCodebeispiele.

Voraussetzungen

Bevor Sie anfangen mit. NET, installieren Sie die erforderlichen NuGet Pakete und stellen Sie sicher, dass die AWSSDK .Core-Version 3.3.107 oder neuer ist, indem Sie die folgenden Befehle ausführen:

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

Erste Schritte

Dieser Abschnitt enthält ein Tutorial für die ersten Schritte mit Amazon Timestream Live Analytics sowie Anweisungen zum Einrichten einer voll funktionsfähigen Beispielanwendung. Sie können mit dem Tutorial oder der Beispielanwendung beginnen, indem Sie einen der folgenden Links auswählen.

Themen

Tutorial

In diesem Tutorial erfahren Sie, wie Sie eine Datenbank mit Beispieldatensätzen erstellen und Beispielabfragen ausführen. Die in diesem Tutorial verwendeten Beispieldatensätze werden häufig in IoT und DevOps Szenarien verwendet. Der IoT-Datensatz enthält Zeitreihendaten wie Geschwindigkeit, Standort und Ladung eines Lkw, um das Flottenmanagement zu rationalisieren und Optimierungsmöglichkeiten zu identifizieren. Der DevOps Datensatz enthält EC2 Instanzmetriken wie CPU Netzwerk- und Speicherauslastung, um die Leistung und Verfügbarkeit von Anwendungen zu verbessern. Hier finden Sie ein Video-Tutorial für die in diesem Abschnitt beschriebenen Anweisungen.

Gehen Sie wie folgt vor, um eine Datenbank mit den Beispieldatensätzen zu erstellen und Beispielabfragen mithilfe der AWS Konsole auszuführen:

Verwenden der Konsole

Gehen Sie wie folgt vor, um eine Datenbank mit den Beispieldatensätzen zu erstellen und Beispielabfragen mithilfe der AWS Konsole auszuführen:

1. Öffnen Sie die AWS Konsole.
2. Wählen Sie im Navigationsbereich Datenbanken
3. Klicken Sie auf Datenbank erstellen.
4. Geben Sie auf der Seite „Datenbank erstellen“ Folgendes ein:
   - Konfiguration wählen — Wählen Sie Beispieldatenbank aus.
   - Name — Geben Sie einen Datenbanknamen Ihrer Wahl ein.

> **Note**
>
> Nachdem Sie eine Datenbank mit Beispieldatensätzen erstellt haben, können Sie zur Verwendung der in der Konsole verfügbaren Beispielabfragen den Datenbanknamen, auf den in der Abfrage verwiesen wird, so anpassen, dass er dem Datenbanknamen entspricht, den Sie hier eingeben. Für jede Kombination aus Beispieldatensatz und Art von Zeitreihendatensätzen gibt es Beispielabfragen.

- Wählen Sie Beispieldatensätze aus — Wählen Sie IoT und DevOps.
- Wählen Sie den Typ der Zeitreihendatensätze aus — Wählen Sie Datensätze mit mehreren Messwerten aus.
- Klicken Sie auf Datenbank erstellen, um eine Datenbank mit zwei Tabellen zu erstellen, die mit Beispieldaten gefüllt sind. Die Tabellennamen für Beispieldatensätze mit Datensätzen mit mehreren Messwerten lauten `DevOpsMulti` und`IoTMulti`. Die Tabellennamen für Beispieldatensätze mit Einzelmessdatensätzen lauten und. `DevOps` `IoT`

5. Wählen Sie im Navigationsbereich den Abfrage-Editor aus
6. Wählen Sie im oberen Menü Beispielabfragen aus.
7. Klicken Sie auf eine der Beispielabfragen für einen Datensatz, den Sie bei der Erstellung der Beispieldatenbank ausgewählt haben. Dadurch gelangen Sie zurück zum Abfrage-Editor, in dem der Editor die Beispielabfrage enthält.
8. Passen Sie den Datenbanknamen für die Beispielabfrage an.
9. Klicken Sie auf Ausführen, um die Abfrage auszuführen und die Abfrageergebnisse zu sehen.

Mit dem SDKs

Timestream Live Analytics bietet eine voll funktionsfähige Beispielanwendung, die Ihnen zeigt, wie Sie eine Datenbank und eine Tabelle erstellen, die Tabelle mit ~126.000 Zeilen mit Beispieldaten füllen und Beispielabfragen ausführen. Die Beispielanwendung ist GitHubfür Java, Python, Node.js, Go und verfügbar. NET.

1. Klonen Sie die Timestream Live Analytics-Beispielanwendungen des GitHub Repositorys, indem Sie den Anweisungen von GitHub folgen.
2. Konfigurieren Sie die AWS SDK, um eine Verbindung zu Amazon Timestream Live Analytics herzustellen, indem Sie den Anweisungen unter folgen. Mit dem AWS SDKs

3. Kompilieren Sie die Beispielanwendung und führen Sie sie mithilfe der folgenden Anweisungen aus:

   - Anweisungen für die Java-Beispielanwendung.
   - Anweisungen für die Java v2-Beispielanwendung.
   - Anweisungen für die Go-Beispielanwendung.
   - Anweisungen für die Python-Beispielanwendung.
   - Anweisungen für die Beispielanwendung Node.js.
   - Anweisungen für die. NETBeispielanwendung.

Beispielanwendung

Timestream wird mit einer voll funktionsfähigen Beispielanwendung geliefert, die zeigt, wie Sie eine Datenbank und eine Tabelle erstellen, die Tabelle mit ~126.000 Zeilen mit Beispieldaten füllen und Beispielabfragen ausführen. Gehen Sie wie folgt vor, um mit der Beispielanwendung in einer der unterstützten Sprachen zu beginnen:

Java

1. Klonen Sie das GitHub Repository Timestream für LiveAnalytics Beispielanwendungen, indem Sie den Anweisungen von GitHubfolgen.
2. Konfigurieren Sie den AWSSDK, um eine Verbindung zu Timestream herzustellen, um den LiveAnalytics Anweisungen unter Erste Schritte mit zu folgen. Java
3. Führen Sie die Java-Beispielanwendung gemäß den hier beschriebenen Anweisungen aus

Java v2

1. Klonen Sie das GitHub Repository Timestream für LiveAnalytics Beispielanwendungen gemäß den Anweisungen von GitHub.
2. Konfigurieren Sie die AWS SDK, um eine Verbindung zu Amazon Timestream herzustellen, um die unter Erste Schritte mit Java v2 beschriebenen Anweisungen zu LiveAnalytics befolgen.
3. Führen Sie die Java 2.0-Beispielanwendung gemäß den hier beschriebenen Anweisungen aus

Go

1. Klonen Sie das GitHub Repository Timestream für LiveAnalytics Beispielanwendungen gemäß den Anweisungen von GitHub.
2. Konfigurieren Sie die AWS SDK, um eine Verbindung zu Amazon Timestream herzustellen, um die unter Erste Schritte mit Go beschriebenen Anweisungen zu LiveAnalytics befolgen.
3. Führen Sie die Go-Beispielanwendung gemäß den hier beschriebenen Anweisungen aus

Python

1. Klonen Sie das GitHub Repository Timestream für LiveAnalytics Beispielanwendungen gemäß den Anweisungen von GitHub.
2. Konfigurieren Sie die AWS SDK, um eine Verbindung zu Amazon Timestream herzustellen, um die unter beschriebenen Anweisungen zu LiveAnalytics befolgen. Python
3. Führen Sie die Python-Beispielanwendung gemäß den hier beschriebenen Anweisungen aus

Node.js

1. Klonen Sie das GitHub Repository Timestream für LiveAnalytics Beispielanwendungen gemäß den Anweisungen von GitHub.
2. Konfigurieren Sie die AWS SDK, um eine Verbindung zu Amazon Timestream herzustellen, um die unter Erste Schritte mit Node.js beschriebenen Anweisungen zu LiveAnalytics befolgen.
3. Führen Sie die Beispielanwendung Node.js gemäß den hier beschriebenen Anweisungen aus

.NET

1. Klonen Sie das GitHub Repository Timestream für LiveAnalytics Beispielanwendungen gemäß den Anweisungen von GitHub.
2. Konfigurieren Sie die AWS SDK, um eine Verbindung zu Amazon Timestream herzustellen, um die unter Erste Schritte mit .NET beschriebenen Anweisungen zu LiveAnalytics befolgen.
3. Führen Sie das aus. NETBeispielanwendung gemäß den hier beschriebenen Anweisungen

Codebeispiele

Sie können mit dem auf Amazon Timestream zugreifen. AWS SDKs Timestream unterstützt zwei SDKs pro Sprache, nämlich Write SDK und Query. SDK Write SDK wird verwendet, um CRUD Operationen auszuführen und Ihre Zeitreihendaten in Timestream einzufügen. Die Abfrage SDK wird verwendet, um Ihre vorhandenen Zeitreihendaten abzufragen, die in Timestream gespeichert sind. Wählen Sie ein Thema aus der folgenden Liste aus, um weitere Informationen zu erhalten, einschließlich Codebeispielen für jedes der unterstützten SDKs Themen.

Themen

- Geplante Abfragen auflisten
- Beschreiben Sie die geplante Abfrage
- Geplante Abfrage ausführen
- Geplante Abfrage aktualisieren
- Geplante Abfrage löschen

SDKKunde schreiben

Sie können die folgenden Codefragmente verwenden, um einen Timestream-Client für Write zu erstellen. SDK Write SDK wird verwendet, um CRUD Operationen auszuführen und Ihre Zeitreihendaten in Timestream einzufügen.

> **Note**
>
> Diese Codefragmente basieren auf vollständigen Beispielanwendungen auf. GitHub Weitere Informationen zu den ersten Schritten mit den Beispielanwendungen finden Sie unter. Beispielanwendung

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

Der folgende Codeausschnitt wird AWS SDK für JavaScript Version 3 verwendet. Weitere Informationen zur Installation und Verwendung des Clients finden Sie unter Timestream Write Client — AWS SDK for v3. JavaScript

Ein zusätzlicher Befehlsimport wird hier gezeigt. Der `CreateDatabaseCommand` Import ist nicht erforderlich, um den Client zu erstellen.

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

Das folgende Snippet verwendet den AWS SDK for JavaScript V2-Stil. Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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);
```

Wir empfehlen Ihnen, die folgende Konfiguration zu verwenden.

- Stellen Sie die Anzahl der SDK Wiederholungsversuche auf ein`10`.
- Verwenden Sie `SDK DEFAULT_BACKOFF_STRATEGY`.
- Stellen Sie den `RequestTimeout` Wert auf `20` Sekunden ein.
- Stellen Sie die maximale Anzahl an Verbindungen auf `5000` oder höher ein.

SDKClient abfragen

Sie können die folgenden Codefragmente verwenden, um einen Timestream-Client für die Abfrage zu erstellen. SDK Die Abfrage SDK wird verwendet, um Ihre vorhandenen Zeitreihendaten abzufragen, die in Timestream gespeichert sind.

> **Note**
>
> Diese Codefragmente basieren auf vollständigen Beispielanwendungen auf. GitHub Weitere Informationen zu den ersten Schritten mit den Beispielanwendungen finden Sie unter. Beispielanwendung

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

Der folgende Codeausschnitt wird AWS SDK für JavaScript Version 3 verwendet. Weitere Informationen zur Installation und Verwendung des Clients finden Sie unter Timestream Query Client -,AWS SDK for v3. JavaScript

Ein zusätzlicher Befehlsimport wird hier gezeigt. Der `QueryCommand` Import ist nicht erforderlich, um den Client zu erstellen.

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

Das folgende Snippet verwendet den AWS SDK for JavaScript V2-Stil. Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf GitHub.

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

.NET

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

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

Datenbank erstellen

Sie können die folgenden Codefragmente verwenden, um eine Datenbank zu erstellen.

> **Note**
>
> Diese Codefragmente basieren auf vollständigen Beispielanwendungen auf. GitHub Weitere Informationen zu den ersten Schritten mit den Beispielanwendungen finden Sie unter. Beispielanwendung

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

Der folgende Codeausschnitt wird AWS SDK für JavaScript Version 3 verwendet. Weitere Informationen zur Installation und Verwendung des Clients finden Sie unter Timestream Write Client — AWS SDK for v3. JavaScript

Siehe auch Klasse CreateDatabaseCommand und 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);
    }
}
```

Das folgende Snippet verwendet den AWS SDK for JavaScript V2-Stil. Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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());
            }

        }
```

Datenbank beschreiben

Sie können die folgenden Codefragmente verwenden, um Informationen über die Attribute Ihrer neu erstellten Datenbank zu erhalten.

> **Note**
>
> Diese Codefragmente basieren auf vollständigen Beispielanwendungen auf. GitHub Weitere Informationen zu den ersten Schritten mit den Beispielanwendungen finden Sie unter. Beispielanwendung

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

Der folgende Codeausschnitt wird AWS SDK für JavaScript Version 3 verwendet. Weitere Informationen zur Installation und Verwendung des Clients finden Sie unter Timestream Write Client — AWS SDK for v3. JavaScript

Siehe auch Klasse DescribeDatabaseCommand und 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;
    }
}
```

Das folgende Snippet verwendet den AWS SDK for JavaScript V2-Stil. Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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());
            }

        }
```

Datenbank aktualisieren

Sie können die folgenden Codefragmente verwenden, um Ihre Datenbanken zu aktualisieren.

> **Note**
>
> Diese Codefragmente basieren auf vollständigen Beispielanwendungen auf. GitHub Weitere Informationen zu den ersten Schritten mit den Beispielanwendungen finden Sie unter. Beispielanwendung

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

Der folgende Codeausschnitt wird AWS SDK für JavaScript Version 3 verwendet. Weitere Informationen zur Installation und Verwendung des Clients finden Sie unter Timestream Write Client — AWS SDK for v3. JavaScript

Siehe auch Klasse UpdateDatabaseCommand und 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);
    }
}
```

Das folgende Snippet verwendet den AWS SDK for JavaScript V2-Stil. Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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}");
        }
```

Datenbank löschen

Sie können den folgenden Codeausschnitt verwenden, um eine Datenbank zu löschen.

> **Note**
>
> Diese Codefragmente basieren auf vollständigen Beispielanwendungen auf. GitHub Weitere Informationen zu den ersten Schritten mit den Beispielanwendungen finden Sie unter. Beispielanwendung

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

Der folgende Codeausschnitt wird AWS SDK für JavaScript Version 3 verwendet. Weitere Informationen zur Installation und Verwendung des Clients finden Sie unter Timestream Write Client — AWS SDK for v3. JavaScript

Siehe auch Klasse DeleteDatabaseCommand und 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;
    }
}
```

Das folgende Snippet verwendet den AWS SDK for JavaScript V2-Stil. Es basiert auf der Beispielanwendung unter [Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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());
            }
        }
```

Datenbanken auflisten

Sie können die folgenden Codefragmente verwenden, um Ihre Datenbanken aufzulisten.

> **Note**
>
> Diese Codefragmente basieren auf vollständigen Beispielanwendungen auf. GitHub Weitere Informationen zu den ersten Schritten mit den Beispielanwendungen finden Sie unter. Beispielanwendung

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

Der folgende Codeausschnitt wird AWS SDK für JavaScript Version 3 verwendet. Weitere Informationen zur Installation und Verwendung des Clients finden Sie unter Timestream Write Client — AWS SDK for v3. JavaScript

Siehe auch Klasse ListDatabasesCommand und 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);
    }
}
```

Das folgende Snippet verwendet den AWS SDK for JavaScript V2-Stil. Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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());
            }

        }
```

Create table

Themen

- Speichert Schreibvorgänge
- Magnetic Store schreibt

Speichert Schreibvorgänge

Sie können den folgenden Codeausschnitt verwenden, um eine Tabelle zu erstellen, in der Magnetspeicher-Schreibvorgänge deaktiviert sind. Daher können Sie nur Daten in das Aufbewahrungsfenster für den Speicherspeicher schreiben.

> **Note**
>
> Diese Codefragmente basieren auf vollständigen Beispielanwendungen auf. GitHub Weitere Informationen zu den ersten Schritten mit den Beispielanwendungen finden Sie unter. Beispielanwendung

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

Der folgende Codeausschnitt wird AWS SDK für JavaScript Version 3 verwendet. Weitere Informationen zur Installation und Verwendung des Clients finden Sie unter Timestream Write Client — AWS SDK for v3. JavaScript

Siehe auch Klasse CreateTableCommand und 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;
    }
}
```

Das folgende Snippet verwendet den AWS SDK for JavaScript V2-Stil. Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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());
            }
```

```
        }
```

Magnetic Store schreibt

Sie können den folgenden Codeausschnitt verwenden, um eine Tabelle zu erstellen, in der Magnetic Store Writes aktiviert sind. Mit Magnetic Store Writes können Sie Daten sowohl in das Aufbewahrungsfenster Ihres Speichers als auch in das Aufbewahrungsfenster Ihres Magnetspeichers schreiben.

> **Note**
>
> Diese Codefragmente basieren auf vollständigen Beispielanwendungen auf. GitHub Weitere Informationen zu den ersten Schritten mit den Beispielanwendungen finden Sie unter. Beispielanwendung

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());
            }

        }
```

Tabelle beschreiben

Sie können die folgenden Codefragmente verwenden, um Informationen zu den Attributen Ihrer Tabelle zu erhalten.

> **Note**
>
> Diese Codefragmente basieren auf vollständigen Beispielanwendungen auf. GitHub Weitere Informationen zu den ersten Schritten mit den Beispielanwendungen finden Sie unter. Beispielanwendung

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

Der folgende Codeausschnitt wird AWS SDK für JavaScript Version 3 verwendet. Weitere Informationen zur Installation und Verwendung des Clients finden Sie unter Timestream Write Client — AWS SDK for v3. JavaScript

Siehe auch Klasse DescribeTableCommand und 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;
    }
}
```

Das folgende Snippet verwendet den AWS SDK for JavaScript V2-Stil. Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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());
            }

        }
```

Tabelle aktualisieren

Sie können die folgenden Codefragmente verwenden, um eine Tabelle zu aktualisieren.

> **Note**
>
> Diese Codefragmente basieren auf vollständigen Beispielanwendungen auf. GitHub Weitere Informationen zu den ersten Schritten mit den Beispielanwendungen finden Sie unter. Beispielanwendung

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

Der folgende Codeausschnitt wird AWS SDK für JavaScript Version 3 verwendet. Weitere Informationen zur Installation und Verwendung des Clients finden Sie unter Timestream Write Client — AWS SDK for v3. JavaScript

Siehe auch Klasse UpdateTableCommand und 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);
}
```

Das folgende Snippet verwendet den AWS SDK for JavaScript V2-Stil. Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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());
            }

        }
```

Tabelle löschen

Sie können die folgenden Codefragmente verwenden, um eine Tabelle zu löschen.

> **Note**
>
> Diese Codefragmente basieren auf vollständigen Beispielanwendungen auf. GitHub Weitere Informationen zu den ersten Schritten mit den Beispielanwendungen finden Sie unter. Beispielanwendung

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

Der folgende Codeausschnitt wird AWS SDK für JavaScript Version 3 verwendet. Weitere Informationen zur Installation und Verwendung des Clients finden Sie unter Timestream Write Client — AWS SDK for v3. JavaScript

Siehe auch Klasse DeleteTableCommand und 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;
    }
}
```

Das folgende Snippet verwendet den AWS SDK for JavaScript V2-Stil. Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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());
            }
        }
```

Auflisten von Tabellen

Sie können die folgenden Codefragmente verwenden, um Tabellen aufzulisten.

> **Note**
>
> Diese Codefragmente basieren auf vollständigen Beispielanwendungen auf. GitHub Weitere Informationen zu den ersten Schritten mit den Beispielanwendungen finden Sie unter. Beispielanwendung

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

Der folgende Codeausschnitt wird AWS SDK für JavaScript Version 3 verwendet. Weitere Informationen zur Installation und Verwendung des Clients finden Sie unter Timestream Write Client — AWS SDK for v3. JavaScript

Siehe auch Klasse ListTablesCommand und 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);
    }
}
```

Das folgende Snippet verwendet den AWS SDK for JavaScript V2-Stil. Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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}");
        }
```

Daten schreiben (Einfügungen und Upserts)

Themen

- Stapel von Datensätzen schreiben
- Schreiben von Datensatzstapeln mit gemeinsamen Attributen
- Datensätze werden aktualisiert
- Beispiel für ein Attribut mit mehreren Kennzahlen
- Behandlung von Schreibfehlern

Stapel von Datensätzen schreiben

Sie können die folgenden Codefragmente verwenden, um Daten in eine Amazon Timestream-Tabelle zu schreiben. Das Schreiben von Daten in Batches hilft dabei, die Schreibkosten zu optimieren. Weitere Informationen finden Sie unter Berechnung der Anzahl der Schreibvorgänge.

> **Note**
>
> Diese Codefragmente basieren auf vollständigen Beispielanwendungen auf. GitHub Weitere Informationen zu den ersten Schritten mit den Beispielanwendungen finden Sie unter. Beispielanwendung

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

Der folgende Codeausschnitt verwendet den AWS SDK for JavaScript V2-Stil. Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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());
    }
  }
```

Schreiben von Datensatzstapeln mit gemeinsamen Attributen

Wenn Ihre Zeitreihendaten Kennzahlen und/oder Dimensionen aufweisen, die vielen Datenpunkten gemeinsam sind, können Sie auch die folgende optimierte Version von verwenden, writeRecords API um Daten in Timestream for einzufügen. LiveAnalytics Durch die Verwendung von gemeinsamen Attributen bei der Batchverarbeitung können die Schreibkosten weiter optimiert werden, wie unter beschrieben. Berechnung der Anzahl der Schreibvorgänge

> **Note**
>
> Diese Codefragmente basieren auf vollständigen Beispielanwendungen auf. GitHub Weitere Informationen zu den ersten Schritten mit den Beispielanwendungen finden Sie unter. Beispielanwendung

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

Der folgende Codeausschnitt verwendet den AWS SDK for JavaScript V2-Stil. Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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());
    }
  }
```

Datensätze werden aktualisiert

Während die Standardschreibvorgänge in Amazon Timestream der Semantik des ersten Writers folgen, bei der Daten nur als Anhängen gespeichert werden und doppelte Datensätze zurückgewiesen werden, gibt es Anwendungen, für die die Fähigkeit erforderlich ist, Daten mithilfe der Semantik des letzten Writers in Amazon Timestream zu schreiben, bei denen der Datensatz mit der höchsten Version im System gespeichert wird. Es gibt auch Anwendungen, die die Möglichkeit erfordern, bestehende Datensätze zu aktualisieren. Um diesen Szenarien zu begegnen, bietet Amazon Timestream die Möglichkeit, Daten zu ändern. Upsert ist eine Operation, die einen Datensatz in das System einfügt, wenn der Datensatz nicht existiert, oder den Datensatz aktualisiert, falls einer existiert.

Sie können Datensätze aktualisieren, indem Sie sie beim Senden einer `Version` `WriteRecords` Anfrage in die Datensatzdefinition aufnehmen. Amazon Timestream speichert den Datensatz mit dem höchsten Wert`Version`. Das folgende Codebeispiel zeigt, wie Sie Daten ändern können:

> **Note**
>
> Diese Codefragmente basieren auf vollständigen Beispielanwendungen auf. GitHub Weitere Informationen zu den ersten Schritten mit den Beispielanwendungen finden Sie unter. Beispielanwendung

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

Der folgende Codeausschnitt verwendet den AWS SDK for JavaScript V2-Stil. Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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());
    }
```

```
}
```

Beispiel für ein Attribut mit mehreren Kennzahlen

Dieses Beispiel veranschaulicht das Schreiben von Attributen mit mehreren Kennzahlen. Attribute mit mehreren Kennzahlen sind nützlich, wenn ein Gerät oder eine Anwendung, die Sie verfolgen, mehrere Messwerte oder Ereignisse gleichzeitig ausgibt.

> **Note**
>
> Diese Codefragmente basieren auf vollständigen Beispielanwendungen auf. GitHub Weitere Informationen zu den ersten Schritten mit den Beispielanwendungen finden Sie unter. Beispielanwendung

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

Der folgende Codeausschnitt verwendet den AWS SDK for JavaScript V2-Stil. Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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());
      }
    }
  }
}
```

Behandlung von Schreibfehlern

Schreibvorgänge in Amazon Timestream können aus einem oder mehreren der folgenden Gründe fehlschlagen:

- Es gibt Datensätze mit Zeitstempeln, die außerhalb der Aufbewahrungsdauer des Speicherspeichers liegen.
- Es gibt Datensätze, die Dimensionen und/oder Kennzahlen enthalten, die die von Timestream definierten Grenzwerte überschreiten.
- Amazon Timestream hat doppelte Datensätze erkannt. Datensätze werden als doppelt markiert, wenn es mehrere Datensätze mit denselben Dimensionen, Zeitstempeln und Kennzahlnamen gibt, aber:
  - Die Kennzahlwerte sind unterschiedlich.
  - Version ist in der Anforderung nicht vorhanden, oder der Wert der Version im neuen Datensatz ist gleich oder niedriger als der vorhandene Wert. Wenn Amazon Timestream Daten aus diesem Grund ablehnt, enthält das `ExistingVersion` Feld in der die `RejectedRecords` aktuelle Version des Datensatzes, wie sie in Amazon Timestream gespeichert ist. Um eine Aktualisierung zu erzwingen, können Sie die Anfrage erneut senden, wobei eine Version für den Datensatz auf einen Wert gesetzt ist, der größer als der ist. `ExistingVersion`

Weitere Informationen zu Fehlern und abgelehnten Datensätzen finden Sie unter Fehler und RejectedRecord.

Wenn Ihre Anwendung `RejectedRecordsException` beim Versuch, Datensätze in Timestream zu schreiben, eine Meldung erhält, können Sie die abgelehnten Datensätze analysieren, um mehr über die Schreibfehler zu erfahren, wie unten dargestellt.

> **Note**
>
> Diese Codefragmente basieren auf vollständigen Beispielanwendungen auf. GitHub Weitere Informationen zu den ersten Schritten mit den Beispielanwendungen finden Sie unter. Beispielanwendung

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

Der folgende Codeausschnitt verwendet den AWS SDK for JavaScript V2-Stil. Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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());
 }
```

Abfrage ausführen

Themen

- Ergebnisse paginieren
- Analysieren von Ergebnismengen

- Zugriff auf den Abfragestatus

Ergebnisse paginieren

Wenn Sie eine Abfrage ausführen, gibt Timestream die Ergebnismenge paginiert zurück, um die Reaktionsfähigkeit Ihrer Anwendungen zu optimieren. Der folgende Codeausschnitt zeigt, wie Sie den Ergebnissatz paginieren können. Sie müssen alle Seiten der Ergebnismenge in einer Schleife durchgehen, bis Sie auf einen Nullwert stoßen. Paginierungstoken laufen 3 Stunden ab, nachdem sie von Timestream für ausgestellt wurden. LiveAnalytics

> **Note**
>
> Diese Codefragmente basieren auf vollständigen Beispielanwendungen auf. GitHub Weitere Informationen zu den ersten Schritten mit den Beispielanwendungen finden Sie unter. Beispielanwendung

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

Der folgende Codeausschnitt verwendet den AWS SDK for JavaScript V2-Stil. Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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());
```

```
            }
        }
```

Analysieren von Ergebnismengen

Sie können die folgenden Codefragmente verwenden, um Daten aus der Ergebnismenge zu extrahieren. Auf die Abfrageergebnisse kann bis zu 24 Stunden nach Abschluss einer Abfrage zugegriffen werden.

> **Note**
>
> Diese Codefragmente basieren auf vollständigen Beispielanwendungen auf. GitHub Weitere Informationen zu den ersten Schritten mit den Beispielanwendungen finden Sie unter. Beispielanwendung

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

Der folgende Codeausschnitt verwendet den AWS SDK for JavaScript V2-Stil. Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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)}]";
        }
```

Zugriff auf den Abfragestatus

Sie können auf den Abfragestatus zugreifen`QueryResponse`, der Informationen über den Fortschritt einer Abfrage, die von einer Abfrage gescannten Byte und die von einer Abfrage gemessenen Byte enthält. Die `bytesScanned` Werte `bytesMetered` und sind kumulativ und werden beim Paging der Abfrageergebnisse kontinuierlich aktualisiert. Sie können diese Informationen verwenden, um zu verstehen, welche Byte von einer einzelnen Abfrage gescannt wurden, und anhand dieser Informationen auch bestimmte Entscheidungen treffen. Wenn Sie beispielsweise annehmen, dass der Abfragepreis 0,01 USD pro gescanntem GB beträgt, möchten Sie möglicherweise Abfragen stornieren, die 25 USD pro Abfrage oder `X` GB überschreiten. Der folgende Codeausschnitt zeigt, wie das gemacht werden kann.

> **Note**
>
> Diese Codefragmente basieren auf vollständigen Beispielanwendungen auf. GitHub Weitere Informationen zu den ersten Schritten mit den Beispielanwendungen finden Sie unter. Beispielanwendung

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

Der folgende Codeausschnitt verwendet den AWS SDK for JavaScript V2-Stil. Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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());
    }
}
```

Weitere Informationen zum Stornieren einer Abfrage finden Sie unterAnfrage abbrechen.

UNLOADAbfrage ausführen

In den folgenden Codebeispielen wird eine UNLOAD Abfrage aufgerufen. Weitere Informationen zu UNLOAD finden Sie unter Wird verwendetUNLOAD, um Abfrageergebnisse von Timestream nach S3 zu exportieren für LiveAnalytics. Beispiele für UNLOAD Abfragen finden Sie unterBeispiel für einen Anwendungsfall UNLOAD von Timestream für LiveAnalytics.

Themen

- Erstellen Sie eine UNLOAD Abfrage und führen Sie sie aus
- Analysieren Sie die UNLOAD Antwort und rufen Sie die Zeilenanzahl, den Manifest-Link und den Metadaten-Link ab
- Lesen und analysieren Sie den Inhalt des Manifests
- Lesen und analysieren Sie den Inhalt von Metadaten

Erstellen Sie eine UNLOAD Abfrage und führen Sie sie aus

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
    }
}
```

Analysieren Sie die UNLOAD Antwort und rufen Sie die Zeilenanzahl, den Manifest-Link und den Metadaten-Link ab

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
}
```

Lesen und analysieren Sie den Inhalt des Manifests

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};
}
```

Lesen und analysieren Sie den Inhalt von Metadaten

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};
}
```

Anfrage abbrechen

Sie können die folgenden Codefragmente verwenden, um eine Abfrage abzubrechen.

> **Note**
>
> Diese Codefragmente basieren auf vollständigen Beispielanwendungen auf. GitHub Weitere Informationen zu den ersten Schritten mit den Beispielanwendungen finden Sie unter. Beispielanwendung

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

Der folgende Codeausschnitt verwendet den AWS SDK for JavaScript V2-Stil. Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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);
            }
        }
```

Batch-Load-Aufgabe erstellen

Sie können die folgenden Codefragmente verwenden, um Batch-Load-Aufgaben zu erstellen.

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

Der folgende Codeausschnitt wird für Version 3 verwendet. AWS SDK JavaScript Weitere Informationen zur Installation und Verwendung des Clients finden Sie unter Timestream Write Client — AWS SDK for v3. JavaScript

APIEinzelheiten finden Sie unter Klasse CreateBatchLoadCommand und 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();
        }
    }
}
```

Beschreiben Sie die Batch-Load-Aufgabe

Sie können die folgenden Codefragmente verwenden, um Batch-Load-Aufgaben zu beschreiben.

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

Der folgende Codeausschnitt wird für Version 3 verwendet. AWS SDK JavaScript Weitere Informationen zur Installation und Verwendung des Clients finden Sie unter Timestream Write Client — AWS SDK for v3. JavaScript

APIEinzelheiten finden Sie unter Klasse DescribeBatchLoadCommand und 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>");
        }
    }
}
```

Batch-Load-Aufgaben auflisten

Sie können die folgenden Codefragmente verwenden, um Batch-Load-Aufgaben aufzulisten.

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

Der folgende Codeausschnitt wird für Version 3 verwendet. AWS SDK JavaScript Weitere Informationen zur Installation und Verwendung des Clients finden Sie unter Timestream Write Client — AWS SDK for v3. JavaScript

APIEinzelheiten finden Sie unter Klasse DescribeBatchLoadCommand und 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();
        }
    }
}
```

Batch-Load-Aufgabe fortsetzen

Sie können die folgenden Codefragmente verwenden, um Batch-Ladeaufgaben wieder aufzunehmen.

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

Der folgende Codeausschnitt wird für Version 3 verwendet. AWS SDK JavaScript Weitere Informationen zur Installation und Verwendung des Clients finden Sie unter Timestream Write Client — AWS SDK for v3. JavaScript

APIEinzelheiten finden Sie unter Klasse CreateBatchLoadCommand und 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());
            }
        }
    }
}
```

Geplante Abfrage erstellen

Sie können die folgenden Codefragmente verwenden, um eine geplante Abfrage mit Multi-Measure-Mapping zu erstellen.

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

Der folgende Codeausschnitt verwendet den for V2-Stil. AWS SDK JavaScript Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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;
    }
}
```

Geplante Abfragen auflisten

Sie können die folgenden Codefragmente verwenden, um Ihre geplanten Abfragen aufzulisten.

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

Der folgende Codeausschnitt verwendet den AWS SDK for V2-Stil. JavaScript Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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;
    }
 }
```

Beschreiben Sie die geplante Abfrage

Sie können die folgenden Codefragmente verwenden, um eine geplante Abfrage zu beschreiben.

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

Der folgende Codeausschnitt verwendet den AWS SDK for V2-Stil. JavaScript Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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;
    }
}
```

Geplante Abfrage ausführen

Sie können die folgenden Codefragmente verwenden, um eine geplante Abfrage auszuführen.

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

Der folgende Codeausschnitt verwendet den AWS SDK for V2-Stil. JavaScript Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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;
        }
    }
```

Geplante Abfrage aktualisieren

Sie können die folgenden Codefragmente verwenden, um eine geplante Abfrage zu aktualisieren.

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

Der folgende Codeausschnitt verwendet den AWS SDK for V2-Stil. JavaScript Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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;
    }
}
```

Geplante Abfrage löschen

Sie können die folgenden Codefragmente verwenden, um eine geplante Abfrage zu löschen.

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

Der folgende Codeausschnitt verwendet den AWS SDK for V2-Stil. JavaScript Es basiert auf der Beispielanwendung unter Node.js Beispiel Amazon Timestream für die LiveAnalytics Anwendung auf 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;
    }
}
```

Verwenden von Batch-Load in Timestream für LiveAnalytics

Mit Batch-Load für Amazon Timestream for LiveAnalytics können Sie in Amazon S3 gespeicherte CSV Dateien stapelweise in Timestream aufnehmen. Mit dieser neuen Funktion können Sie Ihre Daten in Timestream speichern, LiveAnalytics ohne sich auf andere Tools verlassen oder benutzerdefinierten Code schreiben zu müssen. Sie können Batch-Load verwenden, um Daten mit flexiblen Wartezeiten aufzufüllen, z. B. Daten, die nicht sofort für Abfragen oder Analysen benötigt werden.

Sie können Batch-Load-Aufgaben erstellen, indem Sie die AWS Management Console AWS CLI, und die verwenden. AWS SDKs Weitere Informationen finden Sie unter Verwenden von Batch-Load mit der Konsole, Verwenden von Batch-Load mit AWS CLI und Verwenden von Batch-Load mit AWS SDKs.

Zusätzlich zum Batch-Load können Sie während des WriteRecords API Vorgangs mehrere Datensätze gleichzeitig schreiben. Hinweise zur Verwendung finden Sie unterWählen Sie zwischen WriteRecords API Operation und Batch-Load.

Themen

- Konzepte zum Laden von Batch in Timestream
- Voraussetzungen für das Batch-Laden
- Bewährte Methoden zum Laden von Batch
- Eine Batch-Load-Datendatei wird vorbereitet
- Datenmodellzuordnungen für das Batch-Laden
- Verwenden von Batch-Load mit der Konsole

- Verwenden von Batch-Load mit AWS CLI
- Verwenden von Batch-Load mit AWS SDKs
- Verwenden von Batch-Load-Fehlerberichten

Konzepte zum Laden von Batch in Timestream

Sehen Sie sich die folgenden Konzepte an, um die Batch-Load-Funktionalität besser zu verstehen.

Batch-Load-Aufgabe — Die Aufgabe, die Ihre Quelldaten und Ihr Ziel in Amazon Timestream definiert. Sie geben zusätzliche Konfigurationen wie das Datenmodell an, wenn Sie die Batch-Load-Aufgabe erstellen. Sie können Batch-Load-Tasks über die AWS Management Console AWS CLI, die und die erstellen AWS SDKs.

Importziel — Die Zieldatenbank und die Zieltabelle in Timestream. Hinweise zum Erstellen von Datenbanken und Tabellen finden Sie unter Erstellen einer -Datenbank undErstellen einer Tabelle.

Datenquelle — Die CSV Quelldatei, die in einem S3-Bucket gespeichert ist. Hinweise zur Vorbereitung der Datendatei finden Sie unterEine Batch-Load-Datendatei wird vorbereitet. Informationen zu den S3-Preisen finden Sie unter Amazon S3 S3-Preise.

Batch-Load-Fehlerbericht — Ein Bericht, der Informationen über die Fehler einer Batch-Load-Aufgabe speichert. Sie definieren den S3-Speicherort für Batch-Load-Fehlerberichte als Teil einer Batch-Load-Aufgabe. Informationen zu den Informationen in den Berichten finden Sie unterVerwenden von Batch-Load-Fehlerberichten.

Datenmodell-Mapping — Eine Batch-Load-Zuordnung für Zeit, Dimensionen und Kennzahlen, die von einer Datenquelle an einem S3-Standort zu einem Ziel-Timestream für eine LiveAnalytics Tabelle übertragen wird. Weitere Informationen finden Sie unter Datenmodellzuordnungen für das Batch-Laden.

Voraussetzungen für das Batch-Laden

Dies ist eine Liste der Voraussetzungen für die Verwendung von Batch-Load. Bewährte Methoden finden Sie unter Bewährte Methoden zum Laden von Batch.

- Batch-Load-Quelldaten werden in Amazon S3 im CSV Format mit Headern gespeichert.
- Für jeden Amazon S3 S3-Quell-Bucket müssen Sie in einer beigefügten Richtlinie über die folgenden Berechtigungen verfügen:

```
"s3:GetObject",
"s3:GetBucketAcl"
"s3:ListBucket"
```

Ebenso müssen Sie für jeden Amazon S3 S3-Ausgabe-Bucket, in den Berichte geschrieben werden, in einer angehängten Richtlinie über die folgenden Berechtigungen verfügen:

```
"s3:PutObject",
"s3:GetBucketAcl"
```

Beispielsweise:

```
{
  "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 for LiveAnalytics analysiert die Daten, CSV indem es die im Datenmodell bereitgestellten Informationen den Headern CSV zuordnet. Die Daten müssen eine Spalte haben, die den Zeitstempel darstellt, mindestens eine Dimensionsspalte und mindestens eine Kennzahlspalte.
- Die beim Batch-Load verwendeten S3-Buckets müssen sich in derselben Region befinden und aus demselben Konto stammen wie der Timestream für die LiveAnalytics Tabelle, der beim Batch-Load verwendet wird.
- Die `timestamp` Spalte muss einen langen Datentyp haben, der die Zeit seit der Unix-Epoche darstellt. Der Zeitstempel `2021-03-25T08:45:21Z` würde beispielsweise als dargestellt werden. `1616661921` Timestream unterstützt Sekunden, Millisekunden, Mikrosekunden und Nanosekunden für die Genauigkeit des Zeitstempels. Wenn Sie die Abfragesprache verwenden, können Sie zwischen Formaten mit Funktionen wie konvertieren. `to_unixtime` Weitere Informationen finden Sie unter Funktionen für Datum und Uhrzeit.
- Timestream unterstützt den String-Datentyp für Dimensionswerte. Es unterstützt die Datentypen Long, Double, String und Boolean für Kennzahlspalten.

Informationen zu Ladebeschränkungen und Kontingenten für Batches finden Sie unter. Batch-Laden

Bewährte Methoden zum Laden von Batch

Die Chargenbeladung funktioniert am besten (hoher Durchsatz), wenn die folgenden Bedingungen und Empfehlungen eingehalten werden:

1. CSVDie zur Aufnahme eingereichten Dateien sind klein, insbesondere mit einer Dateigröße von 100 MB bis 1 GB, um die Parallelität und Geschwindigkeit der Aufnahme zu verbessern.
2. Vermeiden Sie es, während des Batchladevorgangs gleichzeitig Daten in dieselbe Tabelle aufzunehmen (z. B. mithilfe des WriteRecords API Vorgangs oder einer geplanten Abfrage). Dies kann zu Drosselungen führen, und der Batch-Ladevorgang schlägt fehl.
3. Fügen Sie dem S3-Bucket, der beim Batch-Load verwendet wird, keine Dateien hinzu, ändern oder entfernen Sie keine Dateien aus dem S3-Bucket, während die Batch-Load-Task ausgeführt wird.
4. Löschen oder entziehen Sie keine Berechtigungen für Tabellen oder Quellen und melden Sie keine S3-Buckets, die geplante oder laufende Batch-Ladeaufgaben enthalten.
5. Wenn Sie Daten mit einem Satz von Dimensionswerten mit hoher Kardinalität aufnehmen, folgen Sie den Anweisungen unter. Empfehlungen für die Partitionierung von Datensätzen mit mehreren Kennzahlen

6. Stellen Sie sicher, dass Sie die Daten auf ihre Richtigkeit testen, indem Sie eine kleine Datei einreichen. Alle Daten, die beim Batch-Load eingereicht werden, werden Ihnen unabhängig von ihrer Richtigkeit in Rechnung gestellt. Weitere Informationen zur Preisgestaltung finden Sie unter Amazon Timestream Timestream-Preise.
7. Nehmen Sie eine Batch-Ladeaufgabe nicht wieder auf, es sei denn, Sie `ActiveMagneticStorePartitions` haben weniger als 250. Der Job ist möglicherweise gedrosselt und schlägt fehl. Wenn Sie mehrere Jobs gleichzeitig für dieselbe Datenbank einreichen, sollte sich die Anzahl verringern.

Im Folgenden finden Sie bewährte Methoden für Konsolen:

1. Verwenden Sie den Builder nur für eine einfachere Datenmodellierung, bei der nur ein Kennzahlname für Datensätze mit mehreren Kennzahlen verwendet wird.
2. Verwenden JSON Sie für eine komplexere Datenmodellierung. Verwenden Sie dies beispielsweise, JSON wenn Sie mehrere Kennzahlnamen verwenden, wenn Sie Datensätze mit mehreren Kennzahlen verwenden.

Weitere Informationen zu LiveAnalytics Best Practices in Timestream finden Sie unter. Bewährte Methoden

Eine Batch-Load-Datendatei wird vorbereitet

Eine Quelldatendatei hat durch Trennzeichen getrennte Werte. Der spezifischere Begriff, kommagetrennte Werte ()CSV, wird allgemein verwendet. Zu den gültigen Spaltentrennzeichen gehören Kommas und senkrechte Linien. Datensätze werden durch neue Zeilen getrennt. Dateien müssen in Amazon S3 gespeichert werden. Wenn Sie eine neue Batch-Load-Aufgabe erstellen, wird der Speicherort der Quelldaten durch ein ARN für die Datei angegeben. Eine Datei enthält Header. Eine Spalte steht für den Zeitstempel. Mindestens eine weitere Spalte steht für eine Kennzahl.

Die beim Batch-Load verwendeten S3-Buckets müssen sich in derselben Region befinden wie der Timestream für die LiveAnalytics Tabelle, die beim Batch-Load verwendet wird. Fügen Sie dem S3-Bucket, der beim Batch-Laden verwendet wird, keine Dateien hinzu oder entfernen Sie sie aus diesem, nachdem die Batch-Load-Aufgabe gesendet wurde. Informationen zur Arbeit mit S3-Buckets finden Sie unter Erste Schritte mit Amazon S3.

> **Note**
>
> CSVDateien, die von einigen Anwendungen wie Excel generiert werden, enthalten möglicherweise eine Markierung für die Bytereihenfolge (BOM), die mit der erwarteten Kodierung in Konflikt steht. Timestream für LiveAnalytics Batch-Load-Aufgaben, die auf eine CSV Datei verweisen und bei ihrer programmgesteuerten Verarbeitung einen Fehler BOM auslösen. Um dies zu vermeiden, können Sie das, ein BOM unsichtbares Zeichen, entfernen. Sie können die Datei beispielsweise aus einer Anwendung wie Notepad++ speichern, mit der Sie eine neue Kodierung angeben können. Sie können auch eine programmatische Option verwenden, die die erste Zeile liest, das Zeichen aus der Zeile entfernt und den neuen Wert über die erste Zeile in der Datei schreibt.
> Beim Speichern aus Excel gibt es mehrere CSV Optionen. Das Speichern mit einer anderen CSV Option könnte das beschriebene Problem verhindern. Sie sollten das Ergebnis jedoch überprüfen, da sich eine Änderung der Kodierung auf einige Zeichen auswirken kann.

CSVParameter formatieren

Sie verwenden Escape-Zeichen, wenn Sie einen Wert darstellen, der ansonsten durch die Formatparameter reserviert ist. Wenn das Anführungszeichen beispielsweise ein doppeltes Anführungszeichen ist, setzen Sie das Escape-Zeichen vor das doppelte Anführungszeichen, um ein doppeltes Anführungszeichen in den Daten darzustellen.

Informationen darüber, wann Sie diese angeben sollten, wenn Sie eine Batch-Load-Task erstellen, finden Sie unterErstellen Sie eine Batch-Load-Aufgabe.

Parameter	Optionen
Spaltentrennzeichen	(Komma (',') | Pipe ('|') | Semikolon (';') | Tab ('/t') | Leerzeichen ('))
Escape-Zeichen	Keine
Charakter zitieren	Konsole: (Doppeltes Anführungszeichen („) | Einfaches Anführungszeichen ('))
Nullwert	Leerzeichen (')

Parameter	Optionen
Leerraum abschneiden	Konsole: (Nein | Ja)

Datenmodellzuordnungen für das Batch-Laden

Im Folgenden wird das Schema für Datenmodellzuordnungen beschrieben und ein Beispiel gegeben.

Schema für Datenmodellzuordnungen

Die `CreateBatchLoadTask` Anforderungssyntax und ein `BatchLoadTaskDescription` Objekt, das durch einen Aufruf zurückgegeben wird, um ein `DataModelConfiguration` Objekt `DescribeBatchLoadTask` einzuschließen, das das `DataModel` für das Batch-Laden einschließt. Das `DataModel` definiert Zuordnungen von Quelldaten, die im CSV Format an einem S3-Speicherort gespeichert sind, zu einem Ziel-Timestream für LiveAnalytics Datenbank und Tabelle.

Das `TimeColumn` Feld gibt den Speicherort der Quelldaten für den Wert an, der der `time` Spalte der Zieltabelle in Timestream for zugeordnet werden soll. LiveAnalytics Das `TimeUnit` gibt die Einheit für`TimeColumn`, an und kann eine von`MILLISECONDS`, `SECONDSMICROSECONDS`, oder sein. `NANOSECONDS` Es gibt auch Zuordnungen für Dimensionen und Kennzahlen. Dimensionszuordnungen bestehen aus Quellspalten und Zielfeldern.

Weitere Informationen finden Sie unter. DimensionMapping Bei den Zuordnungen für Kennzahlen gibt es zwei Optionen: und`MixedMeasureMappings`. `MultiMeasureMappings`

Zusammenfassend lässt sich sagen, dass a Zuordnungen von einer Datenquelle an einem S3-Standort zu einem Ziel-Timestream für die folgende Tabelle `DataModel` enthält. LiveAnalytics

- Zeit
- Dimensionen
- Maßnahmen

Wenn möglich, empfehlen wir, dass Sie Messdaten Datensätzen mit mehreren Messwerten in Timestream for zuordnen. LiveAnalytics Informationen zu den Vorteilen von Datensätzen mit mehreren Kennzahlen finden Sie unter. Datensätze mit mehreren Messwerten

Wenn mehrere Kennzahlen in den Quelldaten in einer Zeile gespeichert sind, können Sie diese mehreren Kennzahlen zur Verwendung den Datensätzen mit mehreren Kennzahlen in Timestream

zuordnen. LiveAnalytics `MultiMeasureMappings` Wenn es Werte gibt, die einem Datensatz mit einer einzigen Kennzahl zugeordnet werden müssen, können Sie Folgendes verwenden. `MixedMeasureMappings`

`MixedMeasureMappings`sund `MultiMeasureMappings` beide beinhalten`MultiMeasureAttributeMappings`. Datensätze mit mehreren Messwerten werden unabhängig davon unterstützt, ob Datensätze mit einer einzigen Kennzahl benötigt werden.

Wenn in Timestream for nur Zieldatensätze mit mehreren Kennzahlen benötigt werden LiveAnalytics, können Sie Kennzahlzuordnungen in der folgenden Struktur definieren.

```
CreateBatchLoadTask
    MeasureNameColumn
    MultiMeasureMappings
        TargetMultiMeasureName
        MultiMeasureAttributeMappings array
```

> **Note**
>
> Wir empfehlen, wann immer möglich zu verwenden. `MultiMeasureMappings`

Wenn in Timestream für Einzelkennzahlen Zieldatensätze benötigt werden LiveAnalytics, können Sie Kennzahlzuordnungen in der folgenden Struktur definieren.

```
CreateBatchLoadTask
    MeasureNameColumn
    MixedMeasureMappings array
        MixedMeasureMapping
            MeasureName
            MeasureValueType
            SourceColumn
            TargetMeasureName
            MultiMeasureAttributeMappings array
```

Wenn Sie es verwenden`MultiMeasureMappings`, ist das `MultiMeasureAttributeMappings` Array immer erforderlich. Wenn Sie das `MixedMeasureMappings` Array verwenden, `MeasureValueType` ist es `MULTI` dafür erforderlich`MixedMeasureMapping`, wenn es für ein bestimmtes Objekt `MultiMeasureAttributeMappings` ist`MixedMeasureMapping`. Andernfalls `MeasureValueType` gibt es den Messtyp für den Datensatz mit einer einzelnen Kennzahl an.

In beiden Fällen steht eine Reihe von `MultiMeasureAttributeMapping` verfügbaren zur Verfügung. Sie definieren die Zuordnungen zu Datensätzen mit mehreren Kennzahlen in den einzelnen `MultiMeasureAttributeMapping` Datensätzen wie folgt:

`SourceColumn`

Die Spalte in den Quelldaten, die sich in Amazon S3 befindet.

`TargetMultiMeasureAttributeName`

Der Name des Ziel-Multi-Measure-Namens in der Zieltabelle. Diese Eingabe ist erforderlich, wenn sie nicht angegeben `MeasureNameColumn` wird. Wenn angegeben, `MeasureNameColumn` wird der Wert aus dieser Spalte als Name für mehrere Kennzahlen verwendet.

`MeasureValueType`

Einer von`DOUBLE`, `BIGINT` `BOOLEAN``VARCHAR`, oder`TIMESTAMP`.

Datenmodell-Mappings mit Beispiel **`MultiMeasureMappings`**

In diesem Beispiel wird die Zuordnung zu Datensätzen mit mehreren Kennzahlen demonstriert, der bevorzugte Ansatz, bei dem jeder Messwert in einer eigenen Spalte gespeichert wird. Sie können ein Beispiel CSV unter Beispiel CSV herunterladen. Das Beispiel hat die folgenden Überschriften, die einer Zielspalte in einer LiveAnalytics Timestream-für-Tabelle zugeordnet werden können.

- `time`
- `measure_name`
- `region`
- `location`
- `hostname`
- `memory_utilization`
- `cpu_utilization`

Identifizieren Sie die `measure_name` Spalten `time` und in der CSV Datei. In diesem Fall werden diese direkt dem Timestream für LiveAnalytics Tabellenspalten mit demselben Namen zugeordnet.

- `time`ordnet zu `time`
- `measure_name`ordnet zu `measure_name` (oder Ihrem ausgewählten Wert)

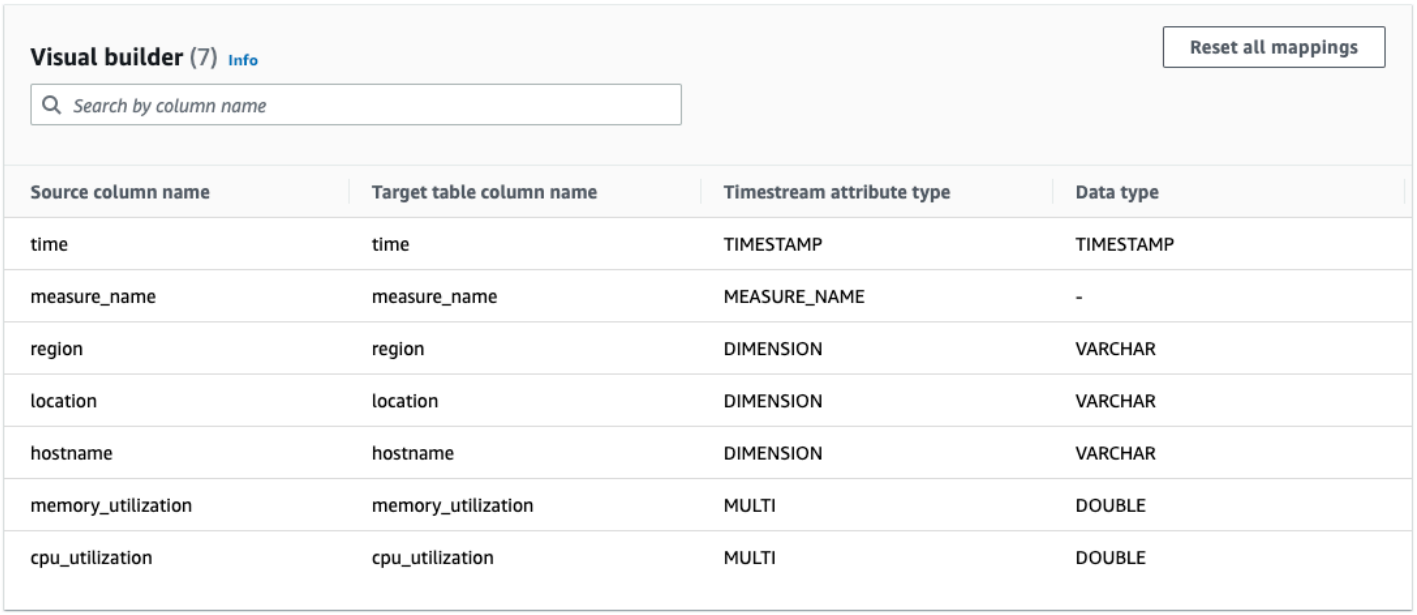
Wenn Sie den verwendenAPI, geben Sie `time` im `TimeColumn` Feld einen unterstützten Wert für die Zeiteinheit an, z. B. `MILLISECONDS` im `TimeUnit` Feld. Diese entsprechen dem Namen der Quellspalte und dem Zeitstempel, der in der Konsole eingegeben wurde. Sie können Datensätze gruppieren oder partitionieren, indem Sie `measure_name` das verwenden, was mit dem `MeasureNameColumn` Schlüssel definiert ist.

In der Stichprobe `hostname` sind `regionlocation`,, und Dimensionen. Dimensionen werden einer Reihe von `DimensionMapping` Objekten zugeordnet.

Bei Kennzahlen `TargetMultiMeasureAttributeName` wird der Wert zu einer Spalte in der Timestream-For-Tabelle. LiveAnalytics Sie können den gleichen Namen wie in diesem Beispiel beibehalten. Oder Sie können einen neuen angeben. `MeasureValueType`ist einer von `DOUBLE`,`BIGINT`,`BOOLEAN`,`VARCHAR`, oder`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"
      }
    ]
  }
}
```



Datenmodell-Mappings mit Beispiel `MixedMeasureMappings`

Wir empfehlen, diesen Ansatz nur zu verwenden, wenn Sie Datensätze mit einer einzigen Kennzahl in Timestream for zuordnen müssen. LiveAnalytics

Verwenden von Batch-Load mit der Konsole

Im Folgenden finden Sie die Schritte zur Verwendung von Batch Load mit dem AWS Management Console. Sie können ein Beispiel CSV unter Beispiel herunterladenCSV.

Themen

- Auf Batch-Load zugreifen
- Erstellen Sie eine Batch-Load-Aufgabe
- Setzen Sie eine Batch-Ladeaufgabe fort
- Verwenden des Visual Builders

Auf Batch-Load zugreifen

Gehen Sie wie folgt vor, um mit dem auf Batch-Load zuzugreifen AWS Management Console.

1. Öffnen Sie die Amazon Timestream Timestream-Konsole.
2. Wählen Sie im Navigationsbereich Management Tools und dann Batch-Load-Tasks aus.
3. Von hier aus können Sie sich die Liste der Batch-Load-Aufgaben ansehen und weitere Informationen zu einer bestimmten Aufgabe abrufen. Sie können auch Aufgaben erstellen und fortsetzen.

Erstellen Sie eine Batch-Load-Aufgabe

Gehen Sie wie folgt vor, um eine Batch-Load-Task mit dem zu erstellen AWS Management Console.

1. Öffnen Sie die Amazon Timestream Timestream-Konsole.
2. Wählen Sie im Navigationsbereich Management Tools und dann Batch-Load-Tasks aus.
3. Wählen Sie Batch-Load-Task erstellen aus.
4. Wählen Sie unter Importziel Folgendes aus.

   - Zieldatenbank — Wählen Sie den Namen der Datenbank aus, die in erstellt wurdeErstellen einer -Datenbank.
   - Zieltabelle — Wählen Sie den Namen der Tabelle aus, die in erstellt wurdeErstellen einer Tabelle.

   Bei Bedarf können Sie in diesem Bereich mit der Schaltfläche Neue Tabelle erstellen eine Tabelle hinzufügen.
5. Wählen Sie unter Datenquelle unter Speicherort der Datenquelle den S3-Bucket aus, in dem die Quelldaten gespeichert sind. Verwenden Sie die Schaltfläche S3 durchsuchen, um S3-Ressourcen anzuzeigen, auf die das aktive AWS Konto Zugriff hat, oder geben Sie den S3-Standort einURL. Die Datenquelle muss sich in derselben Region befinden.
6. In den Dateiformateinstellungen (erweiterbarer Abschnitt) können Sie die Standardeinstellungen verwenden, um Eingabedaten zu analysieren. Sie können auch Erweiterte Einstellungen wählen. Von dort aus können Sie CSVFormatparameter und Parameter zum Analysieren von Eingabedaten auswählen. Informationen zu diesen Parametern finden Sie unterCSVParameter formatieren.

7. Konfigurieren Sie unter Datenmodellzuordnung konfigurieren das Datenmodell. Weitere Anleitungen zum Datenmodell finden Sie unter Datenmodellzuordnungen für das Batch-Laden

   - Wählen Sie unter Datenmodell-Mapping die Option Mapping-Konfigurationseingabe und dann eine der folgenden Optionen aus.
     - Visual Builder — Um Daten visuell zuzuordnen, wählen Sie TargetMultiMeasureNameoder MeasureNameColumn. Ordnen Sie dann in Visual Builder die Spalten zu.

       Visual Builder erkennt und lädt automatisch die Quellspaltenüberschriften aus der Datenquellendatei, wenn eine einzelne CSV Datei als Datenquelle ausgewählt wird. Wählen Sie das Attribut und den Datentyp aus, um Ihre Zuordnung zu erstellen.

       Informationen zur Verwendung von Visual Builder finden Sie unterVerwenden des Visual Builders.
     - JSONeditor — Ein JSON Freiform-Editor zur Konfiguration Ihres Datenmodells. Wählen Sie diese Option, wenn Sie mit Timestream for vertraut sind LiveAnalytics und erweiterte Datenmodell-Mappings erstellen möchten.
     - JSONDatei aus S3 — Wählen Sie eine JSON Modelldatei aus, die Sie in S3 gespeichert haben. Wählen Sie diese Option, wenn Sie bereits ein Datenmodell konfiguriert haben und es für weitere Batch-Loads wiederverwenden möchten.
8. Wählen Sie im Fehlerprotokollbericht unter S3-Speicherort der Fehlerprotokolle den S3-Speicherort aus, der für die Meldung von Fehlern verwendet werden soll. Informationen zur Verwendung dieses Berichts finden Sie unterVerwenden von Batch-Load-Fehlerberichten.
9. Wählen Sie für den Typ des Verschlüsselungsschlüssels eine der folgenden Optionen aus.

   - Von Amazon SSE S3 verwalteter Schlüssel (-S3) — Ein Verschlüsselungsschlüssel, den Amazon S3 für Sie erstellt, verwaltet und verwendet.
   - AWS KMS key (SSE-KMS) — Ein durch AWS Key Management Service () geschützter Verschlüsselungsschlüssel.AWS KMS
10. Wählen Sie Weiter.
11. Überprüfen Sie auf der Seite Überprüfen und erstellen die Einstellungen und bearbeiten Sie sie gegebenenfalls.

> **Note**
>
> Sie können die Einstellungen für Batch-Load-Aufgaben nicht mehr ändern, nachdem die Aufgabe erstellt wurde. Die Bearbeitungszeiten für Aufgaben hängen von der Menge der importierten Daten ab.

12. Wählen Sie Batch-Load-Task erstellen aus.

Setzen Sie eine Batch-Ladeaufgabe fort

Wenn Sie eine Batch-Ladeaufgabe mit dem Status „Fortschritt gestoppt“ auswählen, die immer noch fortgesetzt werden kann, werden Sie aufgefordert, die Aufgabe fortzusetzen. Es gibt auch ein Banner mit der Schaltfläche „Aufgabe fortsetzen“, wenn Sie sich die Details für diese Aufgaben ansehen. Aufgaben, die wieder aufgenommen werden können, haben ein Datum, an dem sie fortgesetzt werden können. Nach Ablauf dieses Datums können Aufgaben nicht wieder aufgenommen werden.

Verwenden des Visual Builders

Sie können den Visual Builder verwenden, um Quelldatenspalten, eine oder mehrere in einem S3-Bucket gespeicherte CSV Datei (en), Zielspalten in einer LiveAnalytics Timestream-for-Tabelle zuzuordnen.

> **Note**
>
> Ihre Rolle benötigt die `SelectObjectContent` Genehmigung für die Datei. Andernfalls müssen Sie Spalten manuell hinzufügen und löschen.

Modus „Quellspalten automatisch laden“

Timestream for LiveAnalytics kann die CSV Quelldatei automatisch nach Spaltennamen durchsuchen, wenn Sie nur einen Bucket angeben. Wenn keine Zuordnungen vorhanden sind, können Sie Quellspalten importieren auswählen.

1. Wählen Sie in den Eingabeeinstellungen für die Mapping-Konfiguration die Visual Builder-Option aus und legen Sie die Zeiteingabe für den Zeitstempel fest. `Milliseconds`ist die Standardeinstellung.

2. Klicken Sie auf die Schaltfläche Quellspalten laden, um die in der Quelldatendatei enthaltenen Spaltenüberschriften zu importieren. Die Tabelle wird mit den Namen der Quellspaltenüberschriften aus der Datenquellendatei gefüllt.
3. Wählen Sie für jede Quellspalte den Spaltennamen der Zieltabelle, den Timestream-Attributtyp und den Datentyp aus.

   Einzelheiten zu diesen Spalten und möglichen Werten finden Sie unterZuordnen von Feldern.
4. Verwenden Sie die drag-to-fill Funktion, um den Wert für mehrere Spalten gleichzeitig festzulegen.

Fügen Sie manuell Quellspalten hinzu

Wenn Sie einen Bucket oder ein CSV Präfix und kein einzelnes verwendenCSV, können Sie im Visual Editor mit den Schaltflächen Spaltenzuordnung hinzufügen und Spaltenzuordnungen löschen. Es gibt auch eine Schaltfläche zum Zurücksetzen von Zuordnungen.

Zuordnen von Feldern

- Name der Quellspalte — Der Name einer Spalte in der Quelldatei, die eine zu importierende Kennzahl darstellt. Timestream for LiveAnalytics kann diesen Wert automatisch auffüllen, wenn Sie Quellspalten importieren verwenden.
- Spaltenname der Zieltabelle — Optionale Eingabe, die den Spaltennamen für die Kennzahl in der Zieltabelle angibt.
- Timestream-Attributtyp — Der Attributtyp der Daten in der angegebenen Quellspalte, z. B. `DIMENSION`
  - TIMESTAMP— Gibt an, wann eine Kennzahl erfasst wurde.
  - MULTI— Es werden mehrere Kennzahlen dargestellt.
  - DIMENSION— Metadaten von Zeitreihen.
  - MEASURE_ NAME — Bei Datensätzen mit einer einzigen Kennzahl ist dies der Name der Kennzahl.
- Datentyp — Der Typ der Timestream-Spalte, z. B. `BOOLEAN`
  - BIGINT— Eine 64-Bit-Ganzzahl.
  - BOOLEAN— Die beiden Wahrheitswerte der Logik — wahr und falsch.
  - DOUBLE— 64-Bit-Zahl mit variabler Genauigkeit.

- TIMESTAMP— Eine Zeitinstanz, die Zeitangaben im Nanosekundenbereich verwendet und die Zeit seit der Unix-Epoche aufzeichnet. UTC

Verwenden von Batch-Load mit AWS CLI

Aufstellen

Führen Sie die folgenden Schritte aus, um mit der Verwendung von Batch-Load zu beginnen.

1. Installieren Sie das AWS CLI anhand der Anweisungen unterZugreifen auf Amazon Timestream für die LiveAnalytics Nutzung des AWS CLI.
2. Führen Sie den folgenden Befehl aus, um sicherzustellen, dass die CLI Timestream-Befehle aktualisiert wurden. Vergewissern Sie sich, dass create-batch-load-task es in der Liste steht.

   ```
   aws timestream-write help
   ```
3. Bereiten Sie eine Datenquelle anhand der Anweisungen unter vorEine Batch-Load-Datendatei wird vorbereitet.
4. Erstellen Sie eine Datenbank und eine Tabelle gemäß den Anweisungen unterZugreifen auf Amazon Timestream für die LiveAnalytics Nutzung des AWS CLI.
5. Erstellen Sie einen S3-Bucket für die Berichtsausgabe. Der Bucket muss sich in derselben Region befinden. Weitere Informationen zu Buckets finden Sie unter Amazon S3 S3-Buckets erstellen, konfigurieren und damit arbeiten.
6. Erstellen Sie eine Batch-Load-Aufgabe. Informationen zu den erforderlichen Schritten finden Sie unter Erstellen Sie eine Batch-Load-Aufgabe.
7. Bestätigen Sie den Status der Aufgabe. Informationen zu den erforderlichen Schritten finden Sie unter Beschreiben Sie die Batch-Load-Aufgabe.

Erstellen Sie eine Batch-Load-Aufgabe

Sie können mit dem `create-batch-load-task` Befehl eine Batch-Load-Task erstellen. Wenn Sie mit dem eine Batch-Load-Task erstellenCLI, können Sie einen JSON Parameter verwenden`cli-input-json`, mit dem Sie die Parameter zu einem einzigen JSON Fragment zusammenfassen können. Sie können diese Details auch mithilfe mehrerer anderer Parameter wie`data-model-configuration`, `data-source-configuration` `report-configuration``target-database-name`, und auseinandernehmen`target-table-name`.

Ein Beispiel finden Sie unter Beispiel für eine Batch-Load-Aufgabe erstellen

Beschreiben Sie die Batch-Load-Aufgabe

Sie können eine Beschreibung der Batch-Load-Aufgabe wie folgt abrufen.

```
aws timestream-write describe-batch-load-task --task-id <value>
```

Nachfolgend finden Sie eine Beispielantwort.

```
{
    "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
    }
}
```

Batch-Load-Aufgaben auflisten

Sie können Batch-Load-Aufgaben wie folgt auflisten.

```
aws timestream-write list-batch-load-tasks
```

Eine Ausgabe sieht wie folgt aus.

```
{
    "BatchLoadTasks": [
        {
            "TaskId": "<TaskId>",
            "TaskStatus": "FAILED",
            "DatabaseName": "BatchLoadExampleDatabase",
            "TableName": "BatchLoadExampleTable",
            "CreationTime": 1677167593.266,
            "LastUpdatedTime": 1677167602.38
        }
    ]
}
```

Setzen Sie die Batch-Ladeaufgabe fort

Sie können eine Batch-Ladeaufgabe wie folgt fortsetzen.

```
aws timestream-write resume-batch-load-task --task-id <value>
```

Eine Antwort kann auf Erfolg hinweisen oder Fehlerinformationen enthalten.

Beispiel für eine Batch-Load-Aufgabe erstellen

Example

1. Erstellen Sie einen Timestream für die LiveAnalytics Datenbank mit dem Namen `BatchLoad` und eine Tabelle mit dem Namen`BatchLoadTest`. Überprüfen Sie die Werte für `MemoryStoreRetentionPeriodInHours` und und passen Sie sie gegebenenfalls an. `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. Erstellen Sie mit der Konsole einen S3-Bucket und kopieren Sie die `sample.csv` Datei an diesen Speicherort. Sie können ein Beispiel CSV unter Beispiel herunterladenCSV.
3. Erstellen Sie mithilfe der Konsole einen S3-Bucket für Timestream LiveAnalytics , um einen Bericht zu schreiben, wenn die Batch-Ladeaufgabe fehlerhaft abgeschlossen wird.
4. Erstellen Sie eine Batch-Load-Aufgabe. Stellen Sie sicher, dass Sie es ersetzen *$INPUT_BUCKET* and *$REPORT_BUCKET* mit den Buckets, die Sie in den vorherigen Schritten erstellt haben.

```
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
```

Der vorherige Befehl gibt die folgende Ausgabe zurück.

```
{
    "TaskId": "TaskId "
}
```

5. Überprüfen Sie den Fortschritt der Aufgabe. Stellen Sie sicher, dass Sie ersetzen *$TASK_ID* mit der Aufgaben-ID, die im vorherigen Schritt zurückgegeben wurde.

```
aws timestream-write describe-batch-load-task --task-id $TASK_ID
```

Beispielausgabe

```
{
    "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
    }
}
```

Verwenden von Batch-Load mit AWS SDKs

Beispiele für das Erstellen, Beschreiben und Auflisten von Batch-Load-Aufgaben mit dem AWS SDKs, finden Sie unter Batch-Load-Aufgabe erstellenBeschreiben Sie die Batch-Load-Aufgabe,Batch-Load-Aufgaben auflisten, undBatch-Load-Aufgabe fortsetzen.

Verwenden von Batch-Load-Fehlerberichten

Batch-Load-Aufgaben haben einen der folgenden Statuswerte:

- `CREATED`(Erstellt) — Die Aufgabe wurde erstellt.
- `IN_PROGRESS`(In Bearbeitung) — Die Aufgabe ist in Bearbeitung.

- `FAILED`(Fehlgeschlagen) — Die Aufgabe wurde abgeschlossen. Es wurden jedoch ein oder mehrere Fehler festgestellt.
- `SUCCEEDED`(Abgeschlossen) — Die Aufgabe wurde ohne Fehler abgeschlossen.
- `PROGRESS_STOPPED`(Fortschritt gestoppt) — Die Aufgabe wurde gestoppt, aber nicht abgeschlossen. Sie können versuchen, die Aufgabe fortzusetzen.
- `PENDING_RESUME`(Ausstehende Wiederaufnahme) — Die Wiederaufnahme der Aufgabe steht noch aus.

Wenn Fehler auftreten, wird in dem dafür definierten S3-Bucket ein Fehlerprotokollbericht erstellt. Fehler werden als taskErrors oder fileErrors in separaten Arrays kategorisiert. Im Folgenden finden Sie ein Beispiel für einen Fehlerbericht.

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

Verwenden von geplanten Abfragen in Timestream für LiveAnalytics

Die Funktion für geplante Abfragen in Amazon Timestream for LiveAnalytics ist eine vollständig verwaltete, serverlose und skalierbare Lösung für die Berechnung und Speicherung von Aggregaten, Rollups und anderen Formen vorverarbeiteter Daten, die normalerweise für operative Dashboards, Geschäftsberichte, Ad-hoc-Analysen und andere Anwendungen verwendet werden. Geplante Abfragen machen Echtzeitanalysen leistungsfähiger und kostengünstiger, sodass Sie zusätzliche Erkenntnisse aus Ihren Daten gewinnen und weiterhin bessere Geschäftsentscheidungen treffen können.

Mit geplanten Abfragen definieren Sie die Echtzeit-Analyseabfragen, die Aggregationen, Rollups und andere Operationen mit den Daten berechnen. Amazon Timestream for führt diese Abfragen LiveAnalytics regelmäßig und automatisch aus und schreibt die Abfrageergebnisse zuverlässig in eine separate Tabelle. Die Daten werden in der Regel innerhalb weniger Minuten berechnet und in diesen Tabellen aktualisiert.

Sie können Ihre Dashboards und Berichte dann so ausrichten, dass sie die Tabellen abfragen, die aggregierte Daten enthalten, anstatt die wesentlich größeren Quelltabellen abzufragen. Dies führt zu Leistungs- und Kostensteigerungen, die Größenordnungen übersteigen können. Dies liegt daran, dass die Tabellen mit aggregierten Daten viel weniger Daten enthalten als die Quelltabellen, sodass sie schnellere Abfragen und eine kostengünstigere Datenspeicherung bieten.

Darüber hinaus bieten Tabellen mit geplanten Abfragen alle vorhandenen Funktionen eines Timestreams für LiveAnalytics Tabellen. Sie können die Tabellen beispielsweise mit SQL abfragen. Sie können die in den Tabellen gespeicherten Daten mit Grafana visualisieren. Sie können Daten auch mit Amazon Kinesis, AmazonMSK, AWS IoT Core und Telegraf in die Tabelle aufnehmen. Sie können Datenaufbewahrungsrichtlinien für diese Tabellen für die automatische Verwaltung des Datenlebenszyklus konfigurieren.

Da die Datenspeicherung der Tabellen, die aggregierte Daten enthalten, vollständig von der der Quelltabellen entkoppelt ist, können Sie sich auch dafür entscheiden, die Datenspeicherung der Quelltabellen zu reduzieren und die aggregierten Daten für einen viel längeren Zeitraum zu einem Bruchteil der Datenspeicherkosten aufzubewahren. Durch geplante Abfragen werden Echtzeitanalysen schneller, kostengünstiger und damit für viel mehr Kunden leichter zugänglich, sodass sie ihre Anwendungen überwachen und bessere datengestützte Geschäftsentscheidungen treffen können.

Themen

Vorteile von geplanten Abfragen

Im Folgenden sind die Vorteile von geplanten Abfragen aufgeführt:

- Einfache Bedienung — Geplante Abfragen sind serverlos und werden vollständig verwaltet.
- Leistung und Kosten — Da geplante Abfragen die Aggregate, Rollups oder andere Echtzeitanalysevorgänge für Ihre Daten vorab berechnen und die Ergebnisse in einer Tabelle speichern, enthalten Abfragen, die auf Tabellen zugreifen, die mit geplanten Abfragen gefüllt wurden, weniger Daten als die Quelltabellen. Daher sind Abfragen, die auf diesen Tabellen ausgeführt werden, schneller und kostengünstiger. Durch geplante Berechnungen aufgefüllte Tabellen enthalten weniger Daten als ihre Quelltabellen und tragen somit zur Senkung der Speicherkosten bei. Sie können diese Daten auch für einen längeren Zeitraum im Speicherspeicher aufbewahren, und das zu einem Bruchteil der Kosten, die für die Aufbewahrung der Quelldaten im Speicherspeicher anfallen.
- Interoperabilität — Mit geplanten Abfragen aufgefüllte Tabellen bieten die gesamte Funktionalität von Timestream für LiveAnalytics Tabellen und können mit allen Diensten und Tools verwendet werden, die mit Timestream for funktionieren. LiveAnalytics Einzelheiten finden Sie unter Arbeiten mit anderen Diensten.

Anwendungsfälle für geplante Abfragen

Sie können geplante Abfragen für Geschäftsberichte verwenden, die die Endbenutzeraktivitäten in Ihren Anwendungen zusammenfassen, sodass Sie Modelle für maschinelles Lernen für die Personalisierung trainieren können. Sie können geplante Abfragen auch für Alarme verwenden, mit denen Anomalien, Netzwerkeinbrüche oder betrügerische Aktivitäten erkannt werden, sodass Sie sofort Abhilfemaßnahmen ergreifen können.

Darüber hinaus können Sie geplante Abfragen für eine effektivere Datenverwaltung verwenden. Sie können dies tun, indem Sie den Quelltabellenzugriff ausschließlich auf die geplanten Abfragen gewähren und Ihren Entwicklern nur Zugriff auf die Tabellen gewähren, die mit geplanten Abfragen gefüllt werden. Dadurch werden die Auswirkungen unbeabsichtigter Abfragen mit langer Laufzeit minimiert.

Beispiel: Einsatz von Echtzeitanalysen, um betrügerische Zahlungen aufzudecken und bessere Geschäftsentscheidungen zu treffen

Stellen Sie sich ein Zahlungssystem vor, das Transaktionen verarbeitet, die von mehreren point-of-sale Terminals in den großen Metropolen der Vereinigten Staaten aus gesendet werden. Sie möchten Amazon Timestream verwenden, um die Transaktionsdaten LiveAnalytics zu speichern und zu analysieren, damit Sie betrügerische Transaktionen erkennen und Analyseabfragen in Echtzeit ausführen können. Diese Abfragen können Ihnen bei der Beantwortung geschäftlicher Fragen helfen, z. B. bei der Identifizierung der am stärksten frequentierten und am wenigsten genutzten point-of-sale Terminals pro Stunde, der geschäftigsten Stunde des Tages für jede Stadt und der Stadt mit den meisten Transaktionen pro Stunde.

Das System verarbeitet ~100.000 Transaktionen pro Minute. Jede in Amazon Timestream gespeicherte Transaktion umfasst LiveAnalytics 100 Byte. Sie haben 10 Abfragen konfiguriert, die jede Minute ausgeführt werden, um verschiedene Arten betrügerischer Zahlungen zu erkennen. Sie haben außerdem 25 Abfragen erstellt, die Ihre Daten aggregieren und nach verschiedenen Dimensionen aufteilen, um Ihre Geschäftsfragen zu beantworten. Jede dieser Abfragen verarbeitet die Daten der letzten Stunde.

Sie haben ein Dashboard erstellt, um die durch diese Abfragen generierten Daten anzuzeigen. Das Dashboard enthält 25 Widgets, es wird jede Stunde aktualisiert und es wird in der Regel von 10 Benutzern gleichzeitig aufgerufen. Schließlich ist Ihr Speicherspeicher mit einer Datenaufbewahrungszeit von 2 Stunden und der Magnetspeicher mit einer Datenaufbewahrungszeit von 6 Monaten konfiguriert.

In diesem Fall können Sie Echtzeitanalyseabfragen verwenden, bei denen die Daten bei jedem Zugriff auf das Dashboard und bei jeder Aktualisierung neu berechnet werden, oder abgeleitete Tabellen für das Dashboard verwenden. Die Abfragekosten für Dashboards, die auf Echtzeitanalyseabfragen basieren, belaufen sich auf 120,70 USD pro Monat. Im Gegensatz dazu belaufen sich die Kosten für Dashboard-Abfragen, die auf abgeleiteten Tabellen basieren, auf 12,27 USD pro Monat (Preise finden Sie unter Amazon Timestream). LiveAnalytics In diesem Fall reduziert die Verwendung von abgeleiteten Tabellen die Abfragekosten um das ~10-fache.

Konzepte für geplante Abfragen

Abfragezeichenfolge — Dies ist die Abfrage, deren Ergebnis Sie vorab berechnen und in einem anderen Timestream für LiveAnalytics eine Tabelle speichern. Sie können eine geplante Abfrage unter Verwendung der gesamten SQL Oberfläche von Timestream für definieren. Dies bietet Ihnen die Flexibilität LiveAnalytics, Abfragen mit gängigen Tabellenausdrücken, verschachtelten Abfragen, Fensterfunktionen oder beliebigen Aggregat- und Skalarfunktionen zu schreiben, die von Timestream für die Abfragesprache unterstützt werden. LiveAnalytics

Zeitplanausdruck — Ermöglicht es Ihnen, anzugeben, wann Ihre geplanten Abfrageinstanzen ausgeführt werden. Sie können die Ausdrücke mithilfe eines Cron-Ausdrucks (z. B. UTC jeden Tag um 8 Uhr ausführen) oder eines Rate-Ausdrucks (z. B. alle 10 Minuten ausführen) angeben.

Zielkonfiguration — Ermöglicht es Ihnen, anzugeben, wie Sie das Ergebnis einer geplanten Abfrage der Zieltabelle zuordnen, in der die Ergebnisse dieser geplanten Abfrage gespeichert werden.

Benachrichtigungskonfiguration — Timestream for führt LiveAnalytics automatisch Instanzen einer geplanten Abfrage auf der Grundlage Ihres Zeitplanausdrucks aus. Sie erhalten eine Benachrichtigung für jede solche Abfrage, die zu einem SNS Thema ausgeführt wird, das Sie bei der Erstellung einer geplanten Abfrage konfigurieren. Diese Benachrichtigung gibt an, ob die Instanz erfolgreich ausgeführt wurde oder ob Fehler aufgetreten sind. Darüber hinaus enthält sie Informationen wie die gemessenen Byte, die in die Zieltabelle geschriebenen Daten, die Uhrzeit des nächsten Aufrufs usw.

Im Folgenden finden Sie ein Beispiel für diese Art von Benachrichtigung.

```
{
    "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
        }
    }
}
```

In dieser Benachrichtigung stehen die Byte, `bytesMetered` die die Abfrage in der Quelltabelle gescannt hat, und dataWrites die Byte, die in die Zieltabelle geschrieben wurden.

> **Note**
>
> Wenn Sie diese Benachrichtigungen programmgesteuert verwenden, beachten Sie, dass der Benachrichtigungsnachricht in future neue Felder hinzugefügt werden könnten.

Speicherort des Fehlerberichts — Geplante Abfragen werden asynchron ausgeführt und Daten werden in der Zieltabelle gespeichert. Wenn eine Instanz auf Fehler stößt (z. B. ungültige Daten, die nicht gespeichert werden konnten), werden die Datensätze, bei denen Fehler aufgetreten sind, in einen Fehlerbericht an der Stelle geschrieben, die Sie bei der Erstellung einer geplanten Abfrage angegeben haben. Sie geben den S3-Bucket und das Präfix für den Standort an. Timestream for LiveAnalytics hängt den Namen der geplanten Abfrage und die Aufrufzeit an dieses Präfix an, damit Sie die Fehler identifizieren können, die mit einer bestimmten Instanz einer geplanten Abfrage verbunden sind.

Tagging — Sie können optional Tags angeben, die Sie einer geplanten Abfrage zuordnen können. Weitere Informationen finden Sie unter Tagging Timestream for Resources. LiveAnalytics

Beispiel

Im folgenden Beispiel berechnen Sie ein einfaches Aggregat mithilfe einer geplanten Abfrage:

```
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`- In diesem Beispiel werden Sie feststellen, dass die Abfrage einen speziellen benannten Parameter akzeptiert`@scheduled_runtime`. Dabei handelt es sich um einen speziellen Parameter (vom Typ Timestamp), den der Dienst beim Aufrufen einer bestimmten Instanz einer geplanten Abfrage festlegt, sodass Sie den Zeitraum, für den eine bestimmte Instanz einer geplanten Abfrage die Daten in der Quelltabelle analysiert, deterministisch steuern können. Sie können ihn `@scheduled_runtime` in Ihrer Abfrage an jedem Ort verwenden, an dem ein Timestamp-Typ erwartet wird.

Stellen Sie sich ein Beispiel vor, in dem Sie einen Zeitplanausdruck festlegen: cron (0/5 * *? *) wobei die geplante Abfrage in den Minuten 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55 jeder Stunde ausgeführt wird. Für die Instanz, die am 01.12.2021 00:05:00 ausgelöst wird, wird der @scheduled_runtime -Parameter mit diesem Wert initialisiert, sodass die Instanz zu diesem Zeitpunkt mit Daten im Bereich 2021-11-30 23:55:00 bis 2021-12-01 00:06:00. arbeitet.

Instanzen mit überlappenden Zeitbereichen — Wie Sie in diesem Beispiel sehen werden, können sich zwei aufeinanderfolgende Instanzen einer geplanten Abfrage in ihren Zeitbereichen überschneiden. Dies können Sie auf der Grundlage Ihrer Anforderungen, der von Ihnen angegebenen Zeitprädikate und des Zeitplanausdrucks steuern. In diesem Fall können diese Berechnungen aufgrund dieser Überschneidung die Aggregate auf der Grundlage aller Daten aktualisieren, deren Ankunft leicht verzögert war, in diesem Beispiel bis zu 10 Minuten. Der am 01.12.2021 00:00:00 ausgelöste Abfragelauf deckt den Zeitraum 2021-11-30 23:50:00 bis 2021-12-30 00:01:00 ab und der um 2021-12-01 00:05:00 ausgelöste Abfragelauf deckt den Bereich 2021-11-30 23:55:00 bis 2021-12-01 00:06:00.

Um die Richtigkeit sicherzustellen und sicherzustellen, dass die in der Zieltabelle gespeicherten Aggregate mit den aus der Quelltabelle berechneten Aggregaten übereinstimmen, LiveAnalytics stellt Timestream for sicher, dass die Berechnung am 01.12.2021 00:05:00 erst ausgeführt wird, nachdem die Berechnung am 01.12.2021 00:00:00 abgeschlossen ist. Die Ergebnisse der letztgenannten Berechnungen können jedes zuvor materialisierte Aggregat aktualisieren, wenn ein neuerer Wert generiert wird. Intern LiveAnalytics verwendet Timestream for Datensatzversionen, bei denen Datensätzen, die von späteren Instanzen einer geplanten Abfrage generiert wurden, eine höhere Versionsnummer zugewiesen wird. Daher können die durch den Aufruf am 01.12.2021 00:05:00 berechneten Aggregate die durch den Aufruf am 01.12.2021 00:00:00 berechneten Aggregate aktualisieren, vorausgesetzt, dass neuere Daten in der Quelltabelle verfügbar sind.

Automatische Trigger im Vergleich zu manuellen Triggern — Nachdem eine geplante Abfrage erstellt wurde, führt Timestream for die Instances automatisch auf der Grundlage des angegebenen Zeitplans aus. LiveAnalytics Solche automatisierten Auslöser werden vollständig vom Service verwaltet.

Es kann jedoch Szenarien geben, in denen Sie einige Instanzen einer geplanten Abfrage manuell initiieren möchten. Beispiele hierfür sind der Ausfall einer bestimmten Instanz bei einem Abfragelauf, wenn nach der Ausführung des automatisierten Zeitplans Daten oder Aktualisierungen in der Quelltabelle zu spät eingingen oder wenn Sie die Zieltabelle für Zeitbereiche aktualisieren möchten, die nicht von automatisierten Abfrageläufen abgedeckt werden (z. B. für Zeitbereiche vor der Erstellung einer geplanten Abfrage).

Sie können den verwenden ExecuteScheduledQuery API, um eine bestimmte Instanz einer geplanten Abfrage manuell zu initiieren, indem Sie den InvocationTime Parameter übergeben, bei dem es sich um einen Wert handelt, der für den @scheduled_runtime -Parameter verwendet wird. Im Folgenden sind einige wichtige Überlegungen bei der Verwendung von aufgeführt ExecuteScheduledQuery API:

- Wenn Sie mehrere dieser Aufrufe auslösen, müssen Sie sicherstellen, dass diese Aufrufe in sich überschneidenden Zeiträumen nicht zu Ergebnissen führen. Wenn Sie nicht sicherstellen können, dass sich die Zeitbereiche nicht überschneiden, stellen Sie sicher, dass diese Abfrageläufe nacheinander initiiert werden. Wenn Sie mehrere Abfrageläufe gleichzeitig initiieren, die sich in ihren Zeiträumen überschneiden, können in den Fehlerberichten für diese Abfrageläufe Triggerfehler auftreten, bei denen es zu Versionskonflikten kommen kann.
- Sie können die Aufrufe mit einem beliebigen Zeitstempelwert für @scheduled_runtime initiieren. Es liegt also in Ihrer Verantwortung, die Werte entsprechend festzulegen, sodass die entsprechenden Zeitbereiche in der Zieltabelle entsprechend den Bereichen aktualisiert werden, in denen Daten in der Quelltabelle aktualisiert wurden.

Planen Sie Ausdrücke für geplante Abfragen

Sie können geplante Abfragen nach einem automatisierten Zeitplan erstellen, indem Sie Amazon Timestream für LiveAnalytics geplante Abfragen verwenden, die Cron- oder Rate-Ausdrücke verwenden. Alle geplanten Abfragen verwenden die UTC Zeitzone, und die minimal mögliche Genauigkeit für Zeitpläne beträgt 1 Minute.

Die Zeitplanausdrücke können auf zwei Arten angegeben werden: Cron und Rate. Cron-Ausdrücke bieten eine detailliertere Steuerung des Zeitplans, während Tarifausdrücke einfacher auszudrücken sind, ihnen jedoch die feinkörnige Steuerung fehlt.

Mit einem Cron-Ausdruck können Sie beispielsweise eine geplante Abfrage definieren, die zu einer bestimmten Zeit an einem bestimmten Tag jeder Woche oder jedes Monats oder nur an Montag bis Freitag zu einer bestimmten Minute pro Stunde usw. ausgelöst wird. Im Gegensatz dazu initiieren Ratenausdrücke eine geplante Abfrage mit einer regulären Geschwindigkeit, z. B. einmal pro Minute, Stunde oder Tag, und zwar ab dem exakten Zeitpunkt, zu dem die geplante Abfrage erstellt wurde.

Cron-Ausdruck

- Syntax

```
cron(fields)
```

Cron-Ausdrücke verfügen über sechs Pflichtfelder, die durch Leerzeichen voneinander getrennt sind.

Feld	Werte	Platzhalter
Minuten	0-59	, - * /
Stunden	0-23	, - * /
D ay-of-month	1-31	, - * ? / L W
Monat	1-12 oder JAN - DEC	, - * /
D ay-of-week	1-7 oder SUN - SAT	, - * ? / L #
Jahr	1970-2199	, - * /

Platzhalterzeichen

- Der Platzhalter *, * (Komma) enthält zusätzliche Werte. Im Feld Monat MAR würde, JANFEB, Januar, Februar und März enthalten.
- Der Platzhalter*-* (Gedankenstrich) gibt Bereiche an. Im Feld "Tag" steht 1-15 für die Tage 1 bis 15 des angegebenen Monats.
- Der Platzhalter *** (Sternchen) umfasst alle Werte im Feld. Im Feld Stunden würde *** jede Stunde einschließen. Sie können *** nicht sowohl in den Day-of-week Feldern als Day-of-month auch verwenden. Wenn Sie es in einem verwenden, müssen Sie * verwenden? * in der anderen.

- Der Platzhalter*/* (Schrägstrich) gibt Inkremente an. Im Feld Minuten könnten Sie 1/10 eingeben, um jede zehnte Minute, beginnend mit der ersten Minute der Stunde, anzugeben (z. B. die 11., 21. und 31. Minute usw.).
- Das *? Der Platzhalter * (Fragezeichen) gibt den einen oder anderen an. In das Day-of-month Feld könntest du *7* eingeben und wenn es dir egal wäre, welcher Wochentag der 7. ist, könntest du * eingeben? * in das Feld Day-of-week.
- Der Platzhalter*L* in den Day-of-week Feldern Day-of-month oder gibt den letzten Tag des Monats oder der Woche an.
- Der Platzhalter W in dem Day-of-month Feld gibt einen Wochentag an. In dem Day-of-month Feld gibt 3W den Wochentag an, der dem dritten Tag des Monats am nächsten liegt.
- Der Platzhalter*#* in dem Day-of-week Feld gibt eine bestimmte Instanz des angegebenen Wochentags innerhalb eines Monats an. Beispiel: 3#2 steht für den zweiten Dienstag des Monats: Die 3 bezieht sich auf Dienstag, da dies der dritte Tag jeder Woche ist, und die 2 bezieht sich auf den zweiten Tag dieses Typs innerhalb des Monats.

> **Note**
>
> Wenn Sie ein '#'-Zeichen verwenden, können Sie nur einen Ausdruck in dem Feld definieren. day-of-week Beispielsweise ist "3#1,6#3" ungültig, da dies als zwei Ausdrücke interpretiert wird.

Einschränkungen

- Sie können die Day-of-week Felder Day-of-month und nicht im selben Cron-Ausdruck angeben. Wenn Sie in einem der Felder einen Wert (oder ein *) angeben, müssen Sie ein *? * (Fragezeichen) in dem anderen.
- Cron-Ausdrücke, die zu schnelleren Häufigkeiten als mit 1 Minute führen, werden nicht unterstützt.

Beispiele

Minuten	Stunden	Tag des Monats	Monat	Wochentag	Jahr	Bedeutung
0	10	*	*	?	*	Laufen Sie jeden Tag um 10:00 Uhr (UTC).
15	12	*	*	?	*	Laufen Sie jeden Tag um 12:15 Uhr (UTC).
0	18	?	*	MON-FRI	*	Läuft jeden Montag bis Freitag um 18:00 Uhr (UTC).
0	8	1	*	?	*	Läuft jeden ersten Tag des Monats um 8:00 Uhr (UTC).
0/15	*	*	*	?	*	Lauf alle 15 Minuten.
0/10	*	*	*	MON-FRI	*	Laufen Sie montags bis freitags alle 10 Minuten.

Minuten	Stunden	Tag des Monats	Monat	Wochentag	Jahr	Bedeutung
0/5	8-17	?	*	MON-FRI	*	Läuft montags bis freitags alle 5 Minuten zwischen 8:00 Uhr und 17:55 Uhr (UTC).

Rate-Ausdrücke

- Ein Rate-Ausdruck beginnt, wenn Sie eine Regel für ein geplantes Ereignis erstellen und mit dem definierten Zeitplan ausführen. Rate-Ausdrücke bestehen aus zwei Pflichtfeldern. Die Felder werden durch ein Leerzeichen voneinander getrennt.

  Syntax

  ```
  rate(value unit)
  ```

  - `value`: Eine positive Zahl.
  - `unit`: Die Zeiteinheit. Für Werte von 1 (z. B. Minute) und Werte über 1 (z. B. Minuten) sind unterschiedliche Einheiten erforderlich. Zulässige Werte: Minute | Minuten | Stunde | Stunden | Tag | Tage

Datenmodellzuordnungen für geplante Abfragen

Timestream for LiveAnalytics unterstützt die flexible Modellierung von Daten in seinen Tabellen. Dieselbe Flexibilität gilt auch für Ergebnisse von geplanten Abfragen, die in einem anderen Timestream for table materialisiert werden. LiveAnalytics Mit geplanten Abfragen können Sie jede Tabelle abfragen, unabhängig davon, ob sie Daten in Datensätzen mit mehreren Kennzahlen oder Datensätzen mit einer einzigen Kennzahl enthält, und die Abfrageergebnisse entweder mithilfe von Datensätzen mit mehreren Kennzahlen oder Datensätzen mit nur einer Kennzahl schreiben.

Sie verwenden das TargetConfiguration in der Spezifikation einer geplanten Abfrage, um die Abfrageergebnisse den entsprechenden Spalten in der abgeleiteten Zieltabelle zuzuordnen. In den folgenden Abschnitten werden die verschiedenen Möglichkeiten beschrieben, dies anzugeben TargetConfiguration , um unterschiedliche Datenmodelle in der abgeleiteten Tabelle zu erhalten. Insbesondere werden Sie Folgendes sehen:

- So schreiben Sie in Datensätze mit mehreren Kennzahlen, wenn das Abfrageergebnis keinen Kennzahlnamen hat und Sie den Namen der Zielkennzahl in der TargetConfiguration angeben.
- So verwenden Sie den Kennzahlnamen im Abfrageergebnis, um Datensätze mit mehreren Kennzahlen zu schreiben.
- Wie Sie ein Modell so definieren können, dass es mehrere Datensätze mit unterschiedlichen Attributen für mehrere Messgrößen schreibt.
- Wie Sie ein Modell definieren können, um in Datensätze mit einer einzigen Messgröße in der abgeleiteten Tabelle zu schreiben.
- Wie Sie Datensätze mit Einzelkennzahlen und/oder Datensätze mit mehreren Kennzahlen in einer geplanten Abfrage abfragen und die Ergebnisse entweder in einem Datensatz mit einer Kennzahl oder einem Datensatz mit mehreren Kennzahlen materialisieren können, sodass Sie die Flexibilität der Datenmodelle wählen können.

Beispiel: Name der Zielkennzahl für Datensätze mit mehreren Kennzahlen

In diesem Beispiel sehen Sie, dass die Abfrage Daten aus einer Tabelle mit Daten aus mehreren Kennzahlen liest und die Ergebnisse mithilfe von Datensätzen mit mehreren Kennzahlen in eine andere Tabelle schreibt. Das geplante Abfrageergebnis hat keine Spalte mit dem Namen einer natürlichen Kennzahl. Hier geben Sie den Namen der Kennzahl in der abgeleiteten Tabelle mithilfe der TargetMultiMeasureName Eigenschaft in an 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"
        }
    }
}
```

Das Mapping in diesem Beispiel erstellt einen Datensatz mit mehreren Kennzahlen mit dem Kennzahlnamen dashboard-metrics und den Attributnamen 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 Beachten Sie die optionale Verwendung von TargetMultiMeasureAttributeName , um die Abfrageausgabespalten in einen anderen Attributnamen umzubenennen, der für die Ergebnismaterialisierung verwendet wird.

Im Folgenden finden Sie das Schema für die Zieltabelle, sobald diese geplante Abfrage materialisiert wurde. Wie Sie dem Timestream für den LiveAnalytics Attributtyp im folgenden Ergebnis entnehmen können, werden die Ergebnisse in einem Datensatz mit mehreren Kennzahlen und einem Namen für eine einzelne Kennzahl materialisiert`dashboard-metrics`, wie im Kennzahlschema dargestellt.

Spalte	Typ	Timestream für den Attributtyp LiveAnalytics
Region	varchar	DIMENSION
measure_name	varchar	MEASURE_NAME
time	Zeitstempel	TIMESTAMP
CpuSystemP100	double	MULTI
avgMemCached	double	MULTI
min_CPU_Leerlauf_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
CpuUserP 100	double	MULTI

Im Folgenden sind die entsprechenden Messwerte aufgeführt, die mit einer SHOW MEASURES Abfrage ermittelt wurden.

measure_name	data_type	Dimensionen
Dashboard-Metriken	mehrfach	[{'dimension_name': 'Region', 'data_type': 'varchar'}]

Beispiel: Verwendung des Kennzahlnamens aus einer geplanten Abfrage in Datensätzen mit mehreren Kennzahlen

In diesem Beispiel sehen Sie, wie eine Abfrage aus einer Tabelle mit Datensätzen für einzelne Kennzahlen liest und die Ergebnisse in Datensätzen mit mehreren Kennzahlen materialisiert. In diesem Fall enthält das geplante Abfrageergebnis eine Spalte, deren Werte als Kennzahlnamen in der Zieltabelle verwendet werden können, in der die Ergebnisse der geplanten Abfrage materialisiert werden. Anschließend können Sie den Kennzahlnamen für den Datensatz mit mehreren Kennzahlen in der abgeleiteten Tabelle mithilfe der MeasureNameColumn Eigenschaft in angeben. 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"
        }
    }
}
```

Das Mapping in diesem Beispiel erstellt Datensätze mit mehreren Kennzahlen mit den Attributen avg_1h, p0_1h, sum_1h, p100_1h und verwendet die Werte der Spalte measure_name im Abfrageergebnis als Kennzahlname für die Datensätze mit mehreren Kennzahlen in der Zieltabelle. Beachten Sie außerdem, dass in den vorherigen Beispielen optional die Zuordnungen mit einer Teilmenge der Zuordnungen verwendet werden, um die Attribute umzubenennen.

TargetMultiMeasureAttributeName Beispielsweise wurde min_1h in p0_1h umbenannt und max_1h wurde in p100_1h umbenannt.

Das Folgende ist das Schema für die Zieltabelle, sobald diese geplante Abfrage materialisiert wurde. Wie Sie dem Timestream für den LiveAnalytics Attributtyp im folgenden Ergebnis entnehmen können, werden die Ergebnisse in einem Datensatz mit mehreren Kennzahlen materialisiert. Wenn Sie sich das Kennzahlschema ansehen, wurden neun verschiedene Kennzahlnamen aufgenommen, die den Werten in den Abfrageergebnissen entsprechen.

Spalte	Typ	Timestream für den Attributtyp LiveAnalytics
Region	varchar	DIMENSION
measure_name	varchar	MEASURE_NAME
time	Zeitstempel	TIMESTAMP
sum_1h	double	MULTI
p100_1h	double	MULTI
p0_1h	double	MULTI
avg_1h	double	MULTI

Im Folgenden sind die entsprechenden Messwerte aufgeführt, die mit einer Abfrage ermittelt wurden. SHOW MEASURES

measure_name	data_type	Dimensionen
cpu_idle	mehrfach	[{'dimension_name': 'Region', 'data_type': 'varchar'}]
CPU_System	mehrfach	[{'dimension_name': 'Region', 'data_type': 'varchar'}]
CPU_Benutzer	mehrfach	[{'dimension_name': 'Region', 'data_type': 'varchar'}]

measure_name	data_type	Dimensionen
disk_free	mehrfach	[{'dimension_name': 'Region', 'data_type': 'varchar'}]
disk_io_schreibt	mehrfach	[{'dimension_name': 'Region', 'data_type': 'varchar'}]
disk_used	mehrfach	[{'dimension_name': 'Region', 'data_type': 'varchar'}]
Speicher_zwischengespeichert	mehrfach	[{'dimension_name': 'Region', 'data_type': 'varchar'}]
speicherfrei	mehrfach	[{'dimension_name': 'Region', 'data_type': 'varchar'}]
speicherfrei	mehrfach	[{'dimension_name': 'Region', 'data_type': 'varchar'}]

Beispiel: Zuordnung der Ergebnisse zu verschiedenen Datensätzen mit mehreren Kennzahlen und unterschiedlichen Attributen

Das folgende Beispiel zeigt, wie Sie verschiedene Spalten in Ihrem Abfrageergebnis verschiedenen Datensätzen mit mehreren Kennzahlen mit unterschiedlichen Kennzahlnamen zuordnen können. Wenn Sie die folgende geplante Abfragedefinition sehen, enthält das Ergebnis der Abfrage die folgenden Spalten: region, hour, 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_cpu_user_1h, max_cpu_system_1h, min_cpu_system_1h. `region`ist der Dimension zugeordnet und `hour` ist der Zeitspalte zugeordnet.

Die MixedMeasureMappings Immobilie in. TargetConfiguration TimestreamConfigurationgibt an, wie die Messwerte Datensätzen mit mehreren Kennzahlen in der abgeleiteten Tabelle zugeordnet werden.

In diesem speziellen Beispiel werden avg_mem_cached_1h, min_mem_free_1h, max_mem_used_1h in einem Multi-Measure-Datensatz mit dem Messnamen mem_aggregates verwendet, total_disk_io_writes_1h, avg_disk_used_1h, avg_disk_free_1h werden in einem anderen Multi-

Measure-Datensatz mit dem Messnamen disk_aggregates verwendet, und schließlich werden max_cpu_user_1h, max_cpu_system_1h, min_cpu_system_1h in einem anderen Datensatz mit mehreren Messungen mit dem Messnamen cpu_aggregates verwendet.

In diesen Zuordnungen können Sie optional auch die Spalte mit den Abfrageergebnissen umbenennen, sodass sie in der Zieltabelle einen anderen Attributnamen hat. TargetMultiMeasureAttributeName Beispielsweise wird die Ergebnisspalte avg_mem_cached_1h umbenannt in, total_disk_io_writes_1h wird umbenannt in usw. avgMemCached totalIOWrites

Wenn Sie die Zuordnungen für Datensätze mit mehreren Kennzahlen definieren, überprüft Timestream for jede Zeile in den Abfrageergebnissen und ignoriert automatisch die Spaltenwerte, die Werte enthalten. LiveAnalytics NULL Wenn also bei Mappings mit mehreren Kennzahlnamen alle Spaltenwerte für diese Gruppe in der Zuordnung für eine bestimmte Zeile gelten, wird kein Wert NULL für diesen Kennzahlnamen für diese Zeile aufgenommen.

In der folgenden Zuordnung werden beispielsweise avg_mem_cached_1h, min_mem_free_1h und max_mem_used_1h dem Messnamen mem_aggregates zugeordnet. Wenn für eine bestimmte Zeile des Abfrageergebnisses alle diese Spaltenwerte lauten, nimmt Timestream for die Kennzahl mem_aggregates für diese Zeile nicht auf. NULL LiveAnalytics Wenn alle neun Spalten für eine bestimmte Zeile vorhanden sindNULL, wird in Ihrem Fehlerbericht ein Benutzerfehler gemeldet.

```
{
    "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"
    }
}
```

```
}
```

Im Folgenden finden Sie das Schema für die Zieltabelle, sobald diese geplante Abfrage materialisiert wurde.

Spalte	Typ	Timestream für den Attributtyp LiveAnalytics
Region	varchar	DIMENSION
measure_name	varchar	MEASURE_NAME
time	Zeitstempel	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

Im Folgenden sind die entsprechenden Messwerte aufgeführt, die mit einer Abfrage ermittelt wurden. SHOW MEASURES

measure_name	data_type	Dimensionen
cpu_aggregates	mehrfach	[{'dimension_name': 'Region', 'data_type': 'varchar'}]

measure_name	data_type	Dimensionen
Festplattenaggregate	mehrfach	[{'dimension_name': 'Region', 'data_type': 'varchar'}]
mem_aggregate	mehrfach	[{'dimension_name': 'Region', 'data_type': 'varchar'}]

Beispiel: Zuordnung von Ergebnissen zu Einzelkennzahldatensätzen mit Kennzahlnamen aus Abfrageergebnissen

Im Folgenden finden Sie ein Beispiel für eine geplante Abfrage, deren Ergebnisse in Einzelkennzahldatensätzen zusammengefasst werden. In diesem Beispiel enthält das Abfrageergebnis die Spalte measure_name, deren Werte als Kennzahlnamen in der Zieltabelle verwendet werden. Sie verwenden das MixedMeasureMappings Attribut in der. TargetConfiguration TimestreamConfiguration um die Zuordnung der Abfrageergebnisspalte zur skalaren Kennzahl in der Zieltabelle anzugeben.

In der folgenden Beispieldefinition wird erwartet, dass das Abfrageergebnis neun verschiedene measure_name-Werte enthält. Sie listen all diese Kennzahlnamen im Mapping auf und geben an, welche Spalte für den Einzelmesswert für diesen Kennzahlnamen verwendet werden soll. Wenn in diesem Mapping beispielsweise der Kennzahlname memory_cached für eine bestimmte Ergebniszeile angezeigt wird, wird der Wert in der Spalte avg_1h als Wert für die Kennzahl verwendet, wenn die Daten in die Zieltabelle geschrieben werden. Sie können optional einen neuen Kennzahlnamen TargetMeasureName für diesen Wert angeben.

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

Im Folgenden finden Sie das Schema für die Zieltabelle, sobald diese geplante Abfrage materialisiert wurde. Wie Sie dem Schema entnehmen können, verwendet die Tabelle Datensätze mit einer einzigen Kennzahl. Wenn Sie das Kennzahlschema für die Tabelle auflisten, sehen Sie die neun

Kennzahlen, in die geschrieben wurde, basierend auf der in der Spezifikation angegebenen Zuordnung.

Spalte	Typ	Timestream für den LiveAnalytics Attributtyp
Region	varchar	DIMENSION
measure_name	varchar	MEASURE_NAME
time	Zeitstempel	TIMESTAMP
Messwert::doppelt	double	MEASURE_VALUE

Im Folgenden sind die entsprechenden Messwerte aufgeführt, die mit einer SHOW MEASURES Abfrage ermittelt wurden.

measure_name	data_type	Dimensionen
AvgMemCached	double	[{'dimension_name': 'region', 'data_type': 'varchar'}]
MinMemFree	double	[{'dimension_name': 'Region', 'data_type': 'varchar'}]
CPU_Leerlauf	double	[{'dimension_name': 'Region', 'data_type': 'varchar'}]
CPU_System	double	[{'dimension_name': 'Region', 'data_type': 'varchar'}]
CPU_Benutzer	double	[{'dimension_name': 'Region', 'data_type': 'varchar'}]
disk_free	double	[{'dimension_name': 'Region', 'data_type': 'varchar'}]
disk_used	double	[{'dimension_name': 'Region', 'data_type': 'varchar'}]

measure_name	data_type	Dimensionen
maxMemUsed	double	[{'dimension_name': 'Region', 'data_type': 'varchar'}]
total-disk-io-writes	double	[{'dimension_name': 'Region', 'data_type': 'varchar'}]

Beispiel: Zuordnung von Ergebnissen zu Einzelkennzahldatensätzen mit Abfrageergebnisspalten als Kennzahlnamen

In diesem Beispiel haben Sie eine Abfrage, deren Ergebnisse keine Spalte mit dem Kennzahlnamen haben. Stattdessen möchten Sie den Namen der Spalte mit dem Abfrageergebnis als Kennzahlnamen verwenden, wenn Sie die Ausgabe Datensätzen mit einer einzelnen Kennzahl zuordnen. Zuvor gab es ein Beispiel, bei dem ein ähnliches Ergebnis in einen Datensatz mit mehreren Kennzahlen geschrieben wurde. In diesem Beispiel erfahren Sie, wie Sie es Datensätzen mit einer Messgröße zuordnen können, sofern dies zu Ihrem Anwendungsszenario passt.

Auch hier geben Sie diese Zuordnung mit der MixedMeasureMappings Eigenschaft in an TargetConfiguration. TimestreamConfiguration. Im folgenden Beispiel sehen Sie, dass das Abfrageergebnis neun Spalten hat. Sie verwenden die Ergebnisspalten als Kennzahlnamen und die Werte als Einzelkennzahlwerte.

Beispielsweise wird für eine bestimmte Zeile im Abfrageergebnis der Spaltenname avg_mem_cached_1h als Spaltenname und Wert verwendet, der der Spalte zugeordnet ist, und avg_mem_cached_1h wird als Messwert für den Einzelmessdatensatz verwendet. Sie können dies auch verwenden, um TargetMeasureName einen anderen Kennzahlnamen in der Zieltabelle zu verwenden. Beispielsweise gibt das Mapping für Werte in der Spalte sum_1h an, total_disk_io_writes_1h als Kennzahlname in der Zieltabelle zu verwenden. Wenn der Wert einer Spalte gleich ist, wird die entsprechende Kennzahl ignoriert. NULL

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

Im Folgenden finden Sie das Schema für die Zieltabelle, sobald diese geplante Abfrage materialisiert wurde. Wie Sie sehen können, speichert die Zieltabelle Datensätze mit Einzelmesswerten vom Typ Double. In ähnlicher Weise zeigt das Kennzahlschema für die Tabelle die neun Kennzahlnamen. Beachten Sie auch, dass der Kennzahlname total_disk_io_writes_1h vorhanden ist, seit das Mapping sum_1h in total_disk_io_writes_1h umbenannt hat.

Spalte	Typ	LiveAnalytics Timestream für den Attributtyp
Region	varchar	DIMENSION
measure_name	varchar	MEASURE_NAME
time	Zeitstempel	TIMESTAMP
Messwert::doppelt	double	MEASURE_VALUE

Im Folgenden sind die entsprechenden Messwerte aufgeführt, die mit einer SHOW MEASURES Abfrage ermittelt wurden.

measure_name	data_type	Dimensionen
avg_disk_free_1h	double	[{'dimension_name': 'Region', 'data_type': 'varchar'}]
avg_disk_used_1h	double	[{'dimension_name': 'Region', 'data_type': 'varchar'}]
avg_mem_cached_1h	double	[{'dimension_name': 'Region', 'data_type': 'varchar'}]
max_cpu_system_1h	double	[{'dimension_name': 'Region', 'data_type': 'varchar'}]
max_cpu_user_1h	double	[{'dimension_name': 'Region', 'data_type': 'varchar'}]
max_mem_used_1h	double	[{'dimension_name': 'Region', 'data_type': 'varchar'}]
min_cpu_idle_1h	double	[{'dimension_name': 'Region', 'data_type': 'varchar'}]

measure_name	data_type	Dimensionen
min_mem_free_1h	double	[{'dimension_name': 'Region', 'data_type': 'varchar'}]
total-disk-io-writes	double	[{'dimension_name': 'Region', 'data_type': 'varchar'}]

Benachrichtigungen über geplante Abfragen

In diesem Abschnitt werden die Nachrichten beschrieben, die Timestream LiveAnalytics beim Erstellen, Löschen, Ausführen oder Aktualisieren des Status einer geplanten Abfrage sendet.

Name der Benachrichtigungsnachricht	Struktur	Beschreibung
CreatingNotificationMessage	`CreatingNotificationMessage { String arn; NotificationType type; }`	Diese Benachrichtigungsnachricht wird vor dem Senden der Antwort für gesendet`CreateScheduledQuery` . Die geplante Abfrage wird nach dem Senden dieser Benachrichtigung aktiviert. arn — Der ARN Name der geplanten Abfrage, die gerade erstellt wird. Typ - SCHEDULED _ QUERY _ CREATING
UpdateNotificationMessage	`UpdateNotificationMessage { String arn; NotificationType type; QueryState state;`	Diese Benachrichtigung wird gesendet, wenn eine geplante Abfrage aktualisiert wird. Timestream for LiveAnalytics kann die geplante Abfrage

Name der Benachrichtigungsnachricht	Struktur	Beschreibung
	}	automatisch deaktivieren, falls ein nicht behebbarer Fehler auftritt, wie z. B.: • AssumeRole Ausfall • Alle 4xx-Fehler, die bei der Kommunikation mit aufgetreten sindKMS, wenn ein vom Kunden verwalteter KMS Schlüssel angegeben ist. • Alle 4xx-Fehler, die bei der Ausführung der geplanten Abfrage aufgetreten sind. • Alle 4xx-Fehler, die bei der Erfassung von Abfrageergebnissen aufgetreten sind arn — Der Name ARN der geplanten Abfrage, die aktualisiert wird. Typ - SCHEDULED _ QUERY _ UPDATE Bundesstaat - ENABLED oder DISABLED

Name der Benachrichtigungsnachricht	Struktur	Beschreibung
DeleteNotificationMessage	`DeletionNotificationMessage { String arn; NotificationType type; }`	Diese Benachrichtigung wird gesendet, wenn eine geplante Abfrage gelöscht wurde. arn — Der ARN Name der geplanten Abfrage, die gerade erstellt wird. Typ - SCHEDULED _ QUERY _ DELETED

Name der Benachrichtigungsnachricht	Struktur	Beschreibung

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 {`
		Diese Benachrichtigung wird gesendet, nachdem die geplante Abfrage ausgeführt und die Ergebnisse erfolgreich aufgenommen wurden.
		
		ARN— Die ARN der geplanten Abfrage, die gelöscht wird.
		
		NotificationType- AUTO _ TRIGGER _ SUCCESS oder MANUAL _ TRIGGER _SUCCESS.
		
		nextInvocationEpochZweitens: Beim nächsten Mal führt Timestream for LiveAnalytics die geplante Abfrage aus.
		
		runSummary— Informationen zum geplanten Abfragelauf.

Name der Benachrichtigungsnachricht	Struktur	Beschreibung

	S3ReportLocation s3ReportLocation;
	}
	
	S3ReportLocation {
	String bucketName;
	String objectKey;
	}	

Name der Benachrichtigungsnachricht	Struktur	Beschreibung

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;`
		Diese Benachrichtigung wird gesendet, wenn während eines geplanten Abfragelaufs oder beim Einlesen der Abfrageergebnisse ein Fehler auftritt.
		
		arn — Der Name ARN der geplanten Abfrage, die gerade ausgeführt wird.
		
		geben Sie - AUTO _ TRIGGER _ FAILURE oder MANUAL _ TRIGGER _ einFAILURE.
		
		runSummary- Informationen zum geplanten Abfragelauf.

Name der Benachrichtigungsnachricht	Struktur	Beschreibung

	`}`
	
	`S3ReportLocation {`
	`  String bucketName;`
	`  String objectKey;`
	`}`	

Fehlerberichte für geplante Abfragen

In diesem Abschnitt werden der Speicherort, das Format und die Gründe für Fehlerberichte beschrieben, die von Timestream für den Fall generiert werden, dass LiveAnalytics bei der Ausführung von geplanten Abfragen Fehler auftreten.

Themen

- Gründe für Fehlerberichte bei geplanten Abfragen
- Bei der geplanten Abfrage wird der Speicherort von Fehlern gemeldet
- Format der Fehlerberichte für geplante Abfragen
- Fehlertypen bei geplanten Abfragen
- Beispiel für Fehlerberichte bei geplanten Abfragen

Gründe für Fehlerberichte bei geplanten Abfragen

Fehlerberichte werden für behebbare Fehler generiert. Für nicht behebbare Fehler werden keine Fehlerberichte generiert. Timestream for LiveAnalytics kann die geplanten Abfragen automatisch deaktivieren, wenn nicht behebbare Fehler auftreten. Dazu zählen:

- `AssumeRole`Ausfall
- Alle 4xx-Fehler, die bei der Kommunikation mit aufgetreten sindKMS, wenn ein vom Kunden verwalteter KMS Schlüssel angegeben wurde
- Alle 4xx-Fehler, die bei der Ausführung einer geplanten Abfrage aufgetreten sind
- Alle 4xx-Fehler, die bei der Erfassung von Abfrageergebnissen aufgetreten sind

Bei nicht behebbaren Fehlern LiveAnalytics sendet Timestream for eine Fehlerbenachrichtigung mit einer nicht behebbaren Fehlermeldung. Außerdem wird eine Aktualisierungsbenachrichtigung gesendet, die darauf hinweist, dass die geplante Abfrage deaktiviert ist.

Bei der geplanten Abfrage wird der Speicherort von Fehlern gemeldet

Ein Speicherort für Berichte über geplante Abfragefehler hat die folgende Benennungskonvention:

```
s3://customer-bucket/customer-prefix/
```

Im Folgenden finden Sie ein Beispiel für eine geplante AbfrageARN:

```
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 zeigt geplante Abfragen an, die von Timestream automatisch geplant wurden für LiveAnalytics und *Manual* weist auf geplante Abfragen hin, die manuell von einem Benutzer über eine `ExecuteScheduledQuery` API Aktion in Amazon Timestream for LiveAnalytics Query ausgelöst wurden. Weitere Informationen zu finden Sie `ExecuteScheduledQuery` unter ExecuteScheduledQuery.

Format der Fehlerberichte für geplante Abfragen

Die Fehlerberichte haben das folgende JSON Format:

```
{
    "reportId": <String>,            // A unique string ID for all error reports
 belonging to a particular scheduled query run
    "errors": [ <Error>, ... ],      // One or more errors
}
```

Fehlertypen bei geplanten Abfragen

Bei dem `Error` Objekt kann es sich um einen von drei Typen handeln:

- Zeichnet Fehler bei der Aufnahme auf

```
{
```

```
    "reason": <String>,              // The error message String
    "records": [ <Record>, ... ],    // One or more rejected records )
}
```

- Fehler bei der Analyse und Überprüfung von Zeilen

```
{
    "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.
}
```

- Allgemeine Fehler

```
{
    "reason": <String>,        // The error message
}
```

Beispiel für Fehlerberichte bei geplanten Abfragen

Im Folgenden finden Sie ein Beispiel für einen Fehlerbericht, der aufgrund von Eingabefehlern erstellt wurde.

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

Geplante Abfragemuster und Beispiele

In diesem Abschnitt werden die Verwendungsmuster für geplante Abfragen sowie end-to-end Beispiele beschrieben.

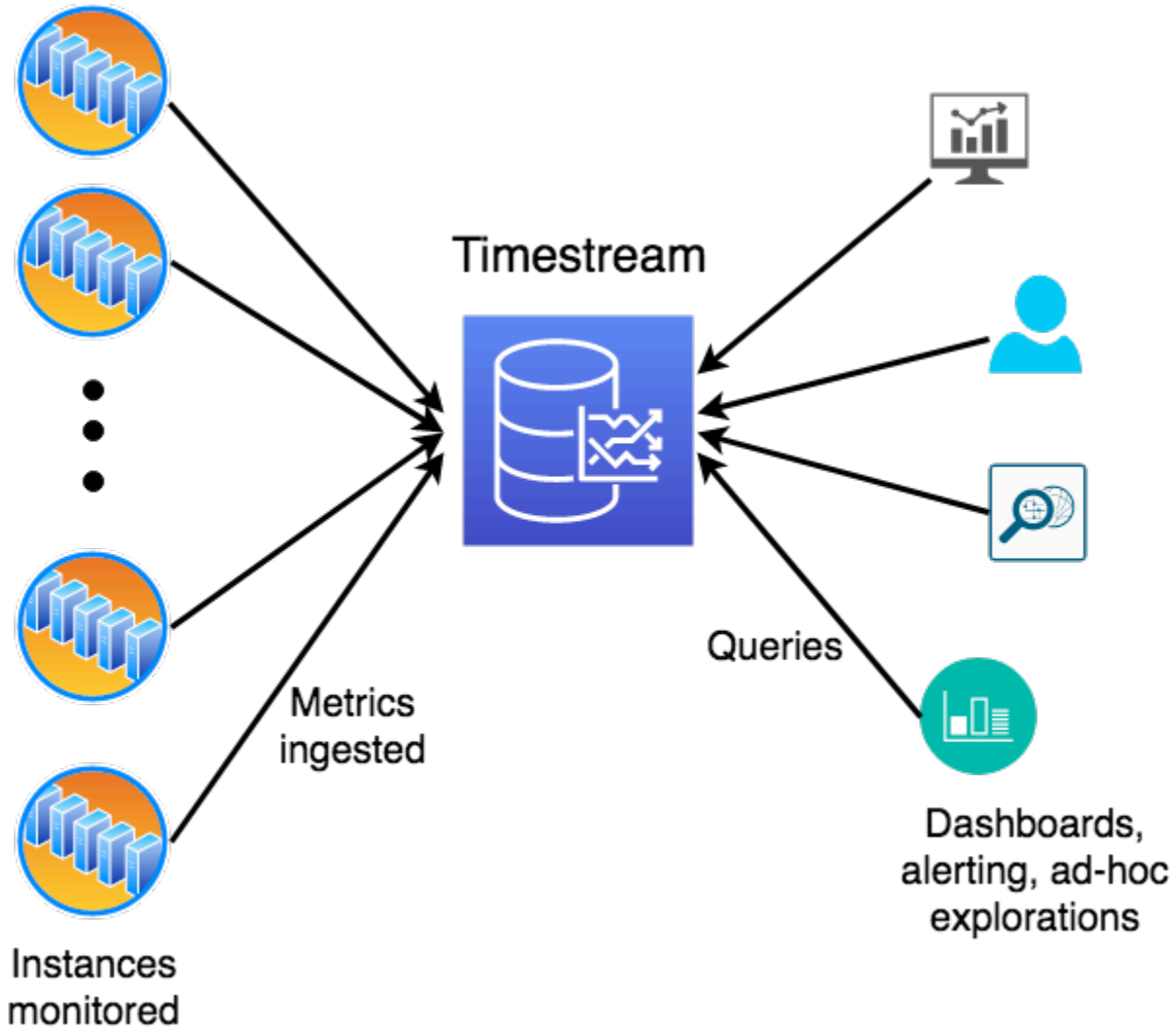
Themen

- Beispielschema für geplante Abfragen
- Geplante Abfragemuster

- Beispiele für geplante Abfragen

Beispielschema für geplante Abfragen

In diesem Beispiel verwenden wir eine Beispielanwendung, die ein DevOps Szenario nachahmt und die Messwerte einer großen Serverflotte überwacht. Benutzer möchten bei anomaler Ressourcennutzung warnen, Dashboards zum Verhalten und zur Auslastung der gesamten Flotte erstellen und ausgefeilte Analysen aktueller und historischer Daten durchführen, um Korrelationen zu finden. Das folgende Diagramm veranschaulicht die Konfiguration, bei der eine Reihe von überwachten Instances Metriken an Timestream sendet. LiveAnalytics Eine andere Gruppe gleichzeitiger Benutzer gibt Abfragen für Benachrichtigungen, Dashboards oder Ad-hoc-Analysen aus, bei denen Abfragen und Erfassung parallel ausgeführt werden.



Die überwachte Anwendung wird als hochgradig skalierter Dienst modelliert, der in mehreren Regionen auf der ganzen Welt eingesetzt wird. Jede Region ist weiter in eine Reihe von Skalierungseinheiten unterteilt, die als Zellen bezeichnet werden und in Bezug auf die Infrastruktur innerhalb der Region ein gewisses Maß an Isolation aufweisen. Jede Zelle ist weiter in Silos unterteilt, die ein gewisses Maß an Softwareisolierung darstellen. Jedes Silo hat fünf Microservices, die eine isolierte Instanz des Dienstes bilden. Jeder Microservice verfügt über mehrere Server mit unterschiedlichen Instanztypen und Betriebssystemversionen, die in drei Verfügbarkeitszonen bereitgestellt werden. Diese Attribute, die die Server identifizieren, die die Metriken ausgeben, werden in Timestream for als Dimensionen modelliert. LiveAnalytics In dieser Architektur haben wir eine Hierarchie von Dimensionen (wie Region, Zelle, Silo und Microservice_Name) und anderen Dimensionen, die sich über die gesamte Hierarchie erstrecken (wie instance_type und availability_zone).

Die Anwendung gibt eine Vielzahl von Metriken (wie cpu_user und memory_free) und Ereignissen (wie task_completed und gc_reclaimed) aus. Jeder Metrik oder jedem Ereignis sind acht Dimensionen (wie Region oder Zelle) zugeordnet, die den Server, der sie ausgibt, eindeutig identifizieren. Die Daten werden mit den 20 Metriken geschrieben, die zusammen in einem Datensatz mit mehreren Kennzahlen mit Kennzahlnamen gespeichert sind, und alle 5 Ereignisse werden zusammen in einem weiteren Datensatz mit mehreren Kennzahlen mit Kennzahlnamenereignissen gespeichert. Das Datenmodell, das Schema und die Datengenerierung finden Sie im Open-Source-Datengenerator. Neben dem Schema und den Datenverteilungen bietet der Datengenerator ein Beispiel für die Verwendung mehrerer Writer, um Daten parallel aufzunehmen, wobei die Aufnahmeskalierung von Timestream verwendet wird, LiveAnalytics um Millionen von Messungen pro Sekunde aufzunehmen. Im Folgenden zeigen wir das Schema (Tabelle und Kennzahlschema) und einige Beispieldaten aus dem Datensatz.

Themen

- Aufzeichnungen mit mehreren Messwerten
- Aufzeichnungen über einzelne Messwerte

Aufzeichnungen mit mehreren Messwerten

Tabellenschema

Im Folgenden finden Sie das Tabellenschema, sobald die Daten mithilfe von Datensätzen mit mehreren Kennzahlen aufgenommen wurden. Es ist die Ausgabe einer Abfrage. DESCRIBE Unter

der Annahme, dass die Daten in die Datenbank raw_data und die Tabelle devops aufgenommen wurden, finden Sie unten die Abfrage.

```
DESCRIBE "raw_data"."devops"
```

Spalte	Typ	LiveAnalytics Timestream für den Attributtyp
availability_zone	varchar	DIMENSION
microservice_name	varchar	DIMENSION
Instanzname	varchar	DIMENSION
Prozessname	varchar	DIMENSION
os_version	varchar	DIMENSION
jdk-Version	varchar	DIMENSION
Zelle	varchar	DIMENSION
Region	varchar	DIMENSION
Silo	varchar	DIMENSION
instance_type	varchar	DIMENSION
measure_name	varchar	MEASURE_NAME
time	Zeitstempel	TIMESTAMP
speicherfrei	double	MULTI
cpu_stehlen	double	MULTI
cpu_iowait	double	MULTI
CPU_Benutzer	double	MULTI
speicher_zwischengespeichert	double	MULTI

Spalte	Typ	LiveAnalytics Timestream für den Attributtyp
disk_io_reads	double	MULTI
CPU_Hi	double	MULTI
Latenz pro Lesevorgang	double	MULTI
Netzwerk-Bytes_Out	double	MULTI
CPU_Leerlauf	double	MULTI
disk_free	double	MULTI
verwendeter Speicher	double	MULTI
CPU_System	double	MULTI
Verwendete Dateideskriptoren	double	MULTI
disk_used	double	MULTI
CPU_nett	double	MULTI
disk_io_schreibt	double	MULTI
cpu_si	double	MULTI
Latenz pro Schreibvorgang	double	MULTI
Netzwerk-Bytes-Eingang	double	MULTI
task_end_state	varchar	MULTI
gc_pause	double	MULTI
aufgabe_abgeschlossen	bigint	MULTI
gc_reklamiert	double	MULTI

Maßnahmenschema

Im Folgenden finden Sie das von der SHOW MEASURES Abfrage zurückgegebene Kennzahlschema.

```
SHOW MEASURES FROM "raw_data"."devops"
```

measure_name	data_type	Dimensionen
Veranstaltungen	mehrfach	[{"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 “ :"varchar“, "dimensio n_name“ :"Prozessname "}, {" data_type“ :"varchar“, "dimension_name“ :"jdk_ver sion "}, {" data_type“ :"varchar “, "dimension_name“ :"cell "}, {" data_type“ :"varchar“, "dimension_name“ :"region "}, {" data_type“ :"dimensi on_type“ :"dimensi on_type“ :"region "} „varchar“, "dimension_name“ :"silo "}]
Kennzahlen	mehrfach	[{"data_type“ :"varchar“, "dimension_name“ :"availab ility_zone "}, {" data_type “ :"varchar“, "dimensio n_name“ :"microservice_nam e "}, {" data_type“ :"varchar“, "dimension_name“ :"instanc

measure_name	data_type	Dimensionen
		e_name "}, {" data_type " :"varchar", "dimensio n_name" :"os_version "}, {" data_type" :"varchar", "dimension_name" :"cell "}, {" data_type" :"varchar", "dimension_name" :"region "}, {" data_type" :"varchar ", "dimension_name" :"silo "}, {" data_type" :"varchar", "dimension_name" :"silo "}, {" data_type" :"varchar" archar", "dimension_name" :"Instanztyp "}]

Beispieldaten

R	Z	S	a	N	Ir	in	o	P	jo	m	Z	C	C	C	C	c	c	C	c	v	S	d	L	d	L	d	d	N	N	D	s	ta	g	a	g	a
u	U	U	U	a	i-	m	A			K	1	6	0	3	0	0	0	0	0	5	8	5	9	3	2	6	2	8	4	3	5					
e	C	C	C		-	e				n	1	-		-						5	t		n		rg			g	B	ra	w	re	st		bs	
	Z	Z	1		2					a	1																									
		Si			Z																															
					1																															
					S																															
					0																															
					a																															
					.c																															
					za																															
					at																															
					u																															

R	Z	S	a	N	Ir	in	o	P	jo	m	Z	C	C	C	C	c	c	C	c	v	S	d	L	d	L	d	d	N	N	D	s	t	g	a	g	g
					-																																
u	U	U	U	a	i-	m	A			K	1	5	0	3	0	0	0	0	0	9	3	5	3	2	9	5	3	7	5	6	2				
	C	C	C		-	e				n	1																										
	Z	Z	1		2						1																										
		S			Z						1																										
		il			1						1																										
					S																																
					0																																
					a																																
					.c																																
					za																																
					au																																
					-																																
					e																																
u	U	U	U	a	i-	m	A			K	1	4	0	4	0	0	0	0	0	9	4	5	3	8	3	7	6	8	5	4	8				
	C	C	C		-	e				n	1																										
	Z	Z	1		2						1																										
		S			Z						1																										
		il			1						1																										
					S																																
					0																																
					a																																
					.c																																
					za																																
					au																																
					-																																
					e																																

R	Z	S	a	N	Ir	in	o	P	jo	m	Z	C	C	C	C	c	c	C	c	v	S	d	L	d	L	d	d	N	N	D	s	ta	g	a	g	ge
			ity	d	m	ty	n	m	Vo	a		z	m	a	e	t				e	zv	e	p	cl	p	d	d	E	ra	ri	re	st			bs	
				M	a	i														S	e		Ln		Sr			g		w						
				st																	t					g					v						
us	UC	UC	UC	at	i-	me	A			Kn	11	3	0	5	0	0	0	0	0	4	7	2	4	4	3	8	6	2	1	2	1				
	Z	Zs	1:		2-						11																									
		il			Z1																															
					S0																															
					a.																															
					cz																															
					at																															
					au																															
					-e																															
us	UC	UC	UC	at	i-	me	A			Kn	11	5	0	3	0	0	0	0	0	5	3	8	5	7	1	8	6	7	6	5	3				
	Z	Zs	1:		2-						11																									
		il			Z1																															
					S0																															
					a.																															
					cz																															
					at																															
					au																															
					-e																															

R	Z	S	a	N	Ir	in	o	P	jo	m	Z	C	C	C	C	c	c	C	c	ve	S	d	L	d	L	d	d	N	N	D	sp	ta	g	a	ga	ge	ka
			ity	d	m	ty	n	m	Vo	a		zo	m	a	e	t					ze	t	e	n	cl	rg			g	Ra	w	re	st		bs		
				M																	S								E		v						
				st																										a								
us	UC	UC	UC	a	i-			He	JI	Vi	11																				7	St	8	3	64	3,8	
	Z	ZS	1		2-					tu	1																					_I					
		il			Z1-						1																					W					
					S0																											R					
					a.																																
					cz																																
					at																																
					au																																
					-e																																
us	UC	UC	UC	a	i-			He	JI	Vi	11																				7	SI	2	4	73	45	W
	Z	ZS	1		2-					tu	1																					T					JL
		il			Z1-						1																										
					S0																																
					a.																																
					cz																																
					at																																
					au																																
					-e																																

R	Z	S	a	N	Ir	in	o	P	jc	m	Z	C	C	C	C	c	c	C	c	v	S	d	L	d	L	d	d	N	N	D	s	ta	g	a	g	g
				M	m	ty	n	m	Vo	a		zo	m	a	e	t				e	zv	e	Ln	cl	rg			Eg	ra	wv	re	st		bs	ie	
us	UC	UC	UC	a	i-			He	JI	Vi	11																				6	SI	6	2	72	2_3
	Z	Zs	1:		2-					tu	1:																					T				W
		il			Z						1:																									JL
					1-																															
					S																															
					0																															
					a																															
					.c																															
					za																															
					au																															
					-																															
					e																															
us	UC	UC	UC	a	i-			He	JI	Vi	11																				2	SI	5	1	99	9_W
	Z	Zs	1:		2-					tu	1:																					T				JL
		il			Z						1:																									
					1-																															
					S																															
					0																															
					a																															
					.c																															
					za																															
					au																															
					-																															
					e																															

R	Z	S	a	N	In	in	o	P	jd	m	Z	C	C	C	C	c	c	c	C	c	v	S	d	L	d	L	d	d	N	N	D	s	ta	g	a	g	ab
				M	m	ty	n	m	Vo	a		z	m	a	e	t					e	zv	e	p	cl	p			g	Ba	w	re	st		bs		
us	UC	UC	UC	a	i-			He	JI	Vi	11											Se		Lo		Lp					Dr	3	St	7	2	82	2,9
e	Z	Zs	1		2-					tu	1											t		n		Sq					iv		R				
		ilo			Z1						1																												
					S0																																		
					a.																																		
					cz																																		
					za																																		
					at																																		
					us																																		
					-																																		
					e																																		

Aufzeichnungen über einzelne Messwerte

Mit Timestream for können Sie die Daten LiveAnalytics auch mit einer Messung pro Zeitreihendatensatz aufnehmen. Im Folgenden finden Sie die Schemadetails, wenn sie mithilfe von Datensätzen mit einer einzigen Kennzahl aufgenommen wurden.

Tabellenschema

Im Folgenden finden Sie das Tabellenschema, sobald die Daten mithilfe von Datensätzen mit mehreren Kennzahlen aufgenommen wurden. Es ist die Ausgabe einer Abfrage. DESCRIBE Unter der Annahme, dass die Daten in die Datenbank raw_data und die Tabelle devops aufgenommen wurden, finden Sie unten die Abfrage.

```
DESCRIBE "raw_data"."devops_single"
```

Spalte	Typ	LiveAnalytics Timestream für den Attributtyp
availability_zone	varchar	DIMENSION
microservice_name	varchar	DIMENSION
Instanzname	varchar	DIMENSION
Prozessname	varchar	DIMENSION
os_version	varchar	DIMENSION
jdk-Version	varchar	DIMENSION
Zelle	varchar	DIMENSION
Region	varchar	DIMENSION
Silo	varchar	DIMENSION
instance_type	varchar	DIMENSION
measure_name	varchar	MEASURE_NAME
time	Zeitstempel	TIMESTAMP
Messwert::doppelt	double	MEASURE_VALUE
Messwert::bigint	bigint	MEASURE_VALUE
Messwert::varchar	varchar	MEASURE_VALUE

Maßnahmenschema

Im Folgenden finden Sie das von der SHOW MEASURES Abfrage zurückgegebene Kennzahlschema.

```
SHOW MEASURES FROM "raw_data"."devops_single"
```

measure_name	data_type	Dimensionen
cpu_hi	double	[{'dimension_name': 'Verfügbarkeitszone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'os_version', 'data_type': 'varchar'}, {'dimension_name': 'zelle', 'data_type': 'varchar'}, {'dimension_name': 'region', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'instance_type', 'data_type': 'varchar'}]
CPU_Leerlauf	double	[{'dimension_name': 'Verfügbarkeitszone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'os_version', 'data_type': 'varchar'}, {'dimension_name': 'zelle', 'data_type': 'varchar'}, {'dimension_name': 'region', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'},

measure_name	data_type	Dimensionen
		{'dimension_name': 'instand ance_type', 'data_type': 'varchar'}]
cpu_iowait	double	[{'dimension_name': 'Verfügba rkeitszone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'os_versi on', 'data_type': 'varchar'}, {'dimension_name': 'zelle', 'data_type': 'varchar'}, {'dimension_name': 'region', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'instand ance_type', 'data_type': 'varchar'}]

measure_name	data_type	Dimensionen
CPU_nett	double	[{'dimension_name': 'Verfügbarkeitszone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'os_version', 'data_type': 'varchar'}, {'dimension_name': 'zelle', 'data_type': 'varchar'}, {'dimension_name': 'region', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'instance_type', 'data_type': 'varchar'}]

measure_name	data_type	Dimensionen
cpu_si	double	[{'dimension_name': 'Verfügbarkeitszone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'os_version', 'data_type': 'varchar'}, {'dimension_name': 'zelle', 'data_type': 'varchar'}, {'dimension_name': 'region', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'instance_type', 'data_type': 'varchar'}]

measure_name	data_type	Dimensionen
CPU_stehlen	double	[{'dimension_name': 'Verfügbarkeitszone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'os_version', 'data_type': 'varchar'}, {'dimension_name': 'zelle', 'data_type': 'varchar'}, {'dimension_name': 'region', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'instandance_type', 'data_type': 'varchar'}]

measure_name	data_type	Dimensionen
CPU-system	double	[{'dimension_name': 'Verfügbarkeitszone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'os_version', 'data_type': 'varchar'}, {'dimension_name': 'zelle', 'data_type': 'varchar'}, {'dimension_name': 'region', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'instandance_type', 'data_type': 'varchar'}]

measure_name	data_type	Dimensionen
CPU_Benutzer	double	[{'dimension_name': 'Verfügbarkeitszone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'os_version', 'data_type': 'varchar'}, {'dimension_name': 'zelle', 'data_type': 'varchar'}, {'dimension_name': 'region', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'instandance_type', 'data_type': 'varchar'}]

measure_name	data_type	Dimensionen
disk_free	double	[{'dimension_name': 'Verfügbarkeitszone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'os_version', 'data_type': 'varchar'}, {'dimension_name': 'zelle', 'data_type': 'varchar'}, {'dimension_name': 'region', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'instandance_type', 'data_type': 'varchar'}]

measure_name	data_type	Dimensionen
disk_io_reads	double	[{'dimension_name': 'Verfügbarkeitszone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'os_version', 'data_type': 'varchar'}, {'dimension_name': 'zelle', 'data_type': 'varchar'}, {'dimension_name': 'region', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'instandance_type', 'data_type': 'varchar'}]

measure_name	data_type	Dimensionen
disk_io_writes	double	[{'dimension_name': 'Verfügbarkeitszone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'os_version', 'data_type': 'varchar'}, {'dimension_name': 'zelle', 'data_type': 'varchar'}, {'dimension_name': 'region', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'instandance_type', 'data_type': 'varchar'}]

measure_name	data_type	Dimensionen
disk_used	double	[{'dimension_name': 'Verfügbarkeitszone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'os_version', 'data_type': 'varchar'}, {'dimension_name': 'zelle', 'data_type': 'varchar'}, {'dimension_name': 'region', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'instance_type', 'data_type': 'varchar'}]

measure_name	data_type	Dimensionen
Verwendete Dateideskriptoren	double	[{'dimension_name': 'Verfügbarkeitszone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'os_version', 'data_type': 'varchar'}, {'dimension_name': 'zelle', 'data_type': 'varchar'}, {'dimension_name': 'region', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'instandance_type', 'data_type': 'varchar'}]

measure_name	data_type	Dimensionen
gc_pause	double	[{'dimension_name': 'Verfügbarkeitszone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'prozessname', 'data_type': 'varchar'}, {'dimension_name': 'jdk_version', 'data_type': 'varchar'}, {'dimension_name': 'zelle', 'data_type': 'varchar'}, {'dimension_name': 'region', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}]

measure_name	data_type	Dimensionen
gc_zurückgewonnen	double	[{'dimension_name': 'Verfügbarkeitszone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'prozessname', 'data_type': 'varchar'}, {'dimension_name': 'jdk_version', 'data_type': 'varchar'}, {'dimension_name': 'zelle', 'data_type': 'varchar'}, {'dimension_name': 'region', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}]

measure_name	data_type	Dimensionen
Latenz pro Lesevorgang	double	[{'dimension_name': 'Verfügbarkeitszone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'os_version', 'data_type': 'varchar'}, {'dimension_name': 'zelle', 'data_type': 'varchar'}, {'dimension_name': 'region', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'instance_type', 'data_type': 'varchar'}]

measure_name	data_type	Dimensionen
Latenz pro Schreibvorgang	double	[{'dimension_name': 'Verfügbarkeitszone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'os_version', 'data_type': 'varchar'}, {'dimension_name': 'zelle', 'data_type': 'varchar'}, {'dimension_name': 'region', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'instandance_type', 'data_type': 'varchar'}]

measure_name	data_type	Dimensionen
memory_cached	double	[{'dimension_name': 'Verfügbarkeitszone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'os_version', 'data_type': 'varchar'}, {'dimension_name': 'zelle', 'data_type': 'varchar'}, {'dimension_name': 'region', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'instandance_type', 'data_type': 'varchar'}]

measure_name	data_type	Dimensionen
speicherfrei	double	[{'dimension_name': 'Verfügbarkeitszone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'prozessname', 'data_type': 'varchar'}, {'dimension_name': 'os_version', 'data_type': 'varchar'}, {'dimension_name': 'jdk_version', 'data_type': 'varchar'}, {'dimension_name': 'zelle', 'data_type': 'varchar'}, {'dimension_name': 'dimension_name' ':' Region ',' data_type ':' varchar '}, {' dimension_name ':' silo ',' data_type ':' varchar '}, {' dimension_name ':' instance_type ', 'data_ Typ: 'varchar'}]

measure_name	data_type	Dimensionen
verwendeter Speicher	double	[{'dimension_name': 'Verfügbarkeitszone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'os_version', 'data_type': 'varchar'}, {'dimension_name': 'zelle', 'data_type': 'varchar'}, {'dimension_name': 'region', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'instance_type', 'data_type': 'varchar'}]

measure_name	data_type	Dimensionen
Netzwerk-Bytes-In	double	[{'dimension_name': 'Verfügbarkeitszone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'os_version', 'data_type': 'varchar'}, {'dimension_name': 'zelle', 'data_type': 'varchar'}, {'dimension_name': 'region', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'instandance_type', 'data_type': 'varchar'}]

measure_name	data_type	Dimensionen
Netzwerk-Bytes_Out	double	[{'dimension_name': 'Verfügbarkeitszone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'os_version', 'data_type': 'varchar'}, {'dimension_name': 'zelle', 'data_type': 'varchar'}, {'dimension_name': 'region', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'instandance_type', 'data_type': 'varchar'}]

measure_name	data_type	Dimensionen
aufgabe_abgeschlossen	bigint	[{'dimension_name': 'Verfügbarkeitszone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'prozessname', 'data_type': 'varchar'}, {'dimension_name': 'jdk_version', 'data_type': 'varchar'}, {'dimension_name': 'zelle', 'data_type': 'varchar'}, {'dimension_name': 'region', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}]

measure_name	data_type	Dimensionen
task_end_state	varchar	[{'dimension_name': 'Verfügbarkeitszone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'prozessname', 'data_type': 'varchar'}, {'dimension_name': 'jdk_version', 'data_type': 'varchar'}, {'dimension_name': 'zelle', 'data_type': 'varchar'}, {'dimension_name': 'region', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}]

Beispieldaten

availa ity_zo	micro ice_n	Instar me	Proze me	os_ve n	jdk-Versi on	Zelle	Regio	Silo	instar type	meas ame	Zeit	Mess :dopp	Mess :bigin	Messwert: :varchar
eu-west--1	Herku	I- -1-Zelle--silo-20000(mazocomzaZsvhercu		AL20		eu-west--zelle-	eu-west-	eu-west--zelle-silo-2	r5.4xle	CPU_	34:57	0,871		

availa ity_zo	micro ice_n	Instar me	Proze me	os_ve n	jdk- Versi on	Zelle	Regio	Silo	instar type	meas ame	Zeit	Mess :dopp	Mess :bigin	Messwert: :varchar
		eu- west												
eu- west- -1	Herku	I- - 1- Zelle- - silo-2 0000(mazo com zaZs\ hercu eu- west		AL20		eu- west- - zelle-	eu- west-	eu- west- - zelle- silo-2	r5.4xl e	CPU_ iv	34:57	3,462		
eu- west- -1	Herku	I- - 1- Zelle- - silo-2 0000(mazo com zaZs\ hercu eu- west		AL20		eu- west- - zelle-	eu- west-	eu- west- - zelle- silo-2	r5.4xl e	cpu_i t	34:57	0,122		

availa ity_zo	micro ice_n	Instar me	Proze me	os_ve n	jdk- Versi on	Zelle	Regio	Silo	instar type	meas ame	Zeit	Mess :dopp	Mess :bigin	Messwert: :varchar
eu- west- -1	Herku	I- - 1- Zelle- - silo-2 0000(mazo com zaZs\ hercu eu- west		AL20		eu- west- - zelle-	eu- west-	eu- west- - zelle- silo-2	r5.4xl e	CPU_	34:57	0,630		
eu- west- -1	Herku	I- - 1- Zelle- - silo-2 0000(mazo com zaZs\ hercu eu- west		AL20		eu- west- - zelle-	eu- west-	eu- west- - zelle- silo-2	r5.4xl e	CPU_	34:57	0,164		

availa ity_zo	micro ice_n	Instar me	Proze me	os_ve n	jdk- Versi on	Zelle	Regio	Silo	instar type	meas ame	Zeit	Mess :dopp	Mess :bigin	Messwert: :varchar
eu- west- -1	Herku	I- - 1- Zelle- - silo-2 0000(mazo com zaZs\ hercu eu- west		AL20		eu- west- - zelle-	eu- west-	eu- west- - zelle- silo-2	r5.4xl e	CPU_ en	34:57	0,107		
eu- west- -1	Herku	I- - 1- Zelle- - silo-2 0000(mazo com zaZs\ hercu eu- west		AL20		eu- west- - zelle-	eu- west-	eu- west- - zelle- silo-2	r5.4xl e	CPU- Syste m	34:57	0,457		

availa ity_zo	micro ice_n	Instar me	Proze me	os_ve n	jdk- Versi on	Zelle	Regio	Silo	instar type	meas ame	Zeit	Mess :dopp	Mess :bigin	Messwert: :varchar
eu- west- -1	Herku	I- - 1- Zelle- - silo-2 0000(mazo com zaZs\ hercu eu- west		AL20		eu- west- - zelle-	eu- west-	eu- west- - zelle- silo-2	r5.4xl e	CPU- Benu zer	34:57	94,20		
eu- west- -1	Herku	I- - 1- Zelle- - silo-2 0000(mazo com zaZs\ hercu eu- west		AL20		eu- west- - zelle-	eu- west-	eu- west- - zelle- silo-2	r5.4xl e	disk_	34:57	72,51		

availa ity_zo	micro ice_n	Instar me	Proze me	os_ve n	jdk- Versi on	Zelle	Regio	Silo	instar type	meas ame	Zeit	Mess :dopp	Mess :bigin	Messwert: :varchar
eu- west- -1	Herku	I- - 1- Zelle- - silo-2 0000(mazo com zaZs\ hercu eu- west		AL20		eu- west- - zelle-	eu- west-	eu- west- - zelle- silo-2	r5.4xl e	disk_i eads	34:57	81,73		
eu- west- -1	Herku	I- - 1- Zelle- - silo-2 0000(mazo com zaZs\ hercu eu- west		AL20		eu- west- - zelle-	eu- west-	eu- west- - zelle- silo-2	r5.4xl e	disk_i chreil	34:57	77,16		

availa ity_zo	micro ice_n	Instar me	Proze me	os_ve n	jdk- Versi on	Zelle	Regio	Silo	instar type	meas ame	Zeit	Mess :dopp	Mess :bigin	Messwert: :varchar
eu- west- -1	Herku	I- - 1- Zelle- - silo-2 0000(mazo com zaZs\ hercu eu- west		AL20		eu- west- - zelle-	eu- west-	eu- west- - zelle- silo-2	r5.4xl e	disk_	34:57	89,42		
eu- west- -1	Herku	I- - 1- Zelle- - silo-2 0000(mazo com zaZs\ hercu eu- west		AL20		eu- west- - zelle-	eu- west-	eu- west- - zelle- silo-2	r5.4xl e	Datei riptor werde verwe	34:57	30,08		

availa ity_zo	micro ice_n	Instar me	Proze me	os_ve n	jdk- Versi on	Zelle	Regio	Silo	instar type	meas ame	Zeit	Mess :dopp	Mess :bigin	Messwert: :varchar
eu- west- -1	Herku	I- - 1- Zelle- - silo-2 0000(mazo com zaZs\ hercu eu- west	serve		JDK_	eu- west- - zelle-	eu- west-	eu- west- - zelle- silo-2		gc_pa	34:57	60,28		
eu- west- -1	Herku	I- - 1- Zelle- - silo-2 0000(mazo com zaZs\ hercu eu- west	serve		JDK_	eu- west- - zelle-	eu- west-	eu- west- - zelle- silo-2		gc_zu gefor	34:57	75,28		

availa ity_zo	micro ice_n	Instar me	Proze me	os_ve n	jdk- Versi on	Zelle	Regio	Silo	instar type	meas ame	Zeit	Mess :dopp	Mess :bigin	Messwert: :varchar
eu- west- -1	Herku	I- - 1- Zelle- - silo-2 0000(mazo com zaZs\ hercu eu- west		AL20		eu- west- - zelle-	eu- west-	eu- west- - zelle- silo-2	r5.4xl e	Laten pro Lese\ ng	34:57	8,076		
eu- west- -1	Herku	I- - 1- Zelle- - silo-2 0000(mazo com zaZs\ hercu eu- west		AL20		eu- west- - zelle-	eu- west-	eu- west- - zelle- silo-2	r5.4xl e	Laten pro Schre rgang	34:57	58,12		

availa ity_zo	micro ice_n	Instar me	Proze me	os_ve n	jdk- Versi on	Zelle	Regic	Silo	instar type	meas ame	Zeit	Mess :dopp	Mess :bigin	Messwert: :varchar
eu- west- -1	Herku	I- - 1- Zelle- - silo-2 0000(mazo com zaZs\ hercu eu- west		AL20		eu- west- - zelle-	eu- west-	eu- west- - zelle- silo-2	r5.4xl e	Speic zwisc espei t	34:57	87,56		
eu- west- -1	Herku	I- - 1- Zelle- - silo-2 0000(mazo com zaZs\ hercu eu- west	serve		JDK_	eu- west- - zelle-	eu- west-	eu- west- - zelle- silo-2		speic rei	34:57	18,95		

availa ity_zo	micro ice_n	Instar me	Proze me	os_ve n	jdk- Versi on	Zelle	Regic	Silo	instar type	meas ame	Zeit	Mess :dopp	Mess :bigin	Messwert: :varchar
eu- west- -1	Herku	I- - 1- Zelle- - silo-2 0000(mazo com zaZs\ hercu eu- west		AL20		eu- west- - zelle-	eu- west-	eu- west- - zelle- silo-2	r5.4xl e	speic rei	34:57	97,20		
eu- west- -1	Herku	I- - 1- Zelle- - silo-2 0000(mazo com zaZs\ hercu eu- west		AL20		eu- west- - zelle-	eu- west-	eu- west- - zelle- silo-2	r5.4xl e	verwe er Speic	34:57	12,37		

availa ity_zo	micro ice_n	Instar me	Proze me	os_ve n	jdk- Versi on	Zelle	Regio	Silo	instar type	meas ame	Zeit	Mess :dopp	Mess :bigin	Messwert: :varchar
eu- west- -1	Herku	I- - 1- Zelle- - silo-2 0000(mazo com zaZs\ hercu eu- west		AL20		eu- west- - zelle-	eu- west-	eu- west- - zelle- silo-2	r5.4xl e	Netzv Bytes Ein gang	34:57	31,02		
eu- west- -1	Herku	I- - 1- Zelle- - silo-2 0000(mazo com zaZs\ hercu eu- west		AL20		eu- west- - zelle-	eu- west-	eu- west- - zelle- silo-2	r5.4xl e	Netzv Bytes	34:57	0,514		

availa ity_zo	micro ice_n	Instar me	Proze me	os_ve n	jdk-Versi on	Zelle	Regic	Silo	instar type	meas ame	Zeit	Mess :dopp	Mess :bigin	Messwert: :varchar
eu-west--1	Herku	I- - 1- Zelle- - silo-2 0000(mazo com zaZs\ hercu eu-west	serve		JDK_	eu-west- - zelle-	eu-west-	eu-west- - zelle-silo-2		Aufga bges(sen	34:57		69	
eu-west--1	Herku	I- - 1- Zelle- - silo-2 0000(mazo com zaZs\ hercu eu-west	serve		JDK_	eu-west- - zelle-	eu-west-	eu-west- - zelle-silo-2		task_ state	34:57			SUCCESS_ ITH_RESUL T

Geplante Abfragemuster

In diesem Abschnitt finden Sie einige gängige Muster, wie Sie Amazon Timestream for LiveAnalytics Scheduled Queries verwenden können, um Ihre Dashboards so zu optimieren, dass sie schneller und kostengünstiger geladen werden. In den folgenden Beispielen werden anhand eines DevOps

Anwendungsszenarios die wichtigsten Konzepte veranschaulicht, die für geplante Abfragen im Allgemeinen gelten, unabhängig vom Anwendungsszenario.

Geplante Abfragen in Timestream for LiveAnalytics ermöglichen es Ihnen, Ihre Abfragen mithilfe der gesamten SQL Oberfläche von Timestream for auszudrücken. LiveAnalytics Ihre Abfrage kann eine oder mehrere Quelltabellen enthalten, Aggregationen oder jede andere in der SQL Sprache von Timestream for LiveAnalytics zulässige Abfrage ausführen und dann die Ergebnisse der Abfrage in einer anderen Zieltabelle in Timestream for materialisieren. LiveAnalytics Der besseren Übersicht halber wird in diesem Abschnitt diese Zieltabelle einer geplanten Abfrage als abgeleitete Tabelle bezeichnet.

Im Folgenden sind die wichtigsten Punkte aufgeführt, die in diesem Abschnitt behandelt werden.

- Anhand eines einfachen Aggregats auf Flottenebene wird erklärt, wie Sie eine geplante Abfrage definieren und einige grundlegende Konzepte verstehen können.
- Wie Sie Ergebnisse aus dem Ziel einer geplanten Abfrage (der abgeleiteten Tabelle) mit den Ergebnissen aus der Quelltabelle kombinieren können, um die Kosten- und Leistungsvorteile einer geplanten Abfrage zu nutzen.
- Was sind Ihre Kompromisse bei der Konfiguration des Aktualisierungszeitraums der geplanten Abfragen?
- Verwendung von geplanten Abfragen für einige gängige Szenarien.
  - Verfolgt den letzten Datenpunkt von jeder Instanz vor einem bestimmten Datum.
  - Eindeutige Werte für eine Dimension, die zum Auffüllen von Variablen in einem Dashboard verwendet werden sollen.
- Wie Sie mit verspätet eingehenden Daten im Kontext von geplanten Abfragen umgehen.
- Wie Sie einmalige manuelle Ausführungen verwenden können, um eine Vielzahl von Szenarien zu bewältigen, die nicht direkt von automatisierten Triggern für geplante Abfragen abgedeckt werden.

Themen

Szenario

In den folgenden Beispielen wird ein DevOps Überwachungsszenario verwendet, das unter beschrieben wird. Beispielschema für geplante Abfragen

Die Beispiele enthalten die Definition einer geplanten Abfrage, in die Sie die entsprechenden Konfigurationen für den Empfang von Benachrichtigungen zum Ausführungsstatus für geplante Abfragen, für den Empfang von Berichten über Fehler, die bei der Ausführung einer geplanten Abfrage aufgetreten sind, und für die IAM Rolle festlegen können, die die geplante Abfrage zur Ausführung ihrer Operationen verwendet.

Sie können diese geplanten Abfragen erstellen, nachdem Sie die vorherigen Optionen ausgefüllt, die Zieltabelle (oder abgeleitete Tabelle) erstellt und die über die ausgeführt haben AWSCLI. Nehmen wir beispielsweise an, dass eine geplante Abfragedefinition in einer Datei gespeichert ist`scheduled_query_example.json`. Sie können die Abfrage mit dem CLI Befehl erstellen.

```
aws timestream-query create-scheduled-query --cli-input-json  file://
scheduled_query_example.json --profile aws_profile --region us-east-1
```

Im vorherigen Befehl muss das mit der Option --profile übergebene Profil über die entsprechenden Berechtigungen verfügen, um geplante Abfragen zu erstellen. Ausführliche Anweisungen zu den Richtlinien und Berechtigungen finden Sie unter Identitätsbasierte Richtlinien für geplante Abfragen.

Einfache Aggregate auf Flottenebene

In diesem ersten Beispiel werden Sie anhand eines einfachen Beispiels zur Berechnung von Aggregaten auf Flottenebene durch einige der grundlegenden Konzepte bei der Arbeit mit geplanten Abfragen geführt. Anhand dieses Beispiels lernen Sie Folgendes.

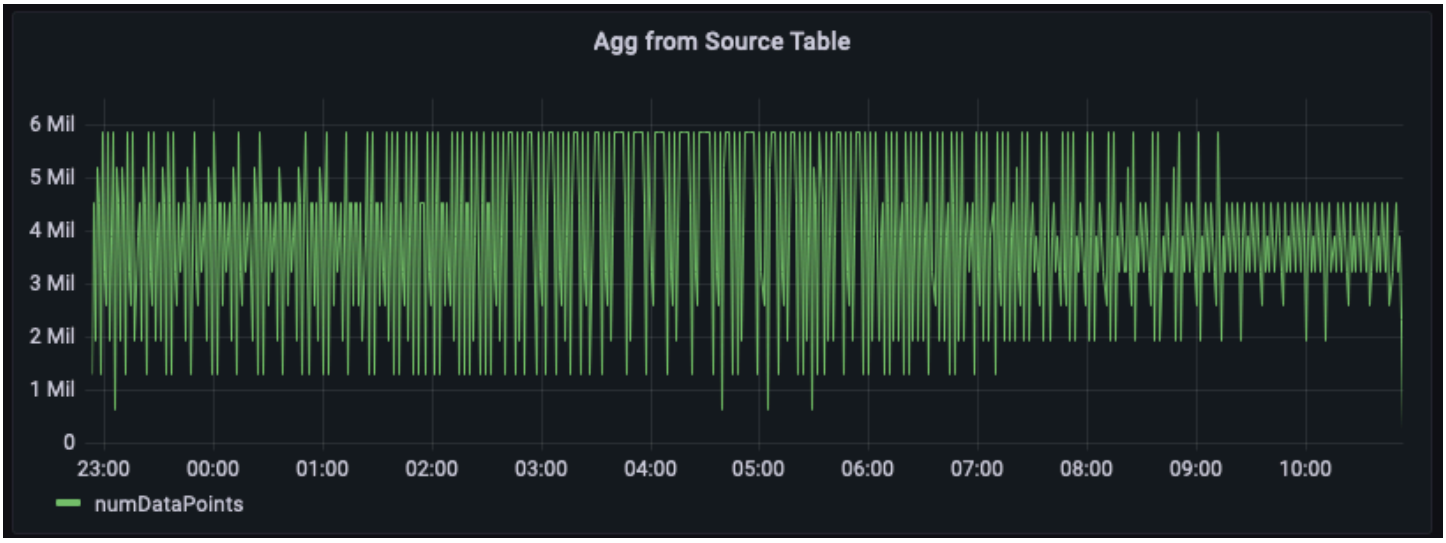
- So ordnen Sie Ihre Dashboard-Abfrage, die zum Abrufen aggregierter Statistiken verwendet wird, einer geplanten Abfrage zu.
- Wie Timestream for die Ausführung der verschiedenen Instanzen Ihrer geplanten Abfrage LiveAnalytics verwaltet.
- Wie Sie dafür sorgen können, dass sich verschiedene Instanzen von geplanten Abfragen in den Zeiträumen überschneiden, und wie die Richtigkeit der Daten in der Zieltabelle gewahrt wird, um sicherzustellen, dass Ihr Dashboard, das die Ergebnisse der geplanten Abfrage verwendet, Ihnen Ergebnisse liefert, die mit demselben Aggregat übereinstimmen, das anhand der Rohdaten berechnet wurde.
- So legen Sie den Zeitraum und die Aktualisierungsfrequenz für Ihre geplante Abfrage fest.

- Wie Sie die Ergebnisse der geplanten Abfragen in Eigenregie nachverfolgen können, um sie so zu optimieren, dass die Ausführungslatenz für die Abfrageinstanzen innerhalb der akzeptablen Verzögerungen für die Aktualisierung Ihrer Dashboards liegt.

Themen

Aus Quelltabellen aggregieren

In diesem Beispiel verfolgen Sie die Anzahl der Metriken, die von den Servern in einer bestimmten Region in jeder Minute ausgegeben werden. Die folgende Grafik ist ein Beispiel für die Darstellung dieser Zeitreihe für die Region us-east-1.



Im Folgenden finden Sie eine Beispielabfrage zur Berechnung dieses Aggregats aus den Rohdaten. Es filtert die Zeilen für die Region us-east-1 und berechnet dann die Summe pro Minute, indem es die 20 Metriken (wenn measure_name Metriken ist) oder 5 Ereignisse (wenn measure_name Events ist) berücksichtigt. In diesem Beispiel zeigt die grafische Darstellung, dass die Anzahl der ausgegebenen Metriken zwischen 1,5 Millionen und 6 Millionen pro Minute variiert. Wenn diese Zeitreihe für mehrere Stunden (in dieser Abbildung die letzten 12 Stunden) dargestellt wird, analysiert diese Abfrage anhand der Rohdaten Hunderte von Millionen von Zeilen.

```
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
```

Geplante Abfrage zur Vorberechnung von Aggregaten

Wenn Sie Ihre Dashboards so optimieren möchten, dass sie schneller geladen werden und Ihre Kosten senken, indem Sie weniger Daten scannen, können Sie eine geplante Abfrage verwenden, um diese Aggregate vorab zu berechnen. Geplante Abfragen in Timestream for LiveAnalytics ermöglichen es Ihnen, diese Vorberechnungen in einem anderen Timestream for LiveAnalytics table zu materialisieren, den Sie anschließend für Ihre Dashboards verwenden können.

Der erste Schritt bei der Erstellung einer geplanten Abfrage besteht darin, die Abfrage zu identifizieren, die Sie vorab berechnen möchten. Beachten Sie, dass das vorherige Dashboard für die Region us-east-1 gezeichnet wurde. Ein anderer Benutzer möchte jedoch möglicherweise dasselbe Aggregat für eine andere Region, z. B. us-west-2 oder eu-west-1. Um zu vermeiden, dass für jede dieser Abfragen eine geplante Abfrage erstellt wird, können Sie das Aggregat für jede Region vorab berechnen und die Aggregate pro Region in einem anderen Timestream für eine Tabelle materialisieren. LiveAnalytics

Die folgende Abfrage enthält ein Beispiel für die entsprechende Vorberechnung. Wie Sie sehen, ähnelt sie dem allgemeinen Tabellenausdruck grouped_data, der in der Abfrage der Rohdaten verwendet wird, mit Ausnahme von zwei Unterschieden: 1) Er verwendet kein Regionsprädikat, sodass wir eine Abfrage verwenden können, um alle Regionen vorab zu berechnen; und 2) er verwendet ein parametrisiertes Zeitprädikat mit einem speziellen Parameter @scheduled_runtime, der weiter unten ausführlich erklärt wird.

```
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
```

Die vorherige Abfrage kann mit der folgenden Spezifikation in eine geplante Abfrage konvertiert werden. Der geplanten Abfrage wird ein Name zugewiesen, bei dem es sich um eine benutzerfreundliche Mnemonik handelt. Sie enthält dann das QueryString, a ScheduleConfiguration, was ein Cron-Ausdruck ist. Es gibt an, für TargetConfiguration welche die Abfrageergebnisse der Zieltabelle in Timestream zugeordnet werden. LiveAnalytics Schließlich gibt es eine Reihe weiterer Konfigurationen an, z. B. die NotificationConfiguration, bei der Benachrichtigungen für einzelne Ausführungen der Abfrage gesendet werden, ErrorReportConfiguration bei denen ein Bericht erstellt wird, falls bei der Abfrage Fehler auftreten, und die Rolle ScheduledQueryExecutionRoleArn, die zur Ausführung von Vorgängen für die geplante Abfrage verwendet wird.

```
{
    "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": "******"
}
```

Im Beispiel ist das ScheduleExpression Cron (0/5 * * *? *) bedeutet, dass die Abfrage einmal alle 5 Minuten zur 5., 10., 15.,.. Minute jeder Stunde eines jeden Tages ausgeführt wird. Diese Zeitstempel, wenn eine bestimmte Instanz dieser Abfrage ausgelöst wird, entsprechen dem in der Abfrage verwendeten @scheduled_runtime -Parameter. Stellen Sie sich zum Beispiel die Instanz dieser geplanten Abfrage vor, die am 01.12.2021 00:00:00 ausgeführt wird. Für diese Instanz wird der @scheduled_runtime -Parameter beim Aufrufen der Abfrage mit dem Zeitstempel 2021-12-01 00:00:00 initialisiert. Daher wird diese spezielle Instanz zum Zeitstempel 2021-12-01 00:00:00 ausgeführt und berechnet die Aggregate pro Minute für den Zeitraum 2021-11-30 23:50:00 bis 2021-12-01 00:01:00. In ähnlicher Weise wird die nächste Instanz dieser Abfrage mit dem Zeitstempel 2021-12-01 00:05:00 ausgelöst. In diesem Fall berechnet die Abfrage Aggregate pro Minute aus dem Zeitraum 2021-11-30 23:55:00 bis 2021-12-01 00:06:00. Daher stellt der @scheduled_runtime -Parameter eine geplante Abfrage bereit, um die Aggregate für die konfigurierten Zeitbereiche anhand der Aufrufzeit für die Abfragen vorab zu berechnen.

Beachten Sie, dass sich zwei aufeinanderfolgende Instanzen der Abfrage in ihren Zeitbereichen überschneiden. Dies können Sie je nach Ihren Anforderungen steuern. In diesem Fall können diese Abfragen aufgrund dieser Überschneidung die Aggregate auf der Grundlage aller Daten aktualisieren, deren Eingang leicht verzögert war, in diesem Beispiel bis zu 5 Minuten. Um die Richtigkeit der materialisierten Abfragen sicherzustellen, LiveAnalytics stellt Timestream for sicher, dass die Abfrage am 01.12.2021 00:05:00 erst ausgeführt wird, nachdem die Abfrage am 01.12.2021 00:00:00 abgeschlossen ist. Die Ergebnisse der letztgenannten Abfragen können jedes zuvor materialisierte Aggregat aktualisieren, wenn ein neuerer Wert generiert wird. Wenn beispielsweise einige Daten mit dem Zeitstempel 2021-11-30 23:59:00 eingetroffen sind, nachdem die Abfrage für 2021-12-01

00:00:00 ausgeführt wurde, aber vor der Abfrage für 2021-12-01 00:05:00, dann berechnet die Ausführung am 2021-12-01 00:05:00 die Aggregate für die Minute 2021-11-30 23:59:00 neu und dies führt dazu, dass das vorherige Aggregat mit dem neu berechneten Wert aktualisiert wird. Sie können sich auf diese Semantik der geplanten Abfragen verlassen, um einen Kompromiss zwischen der Geschwindigkeit, mit der Sie Ihre Vorberechnungen aktualisieren, und der Art und Weise, wie Sie einige Daten bei verspäteter Ankunft elegant verarbeiten können, zu finden. Im Folgenden werden weitere Überlegungen dazu erörtert, wie Sie diesen Aktualisierungsrhythmus mit der Aktualität der Daten abwägen und wie Sie die Aktualisierung der Aggregate für Daten angehen, die noch verzögerter ankommen, oder ob Ihre Quelle der geplanten Berechnung aktualisierte Werte enthält, die eine Neuberechnung der Aggregate erfordern würden.

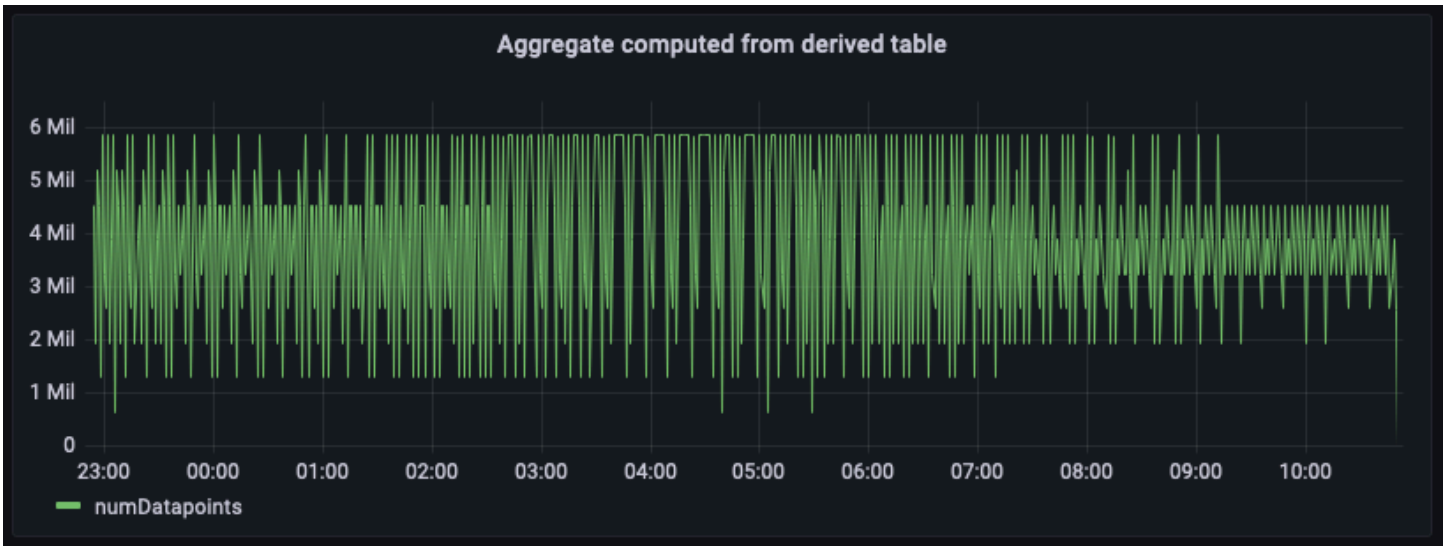
Jede geplante Berechnung hat eine Benachrichtigungskonfiguration, bei der Timestream for LiveAnalytics über jede Ausführung einer geplanten Konfiguration eine Benachrichtigung sendet. Sie können ein SNS Thema so konfigurieren, dass es Benachrichtigungen für jeden Aufruf erhält. Neben dem Erfolgs- oder Fehlerstatus einer bestimmten Instanz verfügt sie auch über verschiedene Statistiken, z. B. die Zeit, die für die Ausführung dieser Berechnung benötigt wurde, die Anzahl der Byte, die die Berechnung gescannt hat, und die Anzahl der Byte, die die Berechnung in die Zieltabelle geschrieben hat. Sie können diese Statistiken verwenden, um Ihre Abfrage weiter zu optimieren, die Konfiguration zu planen oder die Ausgaben für Ihre geplanten Abfragen nachzuverfolgen. Ein erwähnenswerter Aspekt ist die Ausführungszeit einer Instanz. In diesem Beispiel ist die geplante Berechnung so konfiguriert, dass sie alle 5 Minuten ausgeführt wird. Die Ausführungszeit bestimmt die Verzögerung, mit der die Vorberechnung verfügbar sein wird. Dies bestimmt auch die Verzögerung in Ihrem Dashboard, wenn Sie die vorberechneten Daten in Ihren Dashboards verwenden. Wenn diese Verzögerung durchweg höher als das Aktualisierungsintervall ist, z. B. wenn die Ausführungszeit für eine Berechnung, die so konfiguriert ist, dass sie alle 5 Minuten aktualisiert wird, mehr als 5 Minuten beträgt, ist es wichtig, Ihre Berechnung so zu optimieren, dass sie schneller ausgeführt wird, um weitere Verzögerungen in Ihren Dashboards zu vermeiden.

Aus abgeleiteter Tabelle aggregieren

Nachdem Sie die geplanten Abfragen eingerichtet haben und die Aggregate vorab berechnet und in einem anderen Timestream für die LiveAnalytics Tabelle materialisiert wurden, der in der Zielkonfiguration der geplanten Berechnung angegeben ist, können Sie die Daten in dieser Tabelle verwenden, um Abfragen für Ihre Dashboards zu schreibenSQL. Im Folgenden finden Sie ein Äquivalent zu der Abfrage, die die materialisierten Voraggregate verwendet, um das Aggregat für die Anzahl der Datenpunkte pro Minute für us-east-1 zu generieren.

```
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
```



In der vorherigen Abbildung wird das anhand der Aggregattabelle berechnete Aggregat dargestellt. Wenn Sie dieses Panel mit dem aus den Rohquelldaten berechneten Panel vergleichen, werden Sie feststellen, dass sie exakt übereinstimmen, obwohl diese Aggregate um einige Minuten verzögert sind. Dies hängt vom Aktualisierungsintervall ab, das Sie für die geplante Berechnung konfiguriert haben, sowie von der Zeit, zu der sie ausgeführt werden soll.

Bei dieser Abfrage anhand der vorberechneten Daten werden im Vergleich zu den Aggregaten, die anhand der Rohquelldaten berechnet wurden, mehrere Größenordnungen weniger Daten gescannt. Abhängig von der Granularität der Aggregationen kann diese Reduzierung leicht zu einer 100-mal geringeren Kosten und Latenz bei Abfragen führen. Die Ausführung dieser geplanten Berechnung ist mit Kosten verbunden. Je nachdem, wie oft diese Dashboards aktualisiert werden und wie viele Benutzer diese Dashboards gleichzeitig laden, können Sie Ihre Gesamtkosten durch die Verwendung dieser Vorberechnungen jedoch erheblich senken. Und das zusätzlich zu den 10- bis 100-mal schnelleren Ladezeiten für die Dashboards.

Aggregat, das Quell- und abgeleitete Tabellen kombiniert

Dashboards, die mit den abgeleiteten Tabellen erstellt wurden, können eine Verzögerung aufweisen. Wenn Ihr Anwendungsszenario erfordert, dass die Dashboards über die neuesten Daten verfügen,

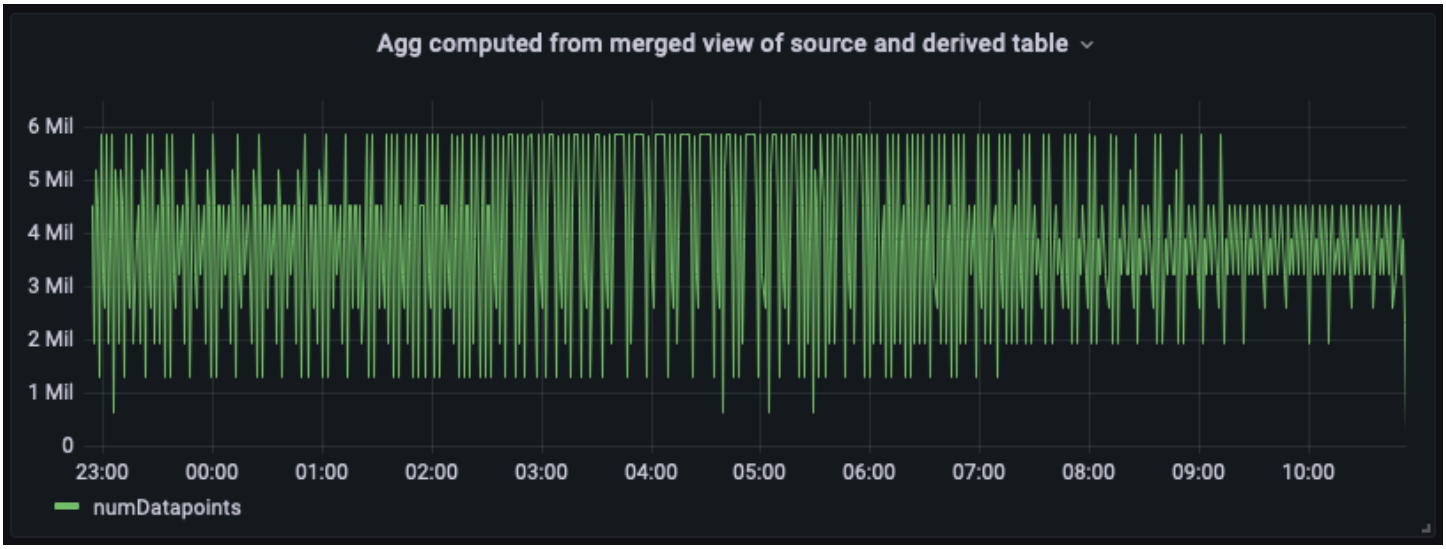
können Sie die Leistungsfähigkeit und Flexibilität der SQL Unterstützung von Timestream for LiveAnalytics nutzen, um die neuesten Daten aus der Quelltabelle mit den historischen Aggregaten aus der abgeleiteten Tabelle zu kombinieren, um eine zusammengeführte Ansicht zu erstellen. Diese zusammengeführte Ansicht verwendet die Union-Semantik SQL und die sich nicht überschneidenden Zeitbereiche aus der Quelle und der abgeleiteten Tabelle. Im folgenden Beispiel verwenden wir das Wort „abgeleitete“.“ abgeleitete Tabelle „per_minute_aggs_pt5m“. Da die geplante Berechnung für diese abgeleitete Tabelle einmal alle 5 Minuten aktualisiert wird (gemäß der Spezifikation für den Zeitplanausdruck), verwendet die folgende Abfrage die letzten 15 Minuten an Daten aus der Quelltabelle und alle Daten, die älter als 15 Minuten sind, aus der abgeleiteten Tabelle und vereint dann die Ergebnisse, um die zusammengeführte Ansicht zu erstellen, die das Beste aus beiden Welten bietet: Wirtschaftlichkeit und geringe Latenz durch Lesen älterer vorberechneter Aggregate aus der abgeleiteten Tabelle und die Aktualität der Aggregate aus der Quelle Tabelle zur Unterstützung Ihrer Anwendungsfälle für Echtzeitanalysen.

Beachten Sie, dass dieser Union-Ansatz im Vergleich zur reinen Abfrage der abgeleiteten Tabelle eine etwas höhere Abfragelatenz hat und dass auch etwas mehr Daten gescannt werden, da die Rohdaten in Echtzeit aggregiert werden, um das letzte Zeitintervall auszufüllen. Diese zusammengeführte Ansicht ist jedoch immer noch deutlich schneller und kostengünstiger als die spontane Aggregation aus der Quelltabelle, insbesondere bei Dashboards, die Daten über Tage oder Wochen rendern. Sie können die Zeitbereiche für dieses Beispiel an die Aktualisierungsanforderungen und die Verzögerungstoleranz Ihrer Anwendung anpassen.

```
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
```

Unten finden Sie das Dashboard-Panel mit dieser vereinheitlichten zusammengeführten Ansicht. Wie Sie sehen können, sieht das Dashboard fast identisch mit der Ansicht aus, die anhand der abgeleiteten Tabelle berechnet wurde, mit der Ausnahme, dass sich die meisten up-to-date Aggregate ganz rechts oben befinden.



Aggregat aus häufig aktualisierten geplanten Berechnungen

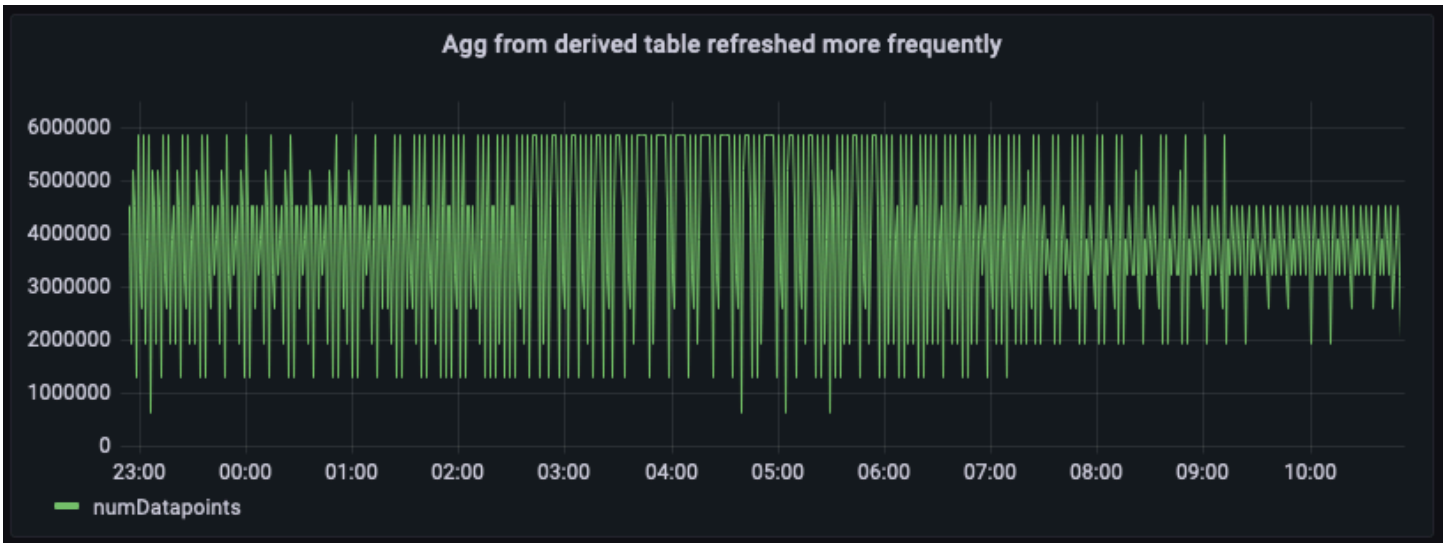
Je nachdem, wie häufig Ihre Dashboards geladen werden und wie viel Latenz Sie für Ihr Dashboard wünschen, gibt es einen anderen Ansatz, um aktuellere Ergebnisse in Ihrem Dashboard zu erzielen: die Aggregate bei der geplanten Berechnung häufiger aktualisieren zu lassen. Im Folgenden finden Sie beispielsweise die Konfiguration derselben geplanten Berechnung, mit der Ausnahme, dass sie einmal pro Minute aktualisiert wird (beachten Sie den Zeitplan-Express-Cron (0/1 * *? *)). Bei dieser Konfiguration verfügt die abgeleitete Tabelle per_minute_aggs_pt1m über wesentlich neuere Aggregate als in dem Szenario, in dem die Berechnung einen Aktualisierungszeitplan von einmal alle 5 Minuten vorsah.

```
{
    "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
```

Da die abgeleitete Tabelle neuere Aggregate enthält, können Sie jetzt die abgeleitete Tabelle per_minute_aggs_pt1m direkt abfragen, um aktuellere Aggregate zu erhalten, wie aus der vorherigen Abfrage und dem Dashboard-Schnappschuss unten ersichtlich ist.



Beachten Sie, dass die Aktualisierung der geplanten Berechnung nach einem schnelleren Zeitplan (z. B. 1 Minute im Vergleich zu 5 Minuten) die Wartungskosten für die geplante Berechnung erhöhen wird. Die Benachrichtigung für die Ausführung jeder Berechnung enthält Statistiken darüber, wie viele Daten gescannt und wie viele in die abgeleitete Tabelle geschrieben wurden. Wenn Sie die zusammengeführte Ansicht verwenden, um die abgeleitete Tabelle zu vereinigen, fragen Sie entsprechend die Kosten in der zusammengeführten Ansicht ab, und die Latenz beim Laden des Dashboards ist höher als beim bloßen Abfragen der abgeleiteten Tabelle. Daher hängt der von Ihnen gewählte Ansatz davon ab, wie oft Ihre Dashboards aktualisiert werden und wie hoch die Wartungskosten für die geplanten Abfragen sind. Wenn Sie Dutzende von Benutzern haben, die die Dashboards etwa einmal pro Minute aktualisieren, führt eine häufigere Aktualisierung Ihrer abgeleiteten Tabelle wahrscheinlich zu insgesamt niedrigeren Kosten.

Letzter Punkt von jedem Gerät

In Ihrer Anwendung müssen Sie möglicherweise die letzte von einem Gerät ausgegebene Messung ablesen. Es kann allgemeinere Anwendungsfälle geben, um die letzte Messung für ein Gerät vor einer bestimmten Messung zu ermittelndate/time or the first measurement for a device after a given date/time. Wenn Sie über Millionen von Geräten und jahrelange Daten verfügen, erfordert diese Suche möglicherweise das Scannen großer Datenmengen.

Im Folgenden finden Sie ein Beispiel dafür, wie Sie geplante Abfragen verwenden können, um die Suche nach dem letzten von einem Gerät ausgegebenen Punkt zu optimieren. Sie können dasselbe Muster auch verwenden, um die erste Punktabfrage zu optimieren, falls Ihre Anwendung sie benötigt.

Themen

Aus der Quelltabelle berechnet

Im Folgenden finden Sie eine Beispielabfrage, um die letzte Messung zu ermitteln, die von den Diensten in einer bestimmten Bereitstellung ausgegeben wurde (z. B. Server für einen bestimmten Microservice innerhalb einer bestimmten Region, Zelle, Silo und Availability_Zone). In der Beispielanwendung gibt diese Abfrage die letzte Messung für Hunderte von Servern zurück. Beachten Sie auch, dass diese Abfrage über ein unbegrenztes Zeitprädikat verfügt und nach Daten sucht, die älter als ein bestimmter Zeitstempel sind.

> **Note**
>
> Hinweise zu den Funktionen `max` und `max_by` finden Sie unter. Aggregationsfunktionen

```
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
```

Abgeleitete Tabelle zur Vorberechnung mit täglicher Granularität

Sie können den vorherigen Anwendungsfall in eine geplante Berechnung umwandeln. Wenn Ihre Anwendungsanforderungen so sind, dass Sie diese Werte möglicherweise für Ihre gesamte Flotte in mehreren Regionen, Zellen, Silos, Verfügbarkeitszonen und Microservices ermitteln müssen, können Sie die Werte für Ihre gesamte Flotte anhand einer Zeitplanberechnung vorab berechnen. Das ist die Stärke der serverlosen geplanten Abfragen von Timestream for, mit LiveAnalytics der diese Abfragen an die Skalierungsanforderungen Ihrer Anwendung angepasst werden können.

Im Folgenden finden Sie eine Abfrage zur Vorberechnung des letzten Punkts auf allen Servern für einen bestimmten Tag. Beachten Sie, dass die Abfrage nur ein Zeitprädikat und kein Prädikat für die Dimensionen hat. Das Zeitprädikat beschränkt die Abfrage auf den letzten Tag ab dem Zeitpunkt, zu dem die Berechnung auf der Grundlage des angegebenen Zeitplanausdrucks ausgelöst wurde.

```
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
```

Im Folgenden finden Sie eine Konfiguration für die geplante Berechnung unter Verwendung der vorherigen Abfrage, bei der diese Abfrage UTC täglich um 01:00 Uhr ausgeführt wird, um das Aggregat für den vergangenen Tag zu berechnen. Der Zeitplanausdruck cron (0, 1) *? *) steuert dieses Verhalten und wird eine Stunde nach Tagesende ausgeführt, um alle Daten zu berücksichtigen, die bis zu einem Tag zu spät eintreffen.

```
{
    "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": "******"
}
```

Aus einer abgeleiteten Tabelle berechnet

Sobald Sie die abgeleitete Tabelle mit der vorherigen Konfiguration definiert haben und mindestens eine Instanz der geplanten Abfrage Daten in der abgeleiteten Tabelle materialisiert hat, können Sie nun die abgeleitete Tabelle abfragen, um die neueste Messung zu erhalten. Im Folgenden finden Sie eine Beispielabfrage für die abgeleitete Tabelle.

```
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
```

Kombination aus Quelle und abgeleiteter Tabelle

Ähnlich wie im vorherigen Beispiel werden für alle Daten aus der abgeleiteten Tabelle nicht die neuesten Schreibvorgänge vorgenommen. Daher können Sie erneut ein ähnliches Muster wie zuvor verwenden, um die Daten aus der abgeleiteten Tabelle für die älteren Daten zusammenzuführen und die Quelldaten für den verbleibenden Tipp zu verwenden. Im Folgenden finden Sie ein Beispiel für eine solche Abfrage, die einen ähnlichen UNION Ansatz verwendet. Da die Anwendung die letzte Messung vor einem bestimmten Zeitraum ermitteln muss und diese Startzeit in der Vergangenheit liegen kann, verwenden Sie beim Schreiben dieser Abfrage die angegebene Zeit, verwenden die Quelldaten, die bis zu einem Tag alt sind, und verwenden dann die abgeleitete Tabelle für die älteren Daten. Wie Sie dem nachfolgenden Abfragebeispiel entnehmen können, ist das Zeitprädikat für die Quelldaten begrenzt. Dadurch wird eine effiziente Verarbeitung in der Quelltabelle gewährleistet, die ein deutlich höheres Datenvolumen aufweist, und dann befindet sich das Prädikat für unbegrenzte Zeit in der abgeleiteten Tabelle.

```
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
```

Das Vorstehende ist nur ein Beispiel dafür, wie Sie die abgeleiteten Tabellen strukturieren können. Wenn Sie über jahrelange Daten verfügen, können Sie mehr Aggregationsebenen verwenden. Sie können beispielsweise monatliche Aggregate zusätzlich zu den täglichen Aggregaten haben, und Sie können stündliche Aggregate vor den täglichen Aggregaten haben. Sie können also die neuesten zusammenführen, um die letzte Stunde auszufüllen, die stündliche, um den letzten Tag auszufüllen, die täglichen, um den letzten Monat auszufüllen, und die monatlichen, um die ältere auszufüllen. Die Anzahl der Ebenen, die Sie einrichten, im Vergleich zum Aktualisierungszeitplan hängt von Ihren Anforderungen ab, d. h. davon, wie häufig diese Abfragen auftreten und wie viele Benutzer diese Abfragen gleichzeitig stellen.

Eindeutige Dimensionswerte

Möglicherweise haben Sie einen Anwendungsfall, in dem Sie Dashboards haben, in denen Sie die Einzelwerte von Dimensionen als Variablen verwenden möchten, um die Metriken zu untersuchen, die einem bestimmten Datensegment entsprechen. Der folgende Snapshot ist ein Beispiel, bei dem das Dashboard die Einzelwerte verschiedener Dimensionen wie Region, Zelle, Silo, Microservice und Availability_Zone vorab ausfüllt. Hier zeigen wir ein Beispiel dafür, wie Sie geplante Abfragen verwenden können, um die Berechnung dieser unterschiedlichen Werte dieser Variablen anhand der Messwerte, die Sie verfolgen, erheblich zu beschleunigen.

Themen

Zu Rohdaten

Sie können SELECT DISTINCT es verwenden, um die verschiedenen Werte zu berechnen, die sich aus Ihren Daten ergeben. Wenn Sie beispielsweise die unterschiedlichen Werte einer Region ermitteln möchten, können Sie die Abfrage in diesem Formular verwenden.

```
SELECT DISTINCT region
FROM "raw_data"."devops"
WHERE time > ago(1h)
ORDER BY 1
```

Möglicherweise verfolgen Sie Millionen von Geräten und Milliarden von Zeitreihen. In den meisten Fällen beziehen sich diese interessanten Variablen jedoch auf Dimensionen mit niedrigerer Kardinalität, bei denen es sich um einige bis zehn Werte handelt. Das DISTINCT Rechnen mit Rohdaten kann das Scannen großer Datenmengen erfordern.

Eindeutige Dimensionswerte vorab berechnen

Sie möchten, dass diese Variablen schnell geladen werden, damit Ihre Dashboards interaktiv sind. Darüber hinaus werden diese Variablen häufig bei jedem Ladevorgang des Dashboards berechnet, sodass sie auch kostengünstig sein sollten. Sie können die Suche nach diesen Variablen optimieren, indem Sie geplante Abfragen verwenden und sie in einer abgeleiteten Tabelle materialisieren.

Zunächst müssen Sie die Dimensionen identifizieren, für die Sie die DISTINCT Werte oder Spalten berechnen müssen, die Sie bei der Berechnung des Werts in den Prädikaten verwenden werden. DISTINCT

In diesem Beispiel können Sie sehen, dass das Dashboard unterschiedliche Werte für die Dimensionen Region, Zelle, Silo, Availability_Zone und Microservice auffüllt. Sie können also die folgende Abfrage verwenden, um diese eindeutigen Werte vorab zu berechnen.

```
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
```

Hier sind einige wichtige Dinge zu beachten.

- Sie können eine geplante Berechnung verwenden, um Werte für viele verschiedene Abfragen vorab zu berechnen. Beispielsweise verwenden Sie die vorherige Abfrage, um Werte für fünf

verschiedene Variablen vorab zu berechnen. Sie benötigen also nicht für jede Variable einen. Sie können dasselbe Muster verwenden, um gemeinsame Berechnungen in mehreren Panels zu identifizieren und so die Anzahl der geplanten Abfragen zu optimieren, die Sie verwalten müssen.

- Bei den Einzelwerten der Dimensionen handelt es sich nicht grundsätzlich um Zeitreihendaten. Sie konvertieren dies also mit dem @scheduled_runtime in Zeitreihen. Indem Sie diese Daten mit dem @scheduled_runtime -Parameter verknüpfen, können Sie auch verfolgen, welche Einzelwerte zu einem bestimmten Zeitpunkt erschienen sind, und so Zeitreihendaten daraus erstellen.
- Im vorherigen Beispiel sehen Sie, wie ein Metrikwert verfolgt wird. In diesem Beispiel wird COUNT (*) verwendet. Sie können andere aussagekräftige Aggregate berechnen, wenn Sie sie für Ihre Dashboards verfolgen möchten.

Im Folgenden finden Sie eine Konfiguration für eine geplante Berechnung unter Verwendung der vorherigen Abfrage. In diesem Beispiel ist sie so konfiguriert, dass sie alle 15 Minuten mit dem Zeitplanausdruck cron (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": "******"
}
```

Berechnung der Variablen aus der abgeleiteten Tabelle

Sobald bei der geplanten Berechnung die Einzelwerte in der abgeleiteten Tabelle hc_unique_dimensions_pt15m vormaterialisiert wurden, können Sie die abgeleitete Tabelle verwenden, um die Einzelwerte der Dimensionen effizient zu berechnen. Im Folgenden finden

Sie Beispielabfragen zur Berechnung der Einzelwerte und zur Verwendung anderer Variablen als Prädikate in diesen Einzelwertabfragen.

Region

```
SELECT DISTINCT region
FROM "derived"."hc_unique_dimensions_pt15m"
WHERE time > ago(1h)
ORDER BY 1
```

Zelle

```
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
```

Mikroservice

```
SELECT DISTINCT microservice_name
FROM "derived"."hc_unique_dimensions_pt15m"
WHERE time > ago(1h)
   AND region = '${region}' AND cell = '${cell}'
ORDER BY 1
```

Availability Zone

```
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
```

Umgang mit spät eintreffenden Daten

Möglicherweise gibt es Szenarien, in denen Daten sehr spät ankommen, z. B. wenn der Zeitpunkt, zu dem die Daten in Timestream aufgenommen wurden, im Vergleich zu dem Zeitstempel, der den aufgenommenen Zeilen zugeordnet LiveAnalytics ist, erheblich verzögert ist. In den vorherigen Beispielen haben Sie gesehen, wie Sie die durch den @scheduled_runtime -Parameter definierten Zeitbereiche verwenden können, um verspätet eingehende Daten zu berücksichtigen. Wenn Sie jedoch Anwendungsfälle haben, in denen sich Daten um Stunden oder Tage verzögern können, benötigen Sie möglicherweise ein anderes Muster, um sicherzustellen, dass Ihre Vorberechnungen in der abgeleiteten Tabelle entsprechend aktualisiert werden, um solche spät eintreffenden Daten widerzuspiegeln. Allgemeine Informationen zu spät eingehenden Daten finden Sie unter. Daten schreiben (Einfügungen und Upserts)

Im Folgenden finden Sie zwei verschiedene Möglichkeiten, mit diesen verspätet eingehenden Daten umzugehen.

- Wenn Sie vorhersehbare Verzögerungen bei der Ankunft Ihrer Daten haben, können Sie eine weitere geplante Berechnung verwenden, um Ihre Aggregate für verspätet eingehende Daten zu aktualisieren.
- Bei unvorhersehbaren Verzögerungen oder gelegentlich zu spät eintreffenden Daten können Sie manuelle Ausführungen verwenden, um die abgeleiteten Tabellen zu aktualisieren.

In dieser Diskussion werden Szenarien für verspätete Dateneingänge behandelt. Die gleichen Prinzipien gelten jedoch für Datenkorrekturen, bei denen Sie die Daten in Ihrer Quelltabelle geändert haben und die Aggregate in Ihren abgeleiteten Tabellen aktualisieren möchten.

Themen

- Geplante Nachholanfragen
- Manuelle Ausführungen für unvorhersehbare spät eintreffende Daten

Geplante Nachholanfragen

Abfragen und aggregieren von Daten, die rechtzeitig eingetroffen sind

Im Folgenden finden Sie ein Muster, in dem Sie sehen, wie Sie Ihre Aggregate automatisiert aktualisieren können, wenn es zu vorhersehbaren Verzögerungen bei der Dateneingabe kommt.

Sehen Sie sich im Folgenden eines der vorherigen Beispiele für eine geplante Berechnung mit Echtzeitdaten an. Bei dieser geplanten Berechnung wird die abgeleitete Tabelle alle 30 Minuten aktualisiert, wobei bereits Daten mit einer Verzögerung von bis zu einer Stunde berücksichtigt werden.

```
{
    "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": "******"
}
```

Nachholabfrage zur Aktualisierung der Aggregate für verspätet eintreffende Daten

Wenn Sie nun den Fall in Betracht ziehen, dass sich Ihre Daten um etwa 12 Stunden verzögern können. Im Folgenden finden Sie eine Variante derselben Abfrage. Der Unterschied besteht jedoch darin, dass die Aggregate anhand von Daten berechnet werden, die im Vergleich zu dem Zeitpunkt, zu dem die geplante Berechnung ausgelöst wird, um bis zu 12 Stunden verzögert sind. Sie sehen beispielsweise die Abfrage im Beispiel unten. Der Zeitraum, auf den diese Abfrage abzielt, liegt zwischen 2 Stunden und 14 Stunden vor dem Auslösen der Abfrage. Außerdem, wenn Ihnen der Zeitplanausdruck cron (0, 0,12 *?) auffällt. *), wird die Berechnung täglich um 00:00 UTC und 12:00 UTC Uhr ausgelöst. Wenn die Abfrage also am 2021-12-01 00:00:00 ausgelöst wird, aktualisiert die Abfrage die Aggregate im Zeitraum 2021-11-30 10:00:00 bis 2021-11-30 22:00:00. Geplante Abfragen verwenden eine Upsert-Semantik ähnlich wie Timestream für LiveAnalytics Schreibvorgänge, bei der diese Nachholabfrage die Aggregatwerte mit neueren Werten aktualisiert, wenn spät eintreffende Daten im Fenster erscheinen oder wenn neuere Aggregate gefunden werden (z. B. wird eine neue Gruppierung in diesem Aggregat angezeigt, die nicht vorhanden war, als die ursprüngliche geplante Berechnung ausgelöst wurde), dann wird das neue Aggregat in die abgeleitete Tabelle eingefügt. Ähnlich verhält es sich, wenn die nächste Instanz am 01.12.2021 12:00:00 ausgelöst wird, dann aktualisiert diese Instanz Aggregate im Bereich 2021-11-30 22:00:00 bis 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": "******"
}
```

Dieses vorherige Beispiel dient zur Veranschaulichung der Annahme, dass Ihre verspätete Ankunft auf 12 Stunden begrenzt ist und es in Ordnung ist, die abgeleitete Tabelle alle 12 Stunden für Daten zu aktualisieren, die später als im Echtzeitfenster ankommen. Sie können dieses Muster so anpassen, dass Ihre abgeleitete Tabelle einmal pro Stunde aktualisiert wird, sodass Ihre abgeleitete Tabelle die verspäteten Daten früher wiedergibt. In ähnlicher Weise können Sie den Zeitraum so anpassen, dass er älter als 12 Stunden ist, z. B. einen Tag oder sogar eine Woche oder länger, um vorhersehbare spät eintreffende Daten verarbeiten zu können.

Manuelle Ausführungen für unvorhersehbare spät eintreffende Daten

Es kann vorkommen, dass unvorhersehbare spät eintreffende Daten vorliegen oder dass Sie Änderungen an den Quelldaten vorgenommen und einige Werte nachträglich aktualisiert haben. In all diesen Fällen können Sie manuell geplante Abfragen auslösen, um die abgeleitete Tabelle zu aktualisieren. Im Folgenden finden Sie ein Beispiel dafür, wie Sie dies erreichen können.

Gehen Sie davon aus, dass Sie den Anwendungsfall haben, bei dem die Berechnung in die abgeleitete Tabelle dp_per_timeseries_per_hr geschrieben wurde. Ihre Basisdaten in der Tabelle

devops wurden im Zeitraum 2021-11-30 23:00:00 - 2021-12-01 00:00:00 aktualisiert. Es gibt zwei verschiedene geplante Abfragen, mit denen diese abgeleitete Tabelle aktualisiert werden kann: Multi 0 und Multi. PT3 mPerHr PerTimeseries DPCount PT12HPerHrPerTimeseriesDPCountCatchUp Jede geplante Berechnung, für die Sie in Timestream erstellen, LiveAnalytics hat ein eindeutiges ARN Objekt, das Sie erhalten, wenn Sie die Berechnung erstellen oder wenn Sie eine Listenoperation ausführen. Sie können den ARN für die Berechnung und einen Wert für den Parameter @scheduled_runtime verwenden, der von der Abfrage verwendet wird, um diesen Vorgang auszuführen.

Gehen Sie davon aus, dass die Berechnung für Multi PT3 0 einen mPerHr PerTimeseries DPCount Wert vom Typ ARN arn_1 hat und Sie diese Berechnung verwenden möchten, um die abgeleitete Tabelle zu aktualisieren. Da die vorherige geplante Berechnung die Aggregate 1 Stunde vor und 1 Stunde nach dem @scheduled_runtime -Wert aktualisiert, können Sie den Zeitraum für die Aktualisierung (2021-11-30 23:00:00 - 2021-12-01 00:00:00) mit dem Wert 2021-12-01 00:00:00 für den @scheduled_runtime -Parameter abdecken. Um dies zu erreichen, können Sie den Wert dieser Berechnung und ExecuteScheduledQuery API den ARN Zeitparameterwert in Epochensekunden (in) übergeben. UTC Im Folgenden finden Sie ein Beispiel für die AWS CLI Verwendung von. Sie können dem gleichen Muster folgen, indem Sie eines der von Timestream SDKs unterstützten Tools für verwenden. LiveAnalytics

```
aws timestream-query execute-scheduled-query --scheduled-query-arn arn_1 --invocation-
time 1638316800 --profile profile --region us-east-1
```

Im vorherigen Beispiel ist Profil das Profil, das über die AWS entsprechenden Rechte verfügt, um diesen API Anruf zu tätigen, und 1638316800 entspricht der Epochensekunde für 2021-12-01 00:00:00. Dieser manuelle Trigger verhält sich fast wie der automatisierte Trigger, vorausgesetzt, das System hat diesen Aufruf im gewünschten Zeitraum ausgelöst.

Wenn Sie in einem längeren Zeitraum ein Update hatten, sagen wir, die Basisdaten wurden für 2021-11-30 23:00:00 - 2021-12-01 11:00:00 aktualisiert, dann können Sie die vorherigen Abfragen mehrmals auslösen, um den gesamten Zeitraum abzudecken. Sie könnten beispielsweise sechs verschiedene Ausführungen wie folgt durchführen.

```
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
```

Die vorherigen sechs Befehle entsprechen der geplanten Berechnung, die am 01.12.2021 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 und 2021-12-01 10:00 Uhr aufgerufen wurde:

Alternativ können Sie die Berechnung Multi verwenden, die am 01.12.2021 13:00:00 ausgelöst wurde, für eine Ausführung, um die Aggregate für den gesamten 12-Stunden-Zeitraum zu aktualisieren. PT12HPerHrPerTimeseriesDPCountCatchUp Wenn beispielsweise arn_2 ARN für diese Berechnung verwendet wird, können Sie den folgenden Befehl von ausführen. CLI

```
aws timestream-query execute-scheduled-query --scheduled-query-arn arn_2 --invocation-
time 1638363600 --profile profile --region us-east-1
```

Beachten Sie, dass Sie für einen manuellen Trigger einen Zeitstempel für den Aufrufzeitparameter verwenden können, der nicht mit den Zeitstempeln des automatisierten Triggers abgestimmt werden muss. Im vorherigen Beispiel haben Sie die Berechnung zum Zeitpunkt 2021-12-01 13:00:00 ausgelöst, obwohl der automatisierte Zeitplan nur bei den Zeitstempeln 2021-12-01 10:00:00, 2021-12-01 12:00:00 und 2021-12-02 00:00:00 ausgelöst wird. Timestream for bietet Ihnen die Flexibilität, es mit den entsprechenden Werten auszulösen, die für Ihre manuellen Operationen erforderlich LiveAnalytics sind.

Im Folgenden finden Sie einige wichtige Überlegungen zur Verwendung von. ExecuteScheduledQuery API

- Wenn Sie mehrere dieser Aufrufe auslösen, müssen Sie sicherstellen, dass diese Aufrufe in sich überschneidenden Zeiträumen nicht zu Ergebnissen führen. In den vorherigen Beispielen gab es beispielsweise sechs Aufrufe. Jeder Aufruf deckt einen Zeitraum von 2 Stunden ab, weshalb die Zeitstempel des Aufrufs jeweils um zwei Stunden verteilt wurden, um Überschneidungen bei den Aktualisierungen zu vermeiden. Dadurch wird sichergestellt, dass die Daten in der abgeleiteten

Tabelle in einem Zustand landen, in dem es sich um Aggregate aus der Quelltabelle handelt. Wenn Sie nicht sicherstellen können, dass sich die Zeitbereiche nicht überschneiden, stellen Sie sicher, dass diese Ausführungen nacheinander ausgelöst werden. Wenn Sie mehrere Ausführungen gleichzeitig auslösen, die sich in ihren Zeiträumen überschneiden, können in den Fehlerberichten für diese Ausführungen Triggerfehler auftreten, bei denen es zu Versionskonflikten kommen kann. Ergebnissen, die durch einen geplanten Abfrageaufruf generiert wurden, wird eine Version zugewiesen, die darauf basiert, wann der Aufruf ausgelöst wurde. Daher haben Zeilen, die durch neuere Aufrufe generiert wurden, höhere Versionen. Ein Datensatz mit einer höheren Version kann einen Datensatz mit einer niedrigeren Version überschreiben. Bei automatisch ausgelösten geplanten Abfragen verwaltet Timestream for die Zeitpläne LiveAnalytics automatisch, sodass diese Probleme auch dann nicht auftreten, wenn sich die Zeitbereiche der nachfolgenden Aufrufe überschneiden.

- Wie bereits erwähnt, können Sie die Aufrufe mit einem beliebigen Zeitstempelwert für @scheduled_runtime auslösen. Es liegt also in Ihrer Verantwortung, die Werte entsprechend festzulegen, sodass die entsprechenden Zeitbereiche in der abgeleiteten Tabelle entsprechend den Bereichen aktualisiert werden, in denen Daten in der Quelltabelle aktualisiert wurden.
- Sie können diese manuellen Trigger auch für geplante Abfragen verwenden, die sich im DISABLED Status befinden. Auf diese Weise können Sie spezielle Abfragen definieren, die nicht in einem automatisierten Zeitplan ausgeführt werden, da sie sich im DISABLED Status befinden. Stattdessen können Sie die manuellen Auslöser für sie verwenden, um Datenkorrekturen oder Anwendungsfälle mit verspäteter Ankunft zu verwalten.

Rückvervollständigung historischer Vorberechnungen

Wenn Sie eine geplante Berechnung erstellen, LiveAnalytics verwaltet Timestream for die Ausführung der Abfragen in Zukunft, wobei die Aktualisierung durch den von Ihnen angegebenen Zeitplanausdruck gesteuert wird. Je nachdem, wie viele historische Daten Ihre Quelltabelle enthält, möchten Sie möglicherweise Ihre abgeleitete Tabelle mit Aggregaten aktualisieren, die den historischen Daten entsprechen. Sie können die obige Logik für manuelle Trigger verwenden, um die historischen Aggregate wieder aufzufüllen.

Wenn wir beispielsweise die abgeleitete Tabelle per_timeseries_lastpoint_pt1d betrachten, wird die geplante Berechnung einmal täglich für den vergangenen Tag aktualisiert. Wenn Ihre Quelltabelle Daten für ein Jahr enthält, können Sie die ARN für diese geplante Berechnung verwenden und sie für jeden Tag, der bis zu einem Jahr alt ist, manuell auslösen, sodass in der abgeleiteten Tabelle alle historischen Abfragen aufgefüllt werden. Beachten Sie, dass hier alle Einschränkungen für manuelle Trigger gelten. Wenn die abgeleitete Tabelle so eingerichtet ist, dass die historische Aufnahme in

den Magnetspeicher der abgeleiteten Tabelle geschrieben wird, sollten Sie außerdem die bewährten Methoden und Beschränkungen für Schreibvorgänge in den Magnetspeicher beachten.

Beispiele für geplante Abfragen

Dieser Abschnitt enthält Beispiele dafür, wie Sie die geplanten Abfragen von Timestream for LiveAnalytics verwenden können, um die Kosten und die Ladezeiten des Dashboards zu optimieren, wenn Sie flottenweite Statistiken visualisieren und Ihre Geräteflotte effektiv überwachen möchten. Geplante Abfragen in Timestream for LiveAnalytics ermöglichen es Ihnen, Ihre Anfragen mithilfe der gesamten SQL Oberfläche von Timestream for zu beantworten. LiveAnalytics Ihre Abfrage kann eine oder mehrere Quelltabellen enthalten, Aggregationen oder jede andere in der SQL Sprache von Timestream for LiveAnalytics zulässige Abfrage ausführen und dann die Ergebnisse der Abfrage in einer anderen Zieltabelle in Timestream for speichern. LiveAnalytics

In diesem Abschnitt wird die Zieltabelle einer geplanten Abfrage als abgeleitete Tabelle bezeichnet.

Als Beispiel verwenden wir eine DevOps Anwendung, bei der Sie eine große Serverflotte überwachen, die in mehreren Bereitstellungen (z. B. Regionen, Zellen und Silos) und mehreren Microservices bereitgestellt werden, und Sie verfolgen die flottenweiten Statistiken mithilfe von Timestream for. LiveAnalytics Das Beispielschema, das wir verwenden werden, ist unter Beispielschema für geplante Abfragen beschrieben.

Die folgenden Szenarien werden beschrieben.

- So konvertieren Sie ein Dashboard, zeichnen aggregierte Statistiken aus den Rohdaten, für die Sie in Timestream aufnehmen, LiveAnalytics in eine geplante Abfrage auf und wie Sie dann Ihre vorberechneten Aggregate verwenden, um ein neues Dashboard mit aggregierten Statistiken zu erstellen.
- So kombinieren Sie geplante Abfragen, um eine aggregierte Ansicht zu erhalten, und die detaillierten Rohdaten, um detaillierte Informationen zu erhalten. Auf diese Weise können Sie die Rohdaten speichern und analysieren und gleichzeitig Ihre allgemeinen flottenweiten Abläufe mithilfe von geplanten Abfragen optimieren.
- So optimieren Sie die Kosten mithilfe von geplanten Abfragen, indem Sie herausfinden, welche Aggregate in mehreren Dashboards verwendet werden und dass dieselbe geplante Abfrage mehrere Bereiche in demselben oder mehreren Dashboards auffüllt.

Themen

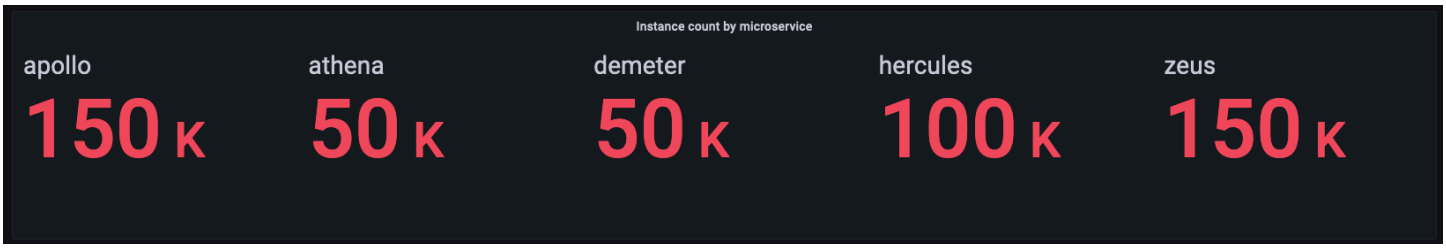
- Umwandeln eines aggregierten Dashboards in eine geplante Abfrage

- Verwendung von geplanten Abfragen und Rohdaten für Drilldowns
- Optimierung der Kosten durch gemeinsame Nutzung von geplanten Abfragen auf mehreren Dashboards
- Vergleich einer Abfrage in einer Basistabelle mit einer Abfrage von geplanten Abfrageergebnissen

Umwandeln eines aggregierten Dashboards in eine geplante Abfrage

Angenommen, Sie berechnen die flottenweiten Statistiken wie die Anzahl der Hosts in der Flotte nach den fünf Microservices und nach den sechs Regionen, in denen Ihr Service bereitgestellt wird. Aus dem folgenden Snapshot können Sie sehen, dass 500.000 Server Metriken ausgeben, und einige der größeren Regionen (z. B. us-east-1) haben >200.000 Server.

Die Berechnung dieser Aggregate, bei der Sie unterschiedliche Instanznamen für Hunderte von Gigabyte an Daten berechnen, kann zusätzlich zu den Kosten für das Scannen der Daten zu einer Abfragelatenz von mehreren zehn Sekunden führen.



Ursprüngliche Dashboard-Abfrage

Das im Dashboard-Bereich angezeigte Aggregat wird anhand der folgenden Abfrage aus Rohdaten berechnet. Die Abfrage verwendet mehrere SQL Konstrukte, z. B. unterschiedliche Zählungen und mehrere Aggregationsfunktionen.

```
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
)
```

Konvertierung in eine geplante Abfrage

Die vorherige Abfrage kann wie folgt in eine geplante Abfrage konvertiert werden. Sie berechnen zunächst die unterschiedlichen Hostnamen innerhalb einer bestimmten Bereitstellung in einer Region, einer Zelle, einem Silo, einer Availability Zone und einem Microservice. Anschließend addieren Sie die Hosts, um die Anzahl der Hosts pro Stunde und Microservice zu berechnen. Mithilfe des `@scheduled_runtime` Parameters, der von den geplanten Abfragen unterstützt wird, können Sie ihn für die letzte Stunde neu berechnen, wenn die Abfrage aufgerufen wurde. Die `bin(@scheduled_runtime, 1h)` `WHERE` In-the-Klausel der inneren Abfrage stellt sicher, dass Sie die Daten für die gesamte Stunde erhalten, auch wenn die Abfrage zu einem Zeitpunkt in der Mitte der Stunde geplant ist.

Die Abfrage berechnet zwar stündliche Aggregate, wie Sie in der Konfiguration für geplante Berechnungen sehen werden, ist jedoch so eingerichtet, dass sie jede halbe Stunde aktualisiert wird, sodass Sie Aktualisierungen in Ihrer abgeleiteten Tabelle früher erhalten. Sie können dies an Ihre Aktualitätsanforderungen anpassen, z. B. die Aggregate alle 15 Minuten neu berechnen oder sie an den Stundengrenzen neu berechnen.

```
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
```

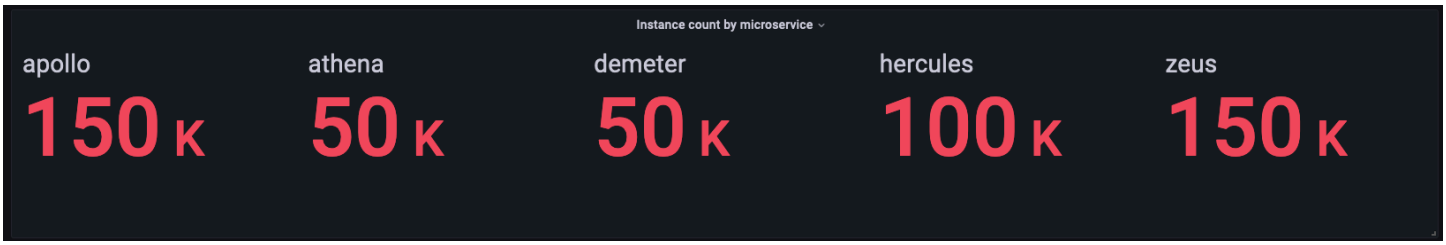
```
{
```

```
    "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": "******"
}
```

Verwendung der vorberechneten Ergebnisse in einem neuen Dashboard

Sie werden nun sehen, wie Sie Ihr Aggregatansichts-Dashboard mithilfe der abgeleiteten Tabelle aus der von Ihnen erstellten geplanten Abfrage erstellen. Anhand des Dashboard-Snapshots können Sie auch überprüfen, ob die aus der abgeleiteten Tabelle und der Basistabelle berechneten Aggregate ebenfalls übereinstimmen. Sobald Sie die Dashboards mithilfe der abgeleiteten Tabellen erstellt haben, werden Sie feststellen, dass die Verwendung der abgeleiteten Tabellen wesentlich kürzere Ladezeiten und geringere Kosten im Vergleich zur Berechnung dieser Aggregate aus den Rohdaten mit sich bringt. Im Folgenden finden Sie eine Momentaufnahme des Dashboards mit vorberechneten Daten und der Abfrage, die zum Rendern dieses Panels mit vorberechneten Daten verwendet wurde, die in der Tabelle „abgeleitet“ gespeichert sind.“ host_count_pt1h“. Beachten Sie, dass die Struktur der Abfrage der Abfrage, die im Dashboard für Rohdaten verwendet wurde, sehr ähnlich ist, mit der Ausnahme, dass sie die abgeleitete Tabelle verwendet, die bereits die einzelnen Zählungen berechnet, die diese Abfrage aggregiert.




```
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
)
```

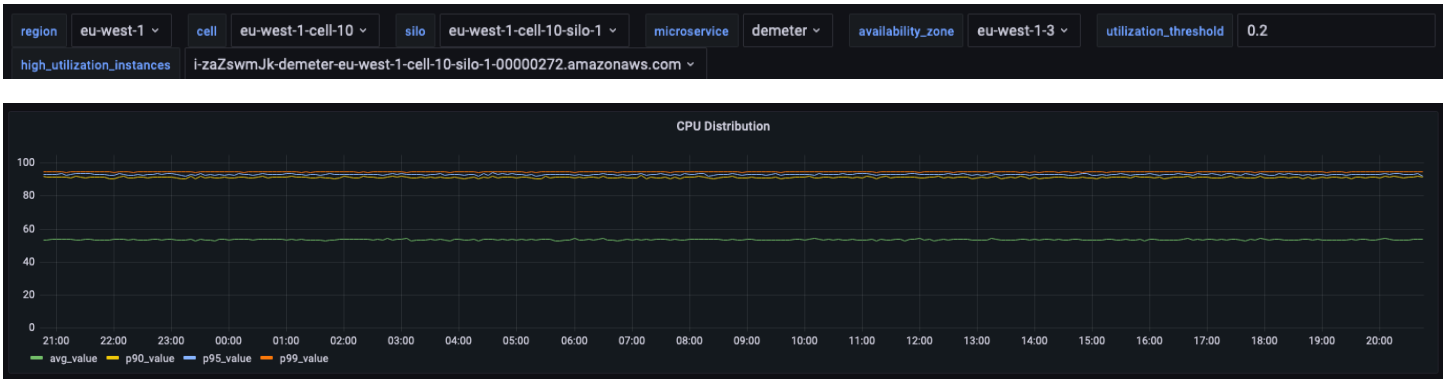
Verwendung von geplanten Abfragen und Rohdaten für Drilldowns

Sie können die aggregierten Statistiken für Ihre gesamte Flotte verwenden, um Bereiche zu identifizieren, in denen Drilldowns erforderlich sind, und dann anhand der Rohdaten detaillierte Daten analysieren, um tiefere Einblicke zu erhalten.

In diesem Beispiel sehen Sie, wie Sie mithilfe des aggregierten Dashboards jede Bereitstellung identifizieren können (eine Bereitstellung ist für einen bestimmten Microservice innerhalb einer bestimmten Region, Zelle, Silo und Availability Zone vorgesehen), die im Vergleich zu anderen Bereitstellungen anscheinend eine höhere CPU Auslastung aufweist. Anschließend können Sie anhand der Rohdaten weitere Informationen abrufen, um sich ein besseres Bild zu machen. Da diese Drilldowns möglicherweise selten sind und nur auf Daten zugreifen, die für die Bereitstellung relevant sind, können Sie die Rohdaten für diese Analyse verwenden und müssen keine geplanten Abfragen verwenden.

Drilldown pro Einsatz

Das unten stehende Dashboard bietet detaillierte Informationen zu detaillierteren Statistiken auf Serverebene innerhalb einer bestimmten Bereitstellung. Um Ihnen zu helfen, die verschiedenen Teile Ihrer Flotte genauer zu untersuchen, verwendet dieses Dashboard Variablen wie Region, Zelle, Silo, Microservice und Availability_Zone. Anschließend werden einige zusammengefasste Statistiken für diese Bereitstellung angezeigt.



In der folgenden Abfrage können Sie sehen, dass die in der Dropdownliste der Variablen ausgewählten Werte als Prädikate in der `WHERE` Klausel der Abfrage verwendet werden, sodass Sie sich nur auf die Daten für die Bereitstellung konzentrieren können. Anschließend zeichnet das Panel

die aggregierten CPU Metriken für Instanzen in dieser Bereitstellung auf. Sie können die Rohdaten verwenden, um diesen Drilldown mit interaktiver Abfragelatenz durchzuführen, um tiefere Einblicke zu gewinnen.

```
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
```

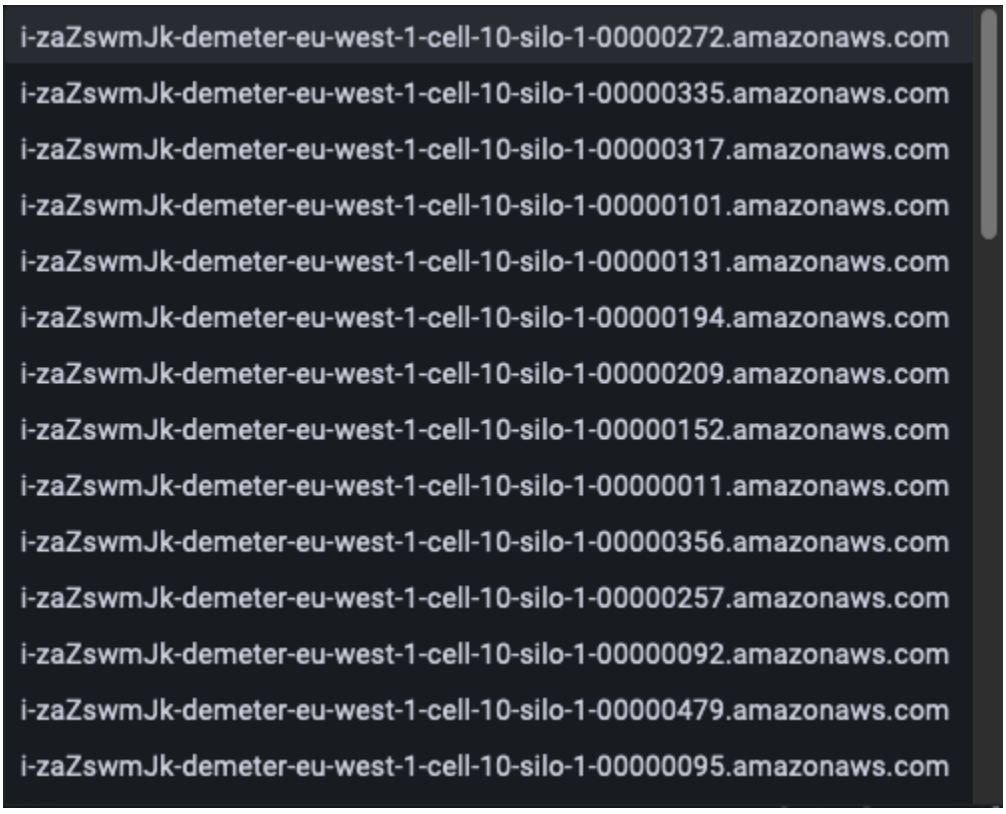
Statistiken auf Instanzebene

In diesem Dashboard wird außerdem eine weitere Variable berechnet, die auch die Server/Instances mit hoher CPU Auslastung auflistet, sortiert in absteigender Reihenfolge der Auslastung. Die zur Berechnung dieser Variablen verwendete Abfrage wird unten angezeigt.

```
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
```

In der vorherigen Abfrage wird die Variable abhängig von den für die anderen Variablen ausgewählten Werten dynamisch neu berechnet. Sobald die Variable für eine Bereitstellung aufgefüllt wurde, können Sie einzelne Instanzen aus der Liste auswählen, um die Metriken dieser Instanz weiter zu visualisieren. Sie können die verschiedenen Instanzen aus der Drop-down-Liste der Instanznamen auswählen, wie aus dem folgenden Snapshot ersichtlich ist.



In den vorherigen Bereichen werden die Statistiken für die ausgewählte Instanz angezeigt. Im Folgenden sind die Abfragen aufgeführt, die zum Abrufen dieser Statistiken verwendet wurden.

```
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
```

Optimierung der Kosten durch gemeinsame Nutzung von geplanten Abfragen auf mehreren Dashboards

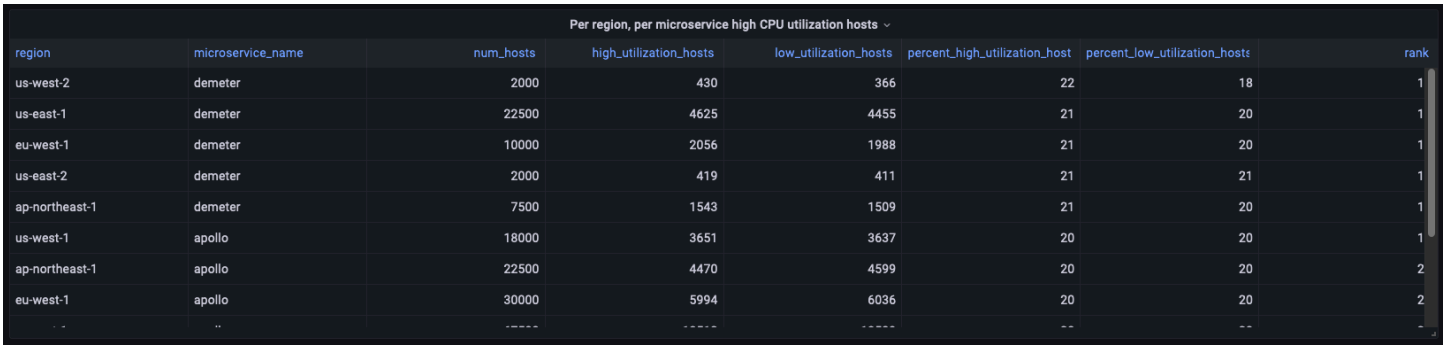
In diesem Beispiel sehen wir uns ein Szenario an, in dem in mehreren Dashboard-Panels Varianten ähnlicher Informationen angezeigt werden (wobei viele CPU Hosts und ein Teil der Flotte mit hoher CPU Auslastung gefunden werden) und wie Sie dieselbe geplante Abfrage verwenden können, um Ergebnisse vorab zu berechnen, die dann zum Füllen mehrerer Panels verwendet werden. Durch diese Wiederverwendung werden Ihre Kosten weiter optimiert, da Sie statt verschiedener geplanter Abfragen, eine für jedes Panel, nur den Eigentümer verwenden.

Dashboard-Panels mit Rohdaten

CPUNutzung pro Region pro Microservice

Das erste Panel berechnet die Instances, deren durchschnittliche CPU CPU Auslastung für eine bestimmte Bereitstellung innerhalb einer Region, einer Zelle, eines Silos, einer Availability Zone und eines Microservices einen Schwellenwert unter oder über der oben genannten Auslastung liegt. Anschließend werden die Region und der Microservice sortiert, der den höchsten Prozentsatz an Hosts mit hoher Auslastung aufweist. Es hilft dabei, festzustellen, wie heiß die Server einer bestimmten Bereitstellung laufen, und anschließend detaillierter zu ermitteln, um die Probleme besser zu verstehen.

Die Abfrage für das Panel zeigt die Flexibilität der SQL Unterstützung LiveAnalytics von Timestream for bei der Ausführung komplexer Analyseaufgaben mit gängigen Tabellenausdrücken, Fensterfunktionen, Verknüpfungen usw.



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

Abfrage:

```
WITH microservice_cell_avg AS (
```

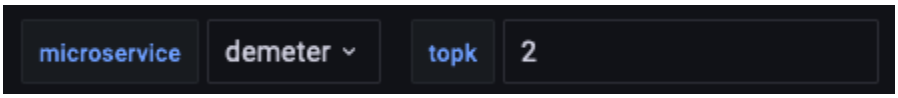
```
    SELECT region, cell, silo, availability_zone, microservice_name, AVG(cpu_user) AS
 microservice_avg_metric
    FROM "raw_data"."devops"
    WHERE time BETWEEN from_milliseconds(1636526593876) AND
 from_milliseconds(1636612993876)
        AND measure_name = 'metrics'
    GROUP BY region, cell, 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
```

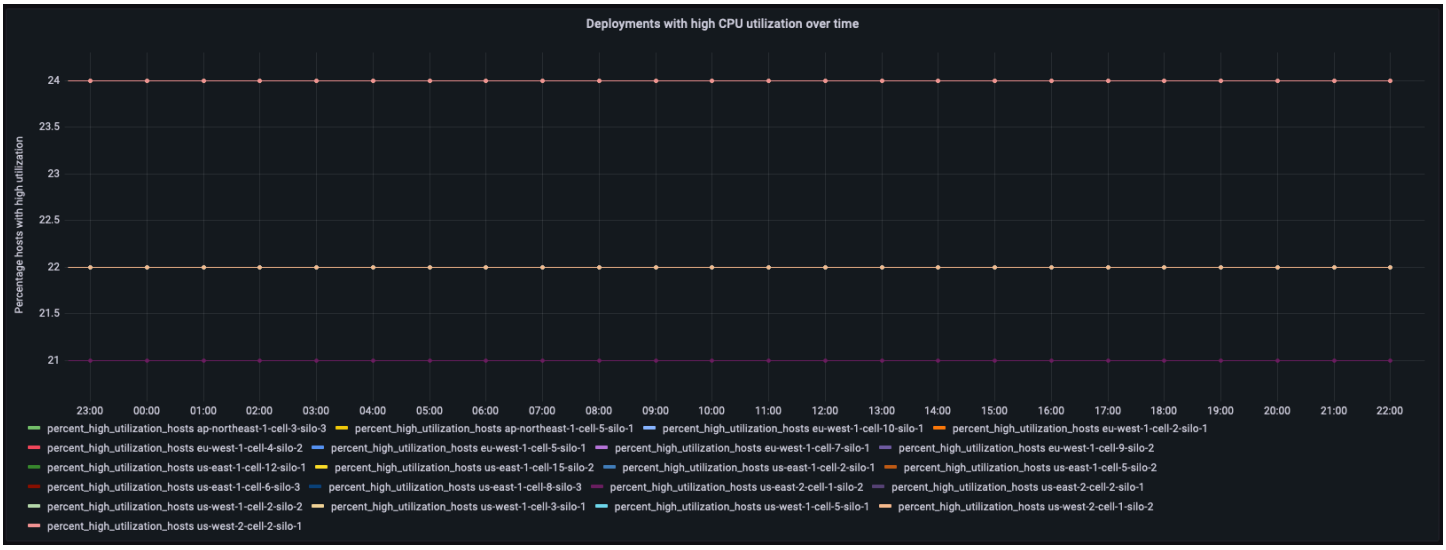
Sehen Sie sich einen Microservice genauer an, um Hotspots zu finden

Im nächsten Dashboard können Sie sich eingehender mit einem der Microservices befassen, um herauszufinden, in welcher Region, in welcher Zelle und in welchem Silo dieser Microservice welcher Bruchteil seiner Flotte am stärksten ausgelastet ist. CPU Im flottenweiten Dashboard haben Sie beispielsweise gesehen, dass der Microservice-Demeter unter den ersten paar Ranglisten auftauchte. In diesem Dashboard möchten Sie also tiefer in diesen Microservice eintauchen.

In diesem Dashboard wird anhand einer Variablen ein Microservice ausgewählt, in den ein Drilldown durchgeführt werden soll, und die Werte der Variablen werden anhand von Einzelwerten der Dimension aufgefüllt. Sobald Sie den Microservice ausgewählt haben, wird der Rest des Dashboards aktualisiert.

Wie Sie unten sehen, zeigt das erste Fenster den Prozentsatz der Hosts in einer Bereitstellung (eine Region, eine Zelle und ein Silo für einen Microservice) im Zeitverlauf sowie die entsprechende Abfrage, die zum Plotten des Dashboards verwendet wird. Dieses Diagramm selbst identifiziert eine bestimmte Bereitstellung mit einem höheren Prozentsatz an Hosts mit hohem Wert. CPU





Abfrage:

```
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
```

Konvertierung in eine einzige geplante Abfrage, die die Wiederverwendung ermöglicht

Es ist wichtig zu beachten, dass eine ähnliche Berechnung in den verschiedenen Bereichen der beiden Dashboards durchgeführt wird. Sie können für jedes Panel eine separate geplante Abfrage definieren. Hier erfahren Sie, wie Sie Ihre Kosten weiter optimieren können, indem Sie eine geplante Abfrage definieren, deren Ergebnisse zum Rendern aller drei Panels verwendet werden können.

Im Folgenden finden Sie die Abfrage, die die Aggregate erfasst, die für die verschiedenen Panels berechnet und verwendet werden. Bei der Definition dieser geplanten Abfrage werden Sie mehrere wichtige Aspekte beachten.

- Die Flexibilität und Leistungsfähigkeit der SQL Oberfläche wird durch geplante Abfragen unterstützt, bei denen Sie allgemeine Tabellenausdrücke, Verknüpfungen, Fallanweisungen usw. verwenden können.
- Sie können eine geplante Abfrage verwenden, um die Statistiken detaillierter zu berechnen, als es für ein bestimmtes Dashboard erforderlich wäre, und zwar für alle Werte, die ein Dashboard möglicherweise für verschiedene Variablen verwendet. Sie werden beispielsweise sehen, dass die Aggregate für eine Region, eine Zelle, ein Silo und einen Microservice berechnet werden. Daher können Sie diese kombinieren, um Aggregate auf Regions- oder Regions- und Microservice-Ebene zu erstellen. In ähnlicher Weise berechnet dieselbe Abfrage die Aggregate für alle Regionen, Zellen, Silos und Microservices. Sie können Filter auf diese Spalten anwenden, um die Aggregate für eine Teilmenge der Werte abzurufen. Sie können beispielsweise die Aggregate für eine beliebige Region berechnen, sagen wir us-east-1, oder einen beliebigen Microservice, sagen wir Demeter, oder Sie können eine spezifische Bereitstellung innerhalb einer Region, einer Zelle, eines Silos und eines Microservices aufschlüsseln. Durch diesen Ansatz werden Ihre Kosten für die Wartung der vorberechneten Aggregate weiter optimiert.

```
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
```

Im Folgenden finden Sie eine geplante Abfragedefinition für die vorherige Abfrage. Der Zeitplanausdruck ist so konfiguriert, dass er alle 30 Minuten aktualisiert wird. Er aktualisiert die Daten bis zu einer Stunde, wobei wiederum das Konstrukt bin (@scheduled_runtime, 1h) verwendet wird, um die Ereignisse für die gesamte Stunde abzurufen. Abhängig von den Aktualitätsanforderungen Ihrer Anwendung können Sie sie so konfigurieren, dass sie mehr oder weniger häufig aktualisiert wird. Durch die Verwendung von WHERE time BETWEEN bin (@scheduled_runtime, 1h) - 1h AND bin (@scheduled_runtime, 1h) + 1h können wir sicherstellen, dass Sie, auch wenn Sie einmal alle 15 Minuten aktualisieren, die vollen Stundendaten für die aktuelle Stunde und die vorherige Stunde erhalten.

Später werden Sie sehen, wie die drei Panels diese in die Tabelle deployment_cpu_stats_per_hr geschriebenen Aggregate verwenden, um die für das Panel relevanten Metriken zu visualisieren.

```
{
    "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"
               },
               {
```

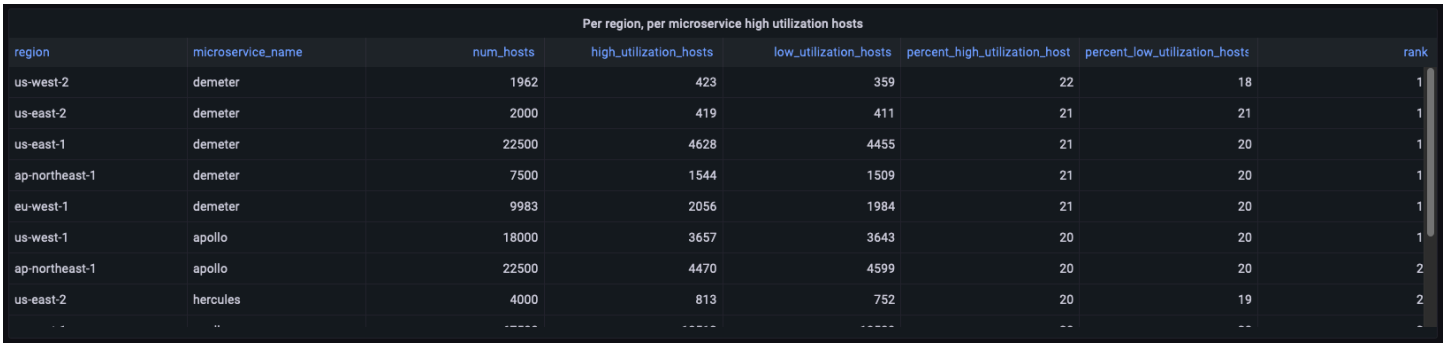
```
                    "Name": "silo",
                    "DimensionValueType": "VARCHAR"
                },
                {
                    "Name": "microservice_name",
                    "DimensionValueType": "VARCHAR"
                }
            ],
            "MultiMeasureMappings": {
                "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": "******"
}
```

Dashboard aus vorberechneten Ergebnissen

Hosts mit hoher Auslastung CPU

Bei Hosts mit hoher Auslastung werden Sie sehen, wie die verschiedenen Panels die Daten aus deployment_cpu_stats_per_hr verwenden, um verschiedene Aggregate zu berechnen, die für die

Panels erforderlich sind. Dieses Panel stellt beispielsweise Informationen auf regionaler Ebene bereit, sodass es Aggregate meldet, die nach Region und Microservice gruppiert sind, ohne dass Regionen oder Microservices gefiltert werden.



Per region, per microservice high utilization hosts

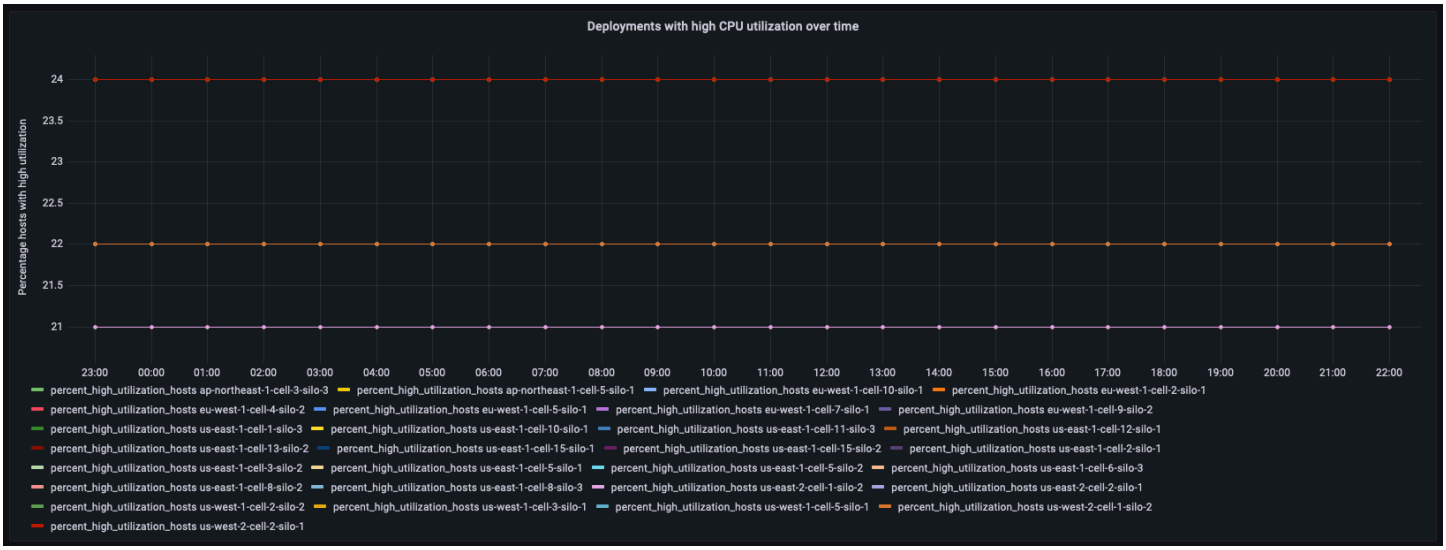
region	microservice_name	num_hosts	high_utilization_hosts	low_utilization_hosts	percent_high_utilization_host	percent_low_utilization_hosts	rank
us-west-2	demeter	1962	423	359	22	18	1
us-east-2	demeter	2000	419	411	21	21	1
us-east-1	demeter	22500	4628	4455	21	20	1
ap-northeast-1	demeter	7500	1544	1509	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
```

Sehen Sie sich einen Microservice genauer an, um Bereitstellungen mit hoher Auslastung zu finden CPU

Im nächsten Beispiel wird erneut die abgeleitete Tabelle deployment_cpu_stats_per_hr verwendet, aber jetzt wird ein Filter für einen bestimmten Microservice angewendet (in diesem Beispiel Demeter, da er Hosts mit hoher Auslastung im Aggregat-Dashboard gemeldet hat). In diesem Bereich wird der Prozentsatz der Hosts mit hoher Auslastung im Zeitverlauf nachverfolgt. CPU




```
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
```

Vergleich einer Abfrage in einer Basistabelle mit einer Abfrage von geplanten Abfrageergebnissen

In diesem Beispiel für eine Timestream-Abfrage verwenden wir das folgende Schema, Beispielabfragen und Ausgaben, um eine Abfrage in einer Basistabelle mit einer Abfrage in einer abgeleiteten Tabelle mit geplanten Abfrageergebnissen zu vergleichen. Mit einer gut geplanten geplanten Abfrage können Sie eine abgeleitete Tabelle mit weniger Zeilen und anderen Merkmalen erhalten, was zu schnelleren Abfragen führen kann, als dies mit der ursprünglichen Basistabelle möglich wäre.

Ein Video, das dieses Szenario beschreibt, finden Sie unter Verbessern der Abfrageleistung und Senkung der Kosten mithilfe von geplanten Abfragen in Amazon Timestream für LiveAnalytics.

Für dieses Beispiel verwenden wir das folgende Szenario:

- Region — us-east-1
- Basistabelle — `"clickstream"."shopping"`
- Abgeleitete Tabelle — `"clickstream"."aggregate"`

Basistabelle

Im Folgenden wird das Schema für die Basistabelle beschrieben.

Spalte	Typ	Timestream für den LiveAnalytics Attributtyp
channel	varchar	MULTI
description	varchar	MULTI
event	varchar	DIMENSION

Spalte	Typ	Timestream für den LiveAnalytics Attributtyp
ip_address	varchar	DIMENSION
measure_name	varchar	MEASURE_NAME
Produkt	varchar	MULTI
product_id	varchar	MULTI
quantity	double	MULTI
query	varchar	MULTI
session_id	varchar	DIMENSION
Benutzergruppe	varchar	DIMENSION
user_id	varchar	DIMENSION

Im Folgenden werden die Kennzahlen für die Basistabelle beschrieben. Eine Basistabelle bezieht sich auf eine Tabelle in Timestream, für die eine geplante Abfrage ausgeführt wird.

- Measure_name — `metrics`
- Daten — mehrere
- Abmessungen:

```
[ ( user_group, varchar ),( user_id, varchar ),( session_id, varchar ),( ip_address,
 varchar ),( event, varchar ) ]
```

Abfrage auf einer Basistabelle

Im Folgenden finden Sie eine Ad-hoc-Abfrage, bei der die Anzahl in einem bestimmten Zeitraum als 5-Minuten-Aggregat erfasst wird.

```
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
```

Ausgabe:

```
duration:1.745 sec
Bytes scanned: 29.89 MB
Query Id: AEBQEANMHG7MHHBHCKJ3BSOE3QUGIDBGWCCP5I6J6YUW5CVJZ2M3JCJ27QRMM7A
Row count:5
```

Geplante Abfrage

Die folgende ist eine geplante Abfrage, die alle 5 Minuten ausgeführt wird.

```
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
```

Abfrage einer abgeleiteten Tabelle

Das Folgende ist eine Ad-hoc-Abfrage für eine abgeleitete Tabelle. Eine abgeleitete Tabelle bezieht sich auf eine Timestream-Tabelle, die die Ergebnisse einer geplanten Abfrage enthält.

```
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'
```

Ausgabe:

```
duration: 0.2960 sec
Bytes scanned: 235.00 B
```

```
QueryID: AEBQEANMHHAAQU4FFTT6CFM6UYXTL4SMLZV22MFP4KV2Z7IRVOPLOMLDD6BR33Q
Row count: 5
```

Vergleich

Im Folgenden werden die Ergebnisse einer Abfrage in einer Basistabelle mit einer Abfrage in einer abgeleiteten Tabelle verglichen. Dieselbe Abfrage in einer abgeleiteten Tabelle, deren Ergebnisse im Rahmen einer geplanten Abfrage aggregiert wurden, wird mit weniger gescannten Byte schneller abgeschlossen.

Diese Ergebnisse zeigen, wie nützlich es ist, geplante Abfragen zu verwenden, um Daten für schnellere Abfragen zu aggregieren.

	Abfrage anhand der Basistabelle	Abfrage in einer abgeleiteten Tabelle
Dauer	1.745 Sekunden	0,2960 Sek
Gescannte Byte	29,89 MB	235 Byte
Anzahl der Zeilen	5	5

Wird verwendetUNLOAD, um Abfrageergebnisse von Timestream nach S3 zu exportieren für LiveAnalytics

Amazon Timestream ermöglicht es Ihnen LiveAnalytics vorerst, Ihre Abfrageergebnisse mithilfe des `UNLOAD` Kontoauszugs kostengünstig und sicher nach Amazon S3 zu exportieren. Mithilfe der `UNLOAD` Anweisung können Sie nun Zeitreihendaten in ausgewählte S3-Buckets entweder im Format Apache Parquet oder Comma Separated Values (CSV) exportieren. Dies bietet Flexibilität beim Speichern, Kombinieren und Analysieren Ihrer Zeitreihendaten mit anderen Services. Die `UNLOAD` Anweisung ermöglicht es Ihnen, die Daten komprimiert zu exportieren, wodurch die übertragenen Daten und der benötigte Speicherplatz reduziert werden. `UNLOAD`unterstützt auch die Partitionierung auf der Grundlage ausgewählter Attribute beim Exportieren der Daten, wodurch die Leistung verbessert und die Verarbeitungszeit nachgelagerter Dienste, die auf die Daten zugreifen, reduziert wird. Darüber hinaus können Sie verwaltete Amazon S3 S3-Schlüssel (SSE-S3) oder AWS Key Management Service () verwaltete Schlüssel (SSE-AWS KMSKMS) verwenden, um Ihre exportierten Daten zu verschlüsseln.

Vorteile von UNLOAD von Timestream für LiveAnalytics

Die wichtigsten Vorteile der Verwendung der `UNLOAD` Anweisung sind wie folgt.

- Einfache Bedienung — Mit der `UNLOAD` Anweisung können Sie Gigabytes an Daten in einer einzigen Abfrageanforderung entweder im Apache Parquet- oder im CSV Format exportieren. Dies bietet Flexibilität bei der Auswahl des für Ihre nachgelagerten Verarbeitungsanforderungen am besten geeigneten Formats und erleichtert die Erstellung von Data Lakes.
- Sicher und kostengünstig — `UNLOAD` Statement bietet die Möglichkeit, Ihre Daten komprimiert in einen S3-Bucket zu exportieren und Ihre Daten mit vom Kunden verwalteten Schlüsseln zu verschlüsseln (SSE- KMS oder SSE _S3), wodurch die Datenspeicherkosten gesenkt und vor unbefugtem Zugriff geschützt werden.
- Leistung — Mithilfe der `UNLOAD` Anweisung können Sie die Daten partitionieren, wenn Sie sie in einen S3-Bucket exportieren. Durch die Partitionierung der Daten können nachgelagerte Dienste die Daten parallel verarbeiten, wodurch ihre Verarbeitungszeit reduziert wird. Darüber hinaus können nachgelagerte Dienste nur die Daten verarbeiten, die sie benötigen, wodurch die erforderlichen Verarbeitungsressourcen und damit die damit verbundenen Kosten reduziert werden.

Anwendungsfälle für UNLOAD von Timestream für LiveAnalytics

Sie können die `UNLOAD` Anweisung verwenden, um Daten wie folgt in Ihren S3-Bucket zu schreiben.

- Data Warehouse erstellen — Sie können Gigabytes an Abfrageergebnissen in den S3-Bucket exportieren und Ihrem Data Lake einfacher Zeitreihendaten hinzufügen. Sie können Dienste wie Amazon Athena und Amazon Redshift verwenden, um Ihre Zeitreihendaten mit anderen relevanten Daten zu kombinieren, um komplexe Geschäftseinblicke abzuleiten.
- KI- und ML-Daten-Pipelines erstellen — Die `UNLOAD` Erklärung ermöglicht es Ihnen, auf einfache Weise Daten-Pipelines für Ihre Machine-Learning-Modelle zu erstellen, die auf Zeitreihendaten zugreifen, was die Verwendung von Zeitreihendaten mit Diensten wie Amazon SageMaker und Amazon erleichtert. EMR
- Vereinfachen Sie die ETL Verarbeitung — Der Export von Daten in S3-Buckets kann die Ausführung von Extraktions-, Transformations- und Load (ETL) -Vorgängen für die Daten vereinfachen, sodass Sie Tools oder AWS Dienste von Drittanbietern wie AWS Glue nahtlos zur Verarbeitung und Transformation der Daten verwenden können.

UNLOADKonzepte

Syntax

```
UNLOAD (SELECT statement)
 TO 's3://bucket-name/folder'
 WITH ( option = expression [, ...] )
```

wo `option` ist

```
{ 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

SELECTAussage

Die Abfrageanweisung, die verwendet wird, um Daten aus einem oder mehreren Timestreams für LiveAnalytics Tabellen auszuwählen und abzurufen.

```
(SELECT column 1, column 2, column 3 from database.table
      where measure_name = "ABC" and timestamp between ago (1d) and now() )
```

TO-Klausel

```
TO 's3://bucket-name/folder'
```

or

```
TO 's3://access-point-alias/folder'
```

Die TO Klausel in der UNLOAD Anweisung gibt das Ziel für die Ausgabe der Abfrageergebnisse an. Sie müssen den vollständigen Pfad angeben, einschließlich entweder des Amazon S3-Bucket-Namens oder Amazon S3 access-point-alias mit dem Speicherort des Ordners auf Amazon S3, in den Timestream for die LiveAnalytics Ausgabedateiobjekte schreibt. Der S3-Bucket sollte demselben Konto gehören und sich in derselben Region befinden. Zusätzlich zum Abfrageergebnissatz LiveAnalytics schreibt Timestream for die Manifest- und Metadatendateien in den angegebenen Zielordner.

PARTITIONED_BY-Klausel

```
partitioned_by = ARRAY [col_name[,…] , (default: none)
```

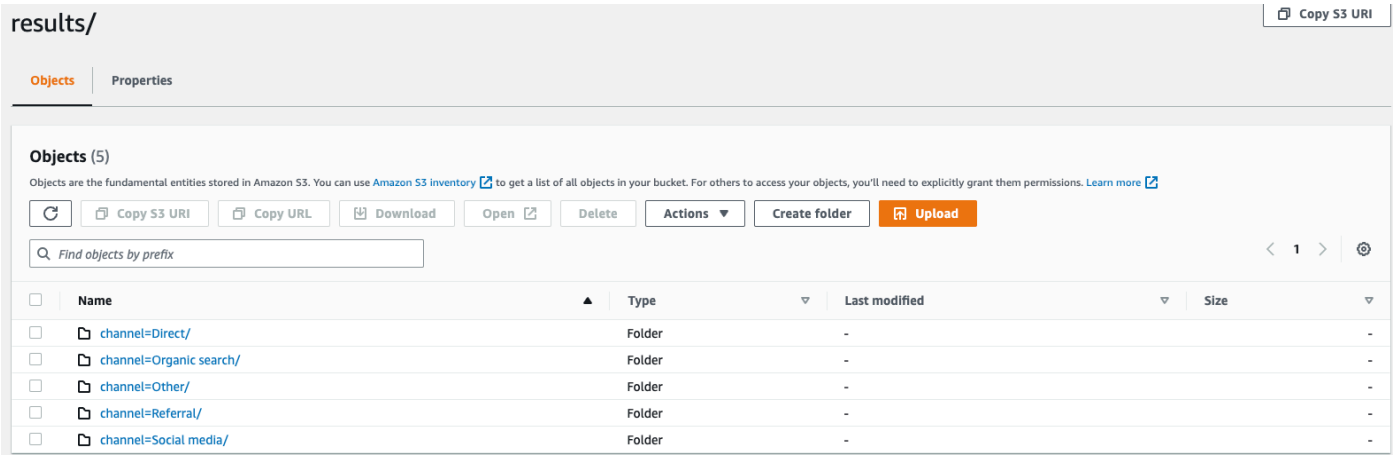
Die `partitioned_by` Klausel wird in Abfragen verwendet, um Daten auf granularer Ebene zu gruppieren und zu analysieren. Wenn Sie Ihre Abfrageergebnisse in den S3-Bucket exportieren, können Sie wählen, ob Sie die Daten auf der Grundlage einer oder mehrerer Spalten in der Auswahlabfrage partitionieren möchten. Bei der Partitionierung der Daten werden die exportierten Daten basierend auf der Partitionsspalte in Teilmengen unterteilt, und jede Teilmenge wird in einem separaten Ordner gespeichert. Innerhalb des Ergebnisordners, der Ihre exportierten Daten enthält, wird automatisch ein Unterordner erstellt`folder/results/partition column = partition value/`. Beachten Sie jedoch, dass partitionierte Spalten nicht in der Ausgabedatei enthalten sind.

`partitioned_by`ist keine obligatorische Klausel in der Syntax. Wenn Sie die Daten ohne Partitionierung exportieren möchten, können Sie die Klausel in der Syntax ausschließen.

Example

Angenommen, Sie überwachen die Clickstream-Daten Ihrer Website und haben 5 Verkehrskanäle`direct`, nämlich, `Social Media` `Organic Search``Other`, und. `Referral` Beim Exportieren der Daten können Sie wählen, ob Sie die Daten mithilfe der Spalte `Channel` partitionieren möchten. In Ihrem Datenordner befinden sich fünf Ordner mit jeweils ihrem jeweiligen Kanalnamen. `s3://bucketname/results/channel=Social Media/.` In diesem Ordner finden Sie beispielsweise die Daten aller Kunden, die über den `Social Media` Kanal auf Ihre Website gelangt sind. `s3://bucketname/results` In ähnlicher Weise werden Sie andere Ordner für die verbleibenden Kanäle haben.

Exportierte Daten, partitioniert nach Kanalspalten



FORMAT

```
format = [ '{ CSV | PARQUET }' , default: CSV
```

Die Schlüsselwörter zur Angabe des Formats der Abfrageergebnisse, die in Ihren S3-Bucket geschrieben werden. Sie können die Daten entweder als kommagetrennten Wert (CSV) mit einem Komma (,) als Standardtrennzeichen oder im Apache Parquet-Format, einem effizienten offenen spaltenbasierten Speicherformat für Analysen, exportieren.

COMPRESSION

```
compression = [ '{ GZIP | NONE }' ], default: GZIP
```

Sie können die exportierten Daten mithilfe des Komprimierungsalgorithmus komprimieren GZIP oder sie dekomprimieren lassen, indem Sie die Option angeben. `NONE`

ENCRYPTION

```
encryption = [ '{ SSE_KMS | SSE_S3 }' ], default: SSE_S3
```

Die Ausgabedateien auf Amazon S3 werden mit der von Ihnen ausgewählten Verschlüsselungsoption verschlüsselt. Zusätzlich zu Ihren Daten werden auch die Manifest- und Metadatendateien auf der Grundlage der von Ihnen ausgewählten Verschlüsselungsoption verschlüsselt. Wir unterstützen derzeit die SSE _S3- und SSE KMS _-Verschlüsselung. SSE_S3 ist eine serverseitige Verschlüsselung, bei der Amazon S3 die Daten mit der 256-Bit-Verschlüsselung des Advanced Encryption Standard () verschlüsselt. AES SSE_ KMS ist eine serverseitige Verschlüsselung zur Verschlüsselung von Daten mit vom Kunden verwalteten Schlüsseln.

KMS_KEY

```
kms_key = '<string>'
```

KMSDer Schlüssel ist ein vom Kunden definierter Schlüssel zur Verschlüsselung exportierter Abfrageergebnisse. KMSDer Schlüssel wird sicher vom AWS Key Management Service (AWS KMS) verwaltet und zum Verschlüsseln von Datendateien auf Amazon S3 verwendet.

FIELD_DELIMITER

```
field_delimiter ='<character>' , default: (,)
```

Beim Exportieren der Daten im CSV Format gibt dieses Feld ein einzelnes ASCII Zeichen an, das zur Trennung von Feldern in der Ausgabedatei verwendet wird, z. B. ein senkrechter Strich (|), ein Komma (,) oder ein Tabulatorzeichen (/t). Das Standardtrennzeichen für CSV Dateien ist ein Komma. Wenn ein Wert in Ihren Daten das gewählte Trennzeichen enthält, wird das Trennzeichen mit einem Anführungszeichen in Anführungszeichen gesetzt. Wenn der Wert in Ihren Daten beispielsweise Folgendes enthält`Time,stream`, wird dieser Wert wie in den exportierten Daten `"Time,stream"` in Anführungszeichen gesetzt. Das von Timestream für verwendete Anführungszeichen LiveAnalytics sind doppelte Anführungszeichen („).

Vermeiden Sie es, das Wagenrücklaufzeichen (ASCII`130D`, Hexadezimalzahl, Text '\ r') oder das Zeilenumbruchzeichen (ASCII10, Hexadezimalzahl 0A, Text '\n') als 'anzugeben, `FIELD_DELIMITER` wenn Sie Header in die aufnehmen möchtenCSV, da dies viele Parser daran hindert, die Header in der resultierenden Ausgabe korrekt zu analysieren. CSV

ESCAPED_VON

```
escaped_by = '<character>', default: (\)
```

Beim Exportieren der Daten im CSV Format gibt dieses Feld das Zeichen an, das in der in den S3-Bucket geschriebenen Datendatei als Escape-Zeichen behandelt werden soll. Escaping findet in den folgenden Szenarien statt:

1. Wenn der Wert selbst das Anführungszeichen („) enthält, wird er mit einem Escape-Zeichen maskiert. Wenn der Wert beispielsweise lautet`Time"stream`, wobei (\) das konfigurierte Escape-Zeichen ist, dann wird er als maskiert`Time\"stream`.
2. Wenn der Wert das konfigurierte Escape-Zeichen enthält, wird es maskiert. Wenn der Wert beispielsweise ist`Time\stream`, wird er als maskiert`Time\\stream`.

> **Note**
>
> Wenn die exportierte Ausgabe komplexe Datentypen wie Arrays, Zeilen oder Zeitreihen enthält, wird sie als Zeichenfolge serialisiert. JSON Im Folgenden sehen Sie ein Beispiel.

Datentyp	Tatsächlicher Wert	Wie der Wert im CSV Format [serialisierte JSON Zeichenfolge] maskiert wird
Array	`[ 23,24,25 ]`	`"[23,24,25]"`
Zeile	`( x=23.0, y=hello )`	`"{\"x\":23.0,\"y\":\"hello\"}"`
Zeitreihen	`[ ( 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'
```

Wenn Sie die Daten im CSV Format exportieren, können Sie in diesem Feld Spaltennamen als erste Zeile der exportierten CSV Datendateien angeben.

Die akzeptierten Werte sind „wahr“ und „falsch“ und der Standardwert ist „falsch“. Optionen zur Texttransformation wie `escaped_by` und `field_delimiter` gelten auch für Überschriften.

> **Note**
>
> Beim Einbeziehen von Kopfzeilen ist es wichtig, dass Sie kein Wagenrücklaufzeichen (ASCII13, Hex 0D, Text '\ r') oder ein Zeilenumbruchzeichen (ASCII10, Hex 0A, Text '\n') als 'auswählen`FIELD_DELIMITER`, da dies viele Parser daran hindert, die Header in der resultierenden Ausgabe korrekt zu analysieren. CSV

MAX_FILE_SIZE

```
max_file_size = 'X[MB|GB]' , default: '78GB'
```

Dieses Feld gibt die maximale Größe der Dateien an, die die `UNLOAD` Anweisung in Amazon S3 erstellt. Die `UNLOAD` Anweisung kann mehrere Dateien erstellen, aber die maximale Größe jeder in Amazon S3 geschriebenen Datei entspricht ungefähr der in diesem Feld angegebenen Größe.

Der Wert des Felds muss zwischen 16 MB und einschließlich 78 GB liegen. Sie können ihn als Ganzzahl wie `12GB` oder in Dezimalzahlen wie `0.5GB` oder angeben. `24.7MB` Der Standardwert ist 78 GB.

Die tatsächliche Dateigröße wird beim Schreiben der Datei geschätzt, sodass die tatsächliche Maximalgröße möglicherweise nicht genau der von Ihnen angegebenen Zahl entspricht.

Was wird in meinen S3-Bucket geschrieben?

Für jede erfolgreich ausgeführte UNLOAD Abfrage LiveAnalytics schreibt Timestream for Ihre Abfrageergebnisse, die Metadatendatei und die Manifestdatei in den S3-Bucket. Wenn Sie die Daten partitioniert haben, befinden sich alle Partitionsordner im Ergebnisordner. Die Manifestdatei enthält eine Liste der Dateien, die mit dem UNLOAD Befehl geschrieben wurden. Die Metadatendatei enthält Informationen, die die Merkmale, Eigenschaften und Attribute der geschriebenen Daten beschreiben.

Wie lautet der Name der exportierten Datei?

Der Name der exportierten Datei enthält zwei Komponenten, die erste Komponente ist die QueryID und die zweite Komponente ist eine eindeutige Kennung.

CSVDateien

```
S3://bucket_name/results/<queryid>_<UUID>.csv
```

```
S3://bucket_name/results/<partitioncolumn>=<partitionvalue>/<queryid>_<UUID>.csv
```

Komprimierte CSV Datei

```
S3://bucket_name/results/<partitioncolumn>=<partitionvalue>/<queryid>_<UUID>.gz
```

Parquet-Datei

```
S3://bucket_name/results/<partitioncolumn>=<partitionvalue>/<queryid>_<UUID>.parquet
```

Metadaten und Manifestdateien

```
S3://bucket_name/<queryid>_<UUID>_manifest.json
S3://bucket_name/<queryid>_<UUID>_metadata.json
```

Da die Daten im CSV Format auf Dateiebene gespeichert werden, hat die Datei beim Komprimieren der Daten beim Exportieren nach S3 die Erweiterung „.gz“. Die Daten in Parquet werden jedoch auf Spaltenebene komprimiert. Selbst wenn Sie die Daten beim Exportieren komprimieren, hat die Datei immer noch die Erweiterung.parquet.

Welche Informationen enthält jede Datei?

Manifestdatei

Die Manifestdatei enthält Informationen zur Liste der Dateien, die bei der UNLOAD Ausführung exportiert werden. Die Manifestdatei ist im bereitgestellten S3-Bucket mit einem Dateinamen verfügbar:`s3://<bucket_name>/<queryid>_<UUID>_manifest.json`. Die Manifestdatei enthält die URL der Dateien im Ergebnisordner, die Anzahl der Datensätze und die Größe der jeweiligen Dateien sowie die Abfrage-Metadaten (d. h. die Gesamtzahl der Byte und der Gesamtzahl der Zeilen, die für die Abfrage nach S3 exportiert wurden).

```
{
  "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"
  }
}
```

Metadaten

Die Metadatendatei enthält zusätzliche Informationen über den Datensatz, z. B. Spaltenname, Spaltentyp und Schema. <queryid>Die Metadatendatei ist im bereitgestellten S3-Bucket mit dem folgenden Dateinamen verfügbar: S3: //bucket_name/ _< >_metadata.json UUID

Es folgt ein Beispiel für eine Metadatendatei.

```
{
    "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"
  }
}
```

Die in der Metadatendatei gemeinsam genutzten Spalteninformationen haben dieselbe Struktur wie die in der API Abfrageantwort für `SELECT` Abfragen `ColumnInfo` gesendeten Informationen.

Ergebnisse

Der Ergebnisordner enthält Ihre exportierten Daten entweder im Apache Parquet- oder im Apache CSV Parquet-Format.

Beispiel

Wenn Sie eine `UNLOAD` Anfrage wie unten über Query einreichenAPI,

```
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')
```

UNL0ADDie Abfrageantwort wird 1 Zeile x 3 Spalten haben. Diese 3 Spalten sind:

- Zeilen des Typs BIGINT — gibt die Anzahl der exportierten Zeilen an
- metadataFile vom Typ VARCHAR — das ist das S3 URI der exportierten Metadatendatei
- manifestFile vom Typ VARCHAR — das ist der S3 URI der exportierten Manifest-Datei

Sie erhalten die folgende Antwort von 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
    }
}
```

Datentypen

Die UNLOAD Anweisung unterstützt alle Datentypen der Abfragesprache von Timestream for LiveAnalytics, die unter beschrieben sind, Unterstützte Datentypen außer `time` undunknown.

Voraussetzungen für UNLOAD von Timestream für LiveAnalytics

Im Folgenden finden Sie die Voraussetzungen für das Schreiben von Daten in S3 mithilfe UNLOAD von Timestream for. LiveAnalytics

- Sie müssen berechtigt sein, Daten aus dem Timestream zu lesen, damit LiveAnalytics Tabellen in einem UNLOAD Befehl verwendet werden können.
- Sie müssen einen Amazon S3 S3-Bucket in derselben AWS Region wie Ihr Timestream für LiveAnalytics Ressourcen haben.
- Stellen Sie für den ausgewählten S3-Bucket sicher, dass die S3-Bucket-Richtlinie auch über Berechtigungen verfügt, damit Timestream LiveAnalytics die Daten exportieren kann.
- Die für die Ausführung der UNLOAD Abfrage verwendeten Anmeldeinformationen müssen über die erforderlichen AWS Identity and Access Management Zugriffsverwaltungsberechtigungen (IAM) verfügen, die es Timestream for ermöglichen, die Daten in S3 LiveAnalytics zu schreiben. Ein Beispiel für eine Richtlinie wäre wie folgt:

```
{
```

```
    "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>/*"
            ]
        }
    ]
}
```

Weitere Informationen zu diesen S3-Schreibberechtigungen finden Sie im Amazon Simple Storage Service-Handbuch. Wenn Sie einen KMS Schlüssel zum Verschlüsseln der exportierten Daten verwenden, finden Sie im Folgenden die zusätzlichen erforderlichen IAM Richtlinien.

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

Bewährte Methoden für UNLOAD von Timestream für LiveAnalytics

Im Folgenden finden Sie bewährte Methoden für den UNLOAD Befehl.

- Die Datenmenge, die mit dem `UNLOAD` Befehl in den S3-Bucket exportiert werden kann, ist unbegrenzt. Bei der Abfrage kommt es jedoch nach 60 Minuten zu einem Timeout, und wir empfehlen, nicht mehr als 60 GB an Daten in einer einzigen Abfrage zu exportieren. Wenn Sie mehr als 60 GB Daten exportieren müssen, teilen Sie den Job auf mehrere Abfragen auf.
- Sie können zwar Tausende von Anfragen an S3 senden, um die Daten hochzuladen, es wird jedoch empfohlen, die Schreibvorgänge auf mehrere S3-Präfixe zu parallelisieren. Die Dokumentation finden Sie hier. Die API S3-Anrufrate könnte gedrosselt werden, wenn mehrere Leser/Schreiber auf denselben Ordner zugreifen.

- Angesichts der Beschränkung der S3-Schlüssellänge für die Definition eines Präfixes empfehlen wir, Bucket- und Ordnernamen innerhalb von 10 bis 15 Zeichen zu verwenden, insbesondere bei Verwendung von Klauseln. `partitioned_by`
- Wenn Sie für Abfragen, die die `UNLOAD` Anweisung enthalten, ein 4XX- oder 5XX-Zeichen erhalten, ist es möglich, dass Teilergebnisse in den S3-Bucket geschrieben werden. Timestream for löscht LiveAnalytics keine Daten aus Ihrem Bucket. Bevor Sie eine weitere `UNLOAD` Abfrage mit demselben S3-Ziel ausführen, empfehlen wir, die durch die fehlgeschlagene Abfrage erstellten Dateien manuell zu löschen. Sie können die Dateien, die durch eine fehlgeschlagene Abfrage geschrieben wurden, anhand der entsprechenden Informationen identifizieren`QueryExecutionId`. Bei fehlgeschlagenen Abfragen exportiert Timestream for LiveAnalytics keine Manifestdatei in den S3-Bucket.
- Timestream for LiveAnalytics verwendet einen mehrteiligen Upload, um Abfrageergebnisse nach S3 zu exportieren. Wenn Sie von Timestream 4XX oder 5XX für Abfragen erhalten, die eine UNLOAD Anweisung enthalten, LiveAnalytics bricht Timestream LiveAnalytics for den mehrteiligen Upload nach besten Kräften ab, es ist jedoch möglich, dass einige unvollständige Teile zurückbleiben. Daher empfehlen wir, eine auto Bereinigung unvollständiger mehrteiliger Uploads in Ihrem S3-Bucket einzurichten, indem Sie die hier aufgeführten Richtlinien befolgen.

Empfehlungen für den Zugriff auf die Daten im CSV Format mithilfe eines Parsers CSV

- CSVParser erlauben es Ihnen nicht, dasselbe Zeichen als Trennzeichen, Escapezeichen und Anführungszeichen zu verwenden.
- Einige CSV Parser können komplexe Datentypen wie Arrays nicht interpretieren. Wir empfehlen, diese mit Deserializer zu interpretieren. JSON

Empfehlungen für den Zugriff auf die Daten im Parquet-Format

1. Wenn Ihr Anwendungsfall die Unterstützung von UTF -8 Zeichen im Schema, auch bekannt als Spaltenname, erfordert, empfehlen wir die Verwendung der Parquet-MR-Bibliothek.
2. Der Zeitstempel in Ihren Ergebnissen wird als 12-Byte-Ganzzahl () dargestellt INT96
3. Zeitreihen werden so dargestellt`array<row<time, value>>`, dass andere verschachtelte Strukturen entsprechende Datentypen verwenden, die im Parquet-Format unterstützt werden

Verwendung der partition_by-Klausel

- Die im `partitioned_by` Feld verwendete Spalte sollte die letzte Spalte in der Auswahlabfrage sein. Wenn in dem `partitioned_by` Feld mehr als eine Spalte verwendet wird, sollten die Spalten die letzten Spalten in der Auswahlabfrage sein und sich in derselben Reihenfolge befinden, in der sie im `partition_by` Feld verwendet wurden.
- Die Spaltenwerte, die zur Partitionierung der Daten (des `partitioned_by` Felds) verwendet werden, dürfen nur ASCII Zeichen enthalten. Während Timestream for UTF -8 Zeichen in den Werten LiveAnalytics zulässt, unterstützt S3 nur ASCII Zeichen als Objektschlüssel.

Beispiel für einen Anwendungsfall UNLOAD von Timestream für LiveAnalytics

Angenommen, Sie überwachen die Sitzungsmetriken der Benutzer, die Zugriffsquellen und die Produktkäufe Ihrer E-Commerce-Website. Sie verwenden Timestream, LiveAnalytics um in Echtzeit Einblicke in Nutzerverhalten und Produktverkäufe zu gewinnen und Marketinganalysen für Verkehrskanäle (organische Suche, soziale Medien, direkter Traffic, bezahlte Kampagnen und andere) durchzuführen, die Kunden auf die Website lenken.

Themen

Exportieren der Daten ohne Partitionen

Sie möchten Ihre Daten der letzten zwei Tage im CSV Format exportieren.

```
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')
```

Daten nach Kanälen partitionieren

Sie möchten die Daten der letzten zwei Tage im CSV Format exportieren, möchten aber die Daten aus jedem Verkehrskanal in einem separaten Ordner haben. Dazu müssen Sie die Daten mithilfe der `channel` Spalte partitionieren, wie im Folgenden gezeigt.

```
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')
```

Daten nach Ereignis partitionieren

Sie möchten die Daten der letzten zwei Tage im CSV Format exportieren, möchten aber die Daten für jedes Ereignis in einem separaten Ordner haben. Dazu müssen Sie die Daten mithilfe der `event` Spalte partitionieren, wie im Folgenden gezeigt.

```
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')
```

Partitionierung der Daten sowohl nach Kanal als auch nach Ereignis

Sie möchten die Daten der letzten beiden Tage im CSV Format exportieren, möchten aber, dass die Daten für jeden Kanal und innerhalb des Kanals jedes Ereignis in einem separaten Ordner gespeichert werden. Dazu müssen Sie die Daten mithilfe von sowohl als auch `channel` anhand von `event` Spalten partitionieren, wie im Folgenden gezeigt.

```
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')
```

Manifest- und Metadatendateien

Manifestdatei

Die Manifestdatei enthält Informationen zur Liste der Dateien, die bei der UNLOAD Ausführung exportiert werden. Die Manifestdatei ist im bereitgestellten S3-Bucket mit einem Dateinamen verfügbar:`S3://bucket_name/<queryid>_<UUID>_manifest.json`. Die Manifestdatei enthält die URL der Dateien im Ergebnisordner, die Anzahl der Datensätze und die Größe der jeweiligen Dateien sowie die Abfrage-Metadaten (d. h. die Gesamtzahl der Byte und der Gesamtzahl der Zeilen, die für die Abfrage nach S3 exportiert wurden).

```
{
  "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"
  }
}
```

Metadaten

Die Metadatendatei enthält zusätzliche Informationen über den Datensatz, z. B. Spaltenname, Spaltentyp und Schema. <queryid>Die Metadatendatei ist im bereitgestellten S3-Bucket mit dem folgenden Dateinamen verfügbar: S3: //bucket_name/ _< >_metadata.json UUID

Es folgt ein Beispiel für eine Metadatendatei.

```
{
    "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"
    }
}
```

Die in der Metadatendatei gemeinsam genutzten Spalteninformationen haben dieselbe Struktur wie die in der API Abfrageantwort für `SELECT` Abfragen `ColumnInfo` gesendeten Informationen.

Verwenden von Glue-Crawlern zum Erstellen des Glue-Datenkatalogs

1. Melden Sie sich für die folgende Überprüfung mit Admin-Anmeldeinformationen bei Ihrem Konto an.
2. Erstellen Sie eine Crawler for Glue-Datenbank anhand der hier angegebenen Richtlinien. Bitte beachten Sie, dass der S3-Ordner, der in der Datenquelle bereitgestellt werden soll, der `UNLOAD` Ergebnisordner sein sollte, z. B. `s3://my_timestream_unloads/results`
3. Führen Sie den Crawler gemäß den hier angegebenen Richtlinien aus.
4. Sehen Sie sich die Glue-Tabelle an.
   - Gehe zu AWS Glue → Tabellen.
   - Sie werden eine neue Tabelle sehen, die mit dem Tabellenpräfix erstellt wurde, das beim Erstellen des Crawlers angegeben wurde.
   - Sie können die Schema- und Partitionsinformationen einsehen, indem Sie auf die Tabellendetailansicht klicken.

Im Folgenden finden Sie weitere AWS Dienste und Open-Source-Projekte, die den AWS Glue-Datenkatalog verwenden.

- Amazon Athena — Weitere Informationen finden Sie unter Grundlegendes zu Tabellen, Datenbanken und Datenkatalogen im Amazon Athena Athena-Benutzerhandbuch.

- Amazon Redshift Spectrum — Weitere Informationen finden Sie unter Abfragen externer Daten mit Amazon Redshift Spectrum im Amazon Redshift Database Developer Guide.
- Amazon EMR — Weitere Informationen finden Sie unter Verwenden von ressourcenbasierten Richtlinien für den EMR Zugriff von Amazon auf den AWS Glue-Datenkatalog im Amazon EMR Management Guide.
- AWS Glue Data Catalog Client für Apache Hive Metastore — Weitere Informationen zu diesem GitHub Projekt finden Sie unter AWS Glue Data Catalog Client für Apache Hive Metastore.

Grenzwerte für von Timestream für UNLOAD LiveAnalytics

Im Folgenden sind die Grenzwerte aufgeführt, die sich auf den `UNLOAD` Befehl beziehen.

- Die Parallelität von Abfragen, die die `UNLOAD` Anweisung verwenden, beträgt 1 Abfrage pro Sekunde (QPS). Eine Überschreitung der Abfragerate kann zu einer Drosselung führen.
- Abfragen, die `UNLOAD` eine Anweisung enthalten, können maximal 100 Partitionen pro Abfrage exportieren. Wir empfehlen, die eindeutige Anzahl der ausgewählten Spalte zu überprüfen, bevor Sie sie zur Partitionierung der exportierten Daten verwenden.
- Bei Abfragen, die `UNLOAD` eine Anweisung enthalten, wird das Timeout nach 60 Minuten überschritten.
- Die maximale Größe der Dateien, die die `UNLOAD` Anweisung in Amazon S3 erstellt, beträgt 78 GB.

Weitere Beschränkungen für Timestream für finden Sie LiveAnalytics unter Kontingente

Verwenden von Abfrageerkenntnissen zur Optimierung von Abfragen in Amazon Timestream

Query Insights ist eine Funktion zur Leistungsoptimierung, mit der Sie Ihre Abfragen optimieren, ihre Leistung verbessern und Kosten senken können. Mit Query Insights können Sie die Effizienz Ihrer Abfragen anhand von Schlüsseln anhand der zeitlichen, zeitbasierten und räumlichen Partition beurteilen. Mithilfe von Abfrageerkenntnissen können Sie auch Bereiche identifizieren, in denen Verbesserungspotenzial besteht, um die Abfrageleistung zu verbessern. Darüber hinaus können Sie mit Query Insights bewerten, wie effektiv Ihre Abfragen die zeitbasierte und partitionsschlüsselbasierte Indizierung nutzen, um den Datenabruf zu optimieren. Um die Abfrageleistung zu optimieren, ist es wichtig, sowohl die zeitlichen als auch die räumlichen Parameter, die die Abfrageausführung steuern, zu optimieren.

Themen

Vorteile von Query Insights

Im Folgenden sind die wichtigsten Vorteile der Verwendung von Query Insights aufgeführt:

- Identifizierung ineffizienter Abfragen — Query Insights bietet Informationen zur zeit- und attributbasierten Bereinigung der Tabellen, auf die mit der Abfrage zugegriffen wird. Diese Informationen helfen Ihnen dabei, die Tabellen zu identifizieren, auf die nicht optimal zugegriffen wird.
- Optimierung Ihres Datenmodells und Partitionierung — Sie können die Informationen aus den Query-Insights verwenden, um auf Ihr Datenmodell und Ihre Partitionierungsstrategie zuzugreifen und diese zu optimieren.
- Optimieren von Abfragen — Query Insights zeigt Möglichkeiten auf, Indizes effektiver zu nutzen.

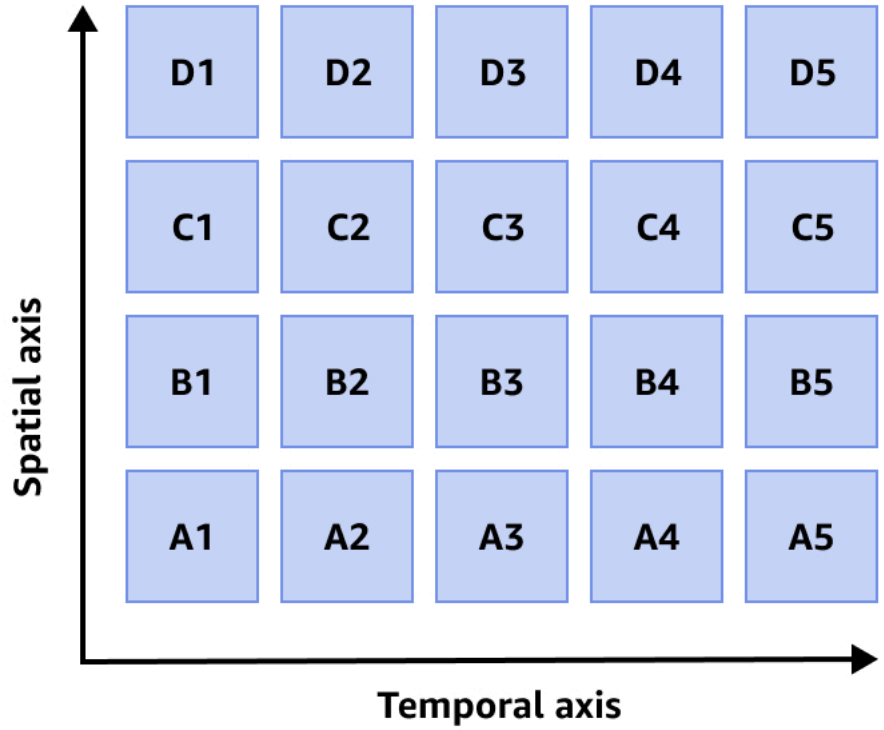
Optimieren des Datenzugriffs in Amazon Timestream

Sie können die Datenzugriffsmuster in Amazon Timestream mithilfe des Timestream-Partitionierungsschemas oder der Datenorganisationstechniken optimieren.

Themen

Timestream-Partitionierungsschema

Amazon Timestream verwendet ein hoch skalierbares Partitionierungsschema, bei dem jede Timestream-Tabelle Hunderte, Tausende oder sogar Millionen unabhängiger Partitionen haben kann. Ein hochverfügbarer Dienst zur Partitionsverfolgung und Indizierung verwaltet die Partitionierung, minimiert die Auswirkungen von Ausfällen und macht das System widerstandsfähiger.



Organisation der Daten

Timestream speichert jeden Datenpunkt, den es aufnimmt, in einer einzigen Partition. Wenn Sie Daten in eine Timestream-Tabelle aufnehmen, erstellt Timestream automatisch Partitionen auf der Grundlage der Zeitstempel, des Partitionsschlüssels und anderer Kontextattribute in den Daten. Neben der zeitlichen Partitionierung der Daten (zeitliche Partitionierung) partitioniert Timestream die Daten auch auf der Grundlage des ausgewählten Partitionierungsschlüssels und anderer Dimensionen (räumliche Partitionierung). Dieser Ansatz ist darauf ausgelegt, den Schreibverkehr zu verteilen und eine effektive Bereinigung von Daten für Abfragen zu ermöglichen.

Die Funktion für Abfrageerkenntnisse bietet wertvolle Einblicke in die Effizienz der Bereinigung von Abfragen, einschließlich der räumlichen Abdeckung von Abfragen und der zeitlichen Abdeckung von Abfragen.

Themen

- QuerySpatialCoverage
- QueryTemporalCoverage

QuerySpatialCoverage

Die QuerySpatialCoverageMetrik bietet Einblicke in die räumliche Abdeckung der ausgeführten Abfrage und in die Tabelle mit der ineffizientesten räumlichen Bereinigung. Diese Informationen können Ihnen helfen, Bereiche zu identifizieren, in denen die Partitionierungsstrategie verbessert werden kann, um die räumliche Kürzung zu verbessern. Der Wert für die `QuerySpatialCoverage` Metrik liegt zwischen 0 und 1. Je niedriger der Wert der Metrik ist, desto optimaler ist die Bereinigung der Abfrage auf der räumlichen Achse. Ein Wert von 0,1 gibt beispielsweise an, dass die Abfrage 10% der räumlichen Achse scannt. Ein Wert von 1 gibt an, dass die Abfrage 100% der räumlichen Achse scannt.

Example Verwenden von Abfrageerkenntnissen zur Analyse der räumlichen Abdeckung einer Abfrage

Angenommen, Sie haben eine Timestream-Datenbank, die Wetterdaten speichert. Gehen Sie davon aus, dass die Temperatur stündlich von Wetterstationen in verschiedenen Bundesstaaten der USA aufgezeichnet wird. Stellen Sie sich vor, Sie wählen `State` als benutzerdefinierten Partitionsschlüssel (CDPK) die Partitionierung der Daten nach Status.

Angenommen, Sie führen eine Abfrage aus, um die Durchschnittstemperatur aller Wetterstationen in Kalifornien zwischen 14 Uhr und 16 Uhr an einem bestimmten Tag abzurufen. Das folgende Beispiel zeigt die Abfrage für dieses Szenario.

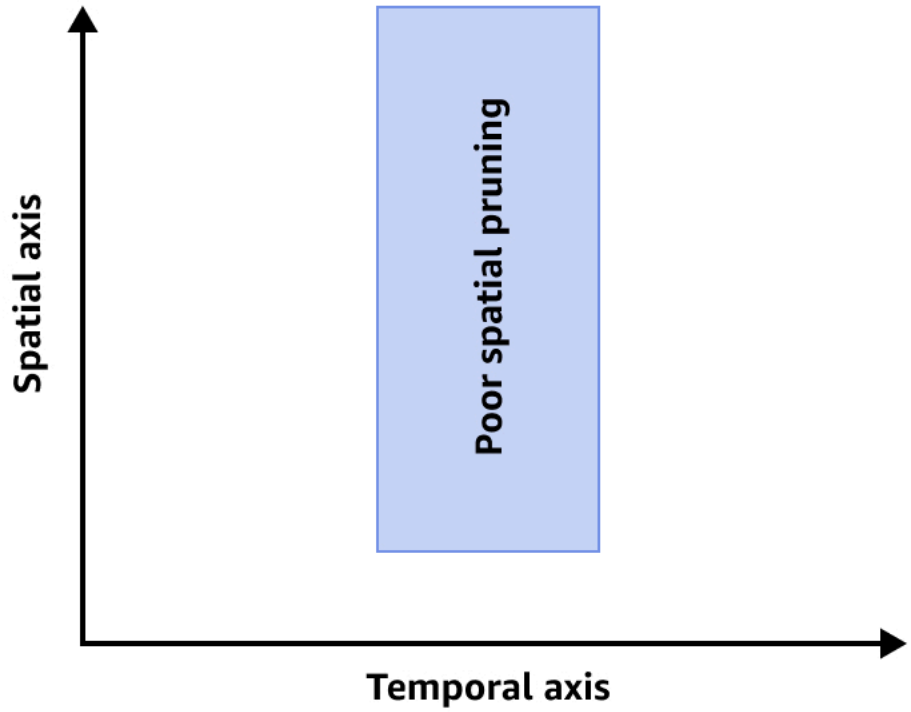
```
SELECT AVG(temperature)
FROM "weather_data"."hourly_weather"
WHERE time >= '2024-10-01 14:00:00' AND time < '2024-10-01 16:00:00'
  AND state = 'CA';
```

Mithilfe der Funktion Query Insights können Sie die räumliche Abdeckung der Abfrage analysieren. Stellen Sie sich vor, dass die `QuerySpatialCoverage` Metrik einen Wert von 0,02 zurückgibt. Das bedeutet, dass die Abfrage nur 2% der räumlichen Achse gescannt hat, was effizient ist. In diesem Fall war die Abfrage in der Lage, den räumlichen Bereich effektiv zu kürzen, indem nur Daten aus Kalifornien abgerufen und Daten aus anderen Bundesstaaten ignoriert wurden.

Im Gegenteil, wenn die `QuerySpatialCoverage` Metrik einen Wert von 0,8 zurückgibt, würde dies bedeuten, dass die Abfrage 80% der räumlichen Achse gescannt hat, was weniger effizient ist. Dies könnte darauf hindeuten, dass die Partitionierungsstrategie verfeinert werden muss, um die räumliche Bereinigung zu verbessern. Sie können den Partitionsschlüssel beispielsweise als Stadt oder Region statt als Bundesland auswählen. Durch die Analyse der `QuerySpatialCoverage` Metrik können

Sie Möglichkeiten zur Optimierung Ihrer Partitionierungsstrategie und zur Verbesserung der Leistung Ihrer Abfragen identifizieren.

Die folgende Abbildung zeigt eine schlechte räumliche Bereinigung.



Um die Effizienz beim räumlichen Beschneiden zu verbessern, können Sie eine oder beide der folgenden Aktionen ausführen:

- Fügen Sie `measure_name` den Standard-Partitionierungsschlüssel hinzu, oder verwenden Sie die CDPK Prädikate in Ihrer Abfrage.
- Wenn Sie die im vorherigen Punkt genannten Attribute bereits hinzugefügt haben, entfernen Sie Funktionen rund um diese Attribute oder Klauseln, z. B. `LIKE`

QueryTemporalCoverage

Die `QueryTemporalCoverage` Metrik bietet Einblicke in den Zeitbereich, der von der ausgeführten Abfrage gescannt wurde, einschließlich der Tabelle mit dem größten gescannten Zeitraum. Der Wert für die `QueryTemporalCoverage` Metrik ist der Zeitbereich, der in Nanosekunden dargestellt wird.

Je niedriger der Wert dieser Metrik ist, desto optimaler ist die Abfragebereinigung im Zeitbereich. Beispielsweise ist eine Abfrage, die Daten der letzten Minuten scannt, leistungsfähiger als eine Abfrage, die den gesamten Zeitraum der Tabelle scannt.

Example

Angenommen, Sie haben eine Timestream-Datenbank, in der IoT-Sensordaten gespeichert werden, wobei jede Minute Messungen von Geräten in einer Produktionsstätte durchgeführt werden. Gehen Sie davon aus, dass Sie Ihre Daten nach partitioniert haben. `device_ID`

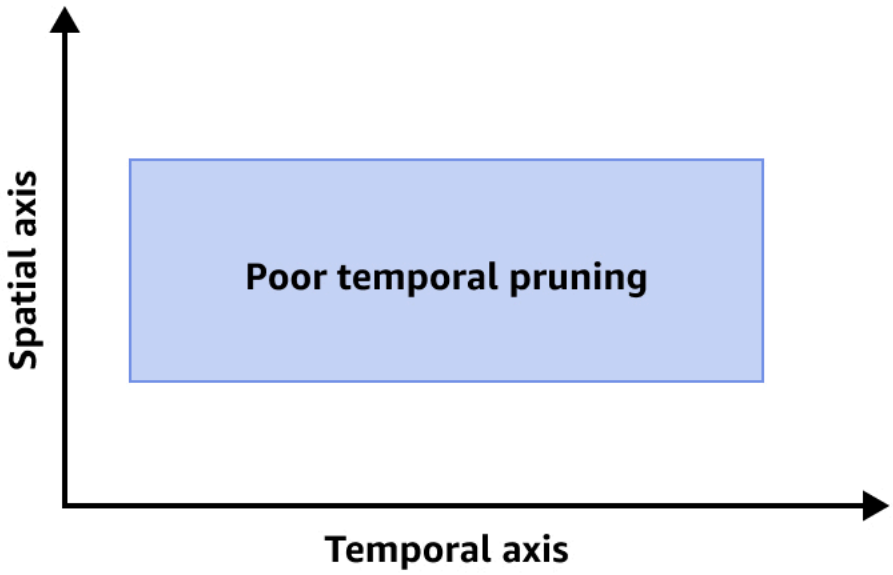
Angenommen, Sie führen eine Abfrage aus, um den durchschnittlichen Sensorwert für ein bestimmtes Gerät in den letzten 30 Minuten abzurufen. Das folgende Beispiel zeigt die Abfrage für dieses Szenario.

```
SELECT AVG(sensor_reading)
FROM "sensor_data"."factory_1"
WHERE device_id = 'DEV_123'
  AND time >= NOW() - INTERVAL 30 MINUTE and time < NOW();
```

Mithilfe der Funktion für Abfrageerkenntnisse können Sie den von der Abfrage gescannten Zeitbereich analysieren. Stellen Sie sich vor, die `QueryTemporalCoverage` Metrik gibt einen Wert von 1800000000000 Nanosekunden (30 Minuten) zurück. Das bedeutet, dass die Abfrage nur die Daten der letzten 30 Minuten gescannt hat, was ein relativ enger Zeitbereich ist. Dies ist ein gutes Zeichen, da es darauf hinweist, dass die Abfrage die zeitliche Partitionierung effektiv bereinigen konnte und nur die angeforderten Daten abgerufen hat.

Im Gegenteil, wenn die `QueryTemporalCoverage` Metrik einen Wert von 1 Jahr in Nanosekunden zurückgibt, bedeutet dies, dass die Abfrage einen Zeitraum von einem Jahr in der Tabelle gescannt hat, was weniger effizient ist. Dies könnte darauf hindeuten, dass die Abfrage nicht für die zeitliche Bereinigung optimiert ist, und Sie könnten sie verbessern, indem Sie Zeitfilter hinzufügen.

Die folgende Abbildung zeigt ein schlechtes zeitliches Beschneiden.



Um den zeitlichen Schnitt zu verbessern, empfehlen wir Ihnen, einen oder alle der folgenden Schritte durchzuführen:

- Fügen Sie der Abfrage die fehlenden Zeitprädikate hinzu und stellen Sie sicher, dass die Zeitprädikate das gewünschte Zeitfenster bereinigen.
- Entfernen Sie Funktionen wie etwa `MAX( )` Zeitprädikate.
- Fügen Sie allen Unterabfragen Zeitprädikate hinzu. Dies ist wichtig, wenn Ihre Unterabfragen große Tabellen verknüpfen oder komplexe Operationen ausführen.

Aktivierung von Query-Insights in Amazon Timestream

Sie können Query-Insights für Ihre Abfragen mithilfe von Erkenntnissen aktivieren, die direkt über die Abfrageantwort bereitgestellt werden. Für die Aktivierung von Query Insights ist keine zusätzliche Infrastruktur erforderlich und es fallen keine zusätzlichen Kosten an. Wenn Sie Query Insights aktivieren, werden zusätzlich zu den Abfrageergebnissen als Teil Ihrer Abfrageantwort Metadatenfelder zurückgegeben, die sich auf die Abfrageleistung beziehen. Sie können diese Informationen verwenden, um Ihre Abfragen zu optimieren, um die Abfrageleistung zu verbessern und die Abfragekosten zu senken.

Informationen zur Aktivierung von Query Insights finden Sie unterAusführen einer -Abfrage.

Beispiele für Antworten, die durch die Aktivierung von Query Insights zurückgegeben wurden, finden Sie unter Beispiele für geplante Abfragen.

> **Note**
>
> - Wenn Sie Query Insights aktivieren, begrenzt die Rate die Abfrage auf 1 Abfrage pro Sekunde (QPS). Um Leistungseinbußen zu vermeiden, empfehlen wir dringend, Query Insights nur während der Evaluierungsphase Ihrer Abfragen zu aktivieren, bevor Sie sie in der Produktion einsetzen.
> - Die in Query Insights bereitgestellten Erkenntnisse sind letztlich konsistent, was bedeutet, dass sie sich ändern können, wenn kontinuierlich neue Daten in die Tabellen aufgenommen werden.

Optimierung von Abfragen mithilfe von Query Insights Response

Angenommen, Sie verwenden Amazon Timestream LiveAnalytics , um den Energieverbrauch an verschiedenen Standorten zu überwachen. Stellen Sie sich vor, Sie haben zwei Tabellen in Ihrer Datenbank mit dem Namen `raw-metrics` und`aggregate-metrics`.

Die `raw-metrics` Tabelle speichert detaillierte Energiedaten auf Geräteebene und enthält die folgenden Spalten:

- Zeitstempel
- Bundesstaat, zum Beispiel Washington
- Geräte-ID
- Energieverbrauch

Die Daten für diese Tabelle werden mit einer bestimmten minute-by-minute Granularität gesammelt und gespeichert. Die Tabelle verwendet `State` als. CDPK

In der `aggregate-metrics` Tabelle wird das Ergebnis einer geplanten Abfrage gespeichert, um die Energieverbrauchsdaten aller Geräte stündlich zu aggregieren. Diese Tabelle enthält die folgenden Spalten:

- Zeitstempel
- Bundesstaat, zum Beispiel Washington
- Gesamter Energieverbrauch

In der `aggregate-metrics` Tabelle werden diese Daten mit stündlicher Granularität gespeichert. Die Tabelle verwendet `State` als. CDPK

Themen

Der Energieverbrauch der letzten 24 Stunden wird abgefragt

Angenommen, Sie möchten den gesamten Energieverbrauch ermitteln, der in Washington in den letzten 24 Stunden verbraucht wurde. Um diese Daten zu finden, können Sie die Stärken der beiden Tabellen nutzen: `raw-metrics` und`aggregate-metrics`. Die `aggregate-metrics` Tabelle enthält stündliche Energieverbrauchsdaten für die letzten 23 Stunden, während die `raw-metrics` Tabelle minutengenaue Daten für die letzte Stunde enthält. Wenn Sie beide Tabellen abfragen, können Sie sich ein vollständiges und genaues Bild vom Energieverbrauch in Washington in den letzten 24 Stunden machen.

```
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()
```

Diese Beispielabfrage dient nur zur Veranschaulichung und funktioniert möglicherweise nicht so, wie sie ist. Sie soll das Konzept veranschaulichen, aber Sie müssen es möglicherweise an Ihren spezifischen Anwendungsfall oder Ihre Umgebung anpassen.

Nach der Ausführung dieser Abfrage stellen Sie möglicherweise fest, dass die Antwortzeit der Abfrage langsamer als erwartet ist. Um die Ursache für dieses Leistungsproblem zu ermitteln, können Sie die Query Insights-Funktion verwenden, um die Leistung der Abfrage zu analysieren und ihre Ausführung zu optimieren.

Das folgende Beispiel zeigt die Antwort auf Query Insights.

```
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
```

Die Query Insights-Antwort enthält die folgenden Informationen:

- Zeitlicher Bereich: Die Abfrage hat einen zu großen Zeitbereich von 365 Tagen für die `aggregate-metrics` Tabelle durchsucht. Dies deutet auf eine ineffiziente Verwendung der zeitlichen Filterung hin.
- Räumliche Abdeckung: Die Abfrage hat den gesamten räumlichen Bereich (100%) der `raw-metrics` Tabelle gescannt. Dies deutet darauf hin, dass die räumliche Filterung nicht effektiv genutzt wird.

Wenn Ihre Abfrage auf mehr als eine Tabelle zugreift, liefert Query Insights die Metriken für die Tabelle mit den suboptimalsten Zugriffsmustern.

Optimierung der Abfrage für den Zeitbereich

Basierend auf der Query Insights-Antwort können Sie die Abfrage für den Zeitbereich optimieren, wie im folgenden Beispiel gezeigt.

```
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'
```

Wenn Sie den `QueryInsights` Befehl erneut ausführen, wird die folgende Antwort zurückgegeben.

```
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
```

Diese Antwort zeigt, dass die räumliche Abdeckung der `aggregate-metrics` Tabelle immer noch 100% beträgt, was ineffizient ist. Im folgenden Abschnitt wird gezeigt, wie die Abfrage für die räumliche Abdeckung optimiert werden kann.

Optimierung der Abfrage im Hinblick auf die räumliche Abdeckung

Auf der Grundlage der Query Insights-Antwort können Sie die Abfrage im Hinblick auf die räumliche Abdeckung optimieren, wie im folgenden Beispiel gezeigt.

```
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'
```

Wenn Sie den `QueryInsights` Befehl erneut ausführen, wird die folgende Antwort zurückgegeben.

```
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
```

Die Abfrageleistung wurde verbessert

Nach der Optimierung der Abfrage bietet Query Insights die folgenden Informationen:

- Der Zeitaufwand für die `aggregate-metrics` Bereinigung der Tabelle beträgt 23 Stunden. Dies bedeutet, dass nur 23 Stunden des Zeitbereichs gescannt werden.
- Der räumliche Schnitt für die `aggregate-metrics` Tabelle beträgt 0,02. Dies bedeutet, dass nur 2% der räumlichen Bereichsdaten der Tabelle gescannt werden. Die Abfrage scannt nur einen sehr kleinen Teil der Tabellen, was zu einer schnelleren Leistung und einer geringeren Ressourcenauslastung führt. Die verbesserte Bereinigungseffizienz deutet darauf hin, dass die Abfrage jetzt leistungsoptimiert ist.

Arbeiten mit AWS Backup

Die Datenschutzfunktion in Amazon Timestream for LiveAnalytics ist eine vollständig verwaltete Lösung, mit der Sie Ihre Anforderungen an die Einhaltung gesetzlicher Vorschriften und die Geschäftskontinuität erfüllen können. Diese Funktionalität wird durch die native Integration mit einem einheitlichen Backup-Service ermöglicht AWS Backup, der die Erstellung, Migration, Wiederherstellung und Löschung von Backups vereinfacht und gleichzeitig eine verbesserte Berichterstattung und Prüfung bietet. Durch die Integration mit AWS Backup können Sie eine vollständig verwaltete, richtliniengesteuerte, zentrale Datenschutzlösung verwenden, um unveränderliche Backups zu erstellen und den Datenschutz Ihrer Anwendungsdaten zentral zu verwalten. Dies umfasst Timestream und andere Dienste, die von unterstützt werden. AWS AWS Backup

Um diese Funktion nutzen zu können, müssen Sie sich für den Schutz Ihrer Timestream-Ressourcen anmelden. AWS Backup Die Opt-in-Optionen gelten für das jeweilige Konto und die AWS Region, sodass Sie sich möglicherweise mit demselben Konto für mehrere Regionen anmelden müssen. Weitere Informationen zu AWS Backup finden Sie im AWS Backup Developer Guide.

Zu den Datenschutzfunktionen, die über verfügbar sind, AWS Backup gehören:

Geplante Backups — Mithilfe von Backup-Plänen können Sie regelmäßig geplante Backups Ihres Timestreams für LiveAnalytics Tabellen einrichten.

Konto- und regionsübergreifendes Kopieren — Sie können Ihre Backups automatisch in einen anderen Backup-Tresor in einer anderen AWS Region oder einem anderen Konto kopieren, sodass Sie Ihre Datenschutzanforderungen erfüllen können.

Cold Storage Tiering — Sie können Ihre Backups so konfigurieren, dass Lebenszyklusregeln implementiert werden, um Backups zu löschen oder in einen kälteren Speicher zu übertragen. Dies kann Ihnen helfen, Ihre Backup-Kosten zu optimieren.

Stichwörter — Sie können Ihre Backups zur Abrechnung und Kostenzuweisung automatisch taggen.

Verschlüsselung — Ihre Backup-Daten werden im Tresor gespeichert. AWS Backup Auf diese Weise können Sie Ihre Backups verschlüsseln und sichern, indem Sie einen AWS KMS Schlüssel verwenden, der unabhängig von Ihrem Timestream für die LiveAnalytics Tabellenverschlüsselung ist.

Sichere Backups mithilfe des WORM Modells — Sie können AWS Backup Vault Lock verwenden, um eine write-once-read-many (WORM) -Einstellung für Ihre Backups zu aktivieren. Mit AWS Backup

Vault Lock können Sie eine zusätzliche Schutzebene hinzufügen, die Backups vor unbeabsichtigten oder böswilligen Löschvorgängen, Änderungen der Aufbewahrungsfristen von Backups und Aktualisierungen der Lebenszykluseinstellungen schützt. Weitere Informationen hierzu finden Sie unter AWS Backup -Vault Lock.

Die Datenschutzfunktion ist in allen Regionen verfügbar. Weitere Informationen zu den Funktionen finden Sie im AWS Backup Entwicklerhandbuch.

Timestream-Tabellen sichern und wiederherstellen: So funktioniert's

Sie können Backups Ihrer Amazon Timestream-Tabellen erstellen. Dieser Abschnitt bietet eine Übersicht über die Aktionen während des Backup- und Wiederherstellungsvorgangs.

Themen

- Sicherungen
- Wiederherstellen

Sicherungen

Sie können die On-Demand-Backup-Funktion verwenden, um vollständige Backups Ihres Amazon Timestream für LiveAnalytics Tabellen zu erstellen. Dieser Abschnitt bietet eine Übersicht über die Aktionen während des Backup- und Wiederherstellungsvorgangs.

Sie können eine Sicherungskopie Ihrer Timestream-Daten mit einer Tabellengranularität erstellen. Sie können eine Sicherung der ausgewählten Tabelle entweder mit der Timestream-Konsole oder der Konsole, oder AWS Backup initiieren. SDK CLI Das Backup wird asynchron erstellt und alle Daten in der Tabelle bis zum Zeitpunkt der Initiierung des Backups sind im Backup enthalten. Es besteht jedoch die Möglichkeit, dass einige der Daten, die während der Sicherung in die Tabelle aufgenommen wurden, auch in der Sicherung enthalten sind. Um Ihre Daten zu schützen, können Sie entweder ein einmaliges On-Demand-Backup erstellen oder ein wiederkehrendes Backup Ihrer Tabelle planen.

Während ein Backup ausgeführt wird, können Sie Folgendes nicht tun.

- Den Backupvorgang anhalten oder abbrechen.
- Die Quelltabelle des Backups löschen.
- Backups für eine Tabelle deaktivieren, wenn ein Backup für diese Tabelle gerade ausgeführt wird.

Nach der Konfiguration AWS Backup bietet es automatische Backup-Zeitpläne, Aufbewahrungsmanagement und Lebenszyklusmanagement, sodass keine benutzerdefinierten Skripts und manuellen Prozesse erforderlich sind. Weitere Informationen finden Sie im AWS Backup Entwicklerhandbuch

Alle LiveAnalytics Timestream-Datensicherungen sind inkrementeller Natur, was bedeutet, dass es sich bei der ersten Sicherung einer Tabelle um eine vollständige Sicherung handelt und bei jeder nachfolgenden Sicherung derselben Tabelle um eine inkrementelle Sicherung, bei der nur die Änderungen an den Daten seit der letzten Sicherung kopiert werden. Da die Daten in Timestream for LiveAnalytics in einer Sammlung von Partitionen gespeichert werden, werden alle Partitionen, die sich entweder aufgrund der Aufnahme neuer Daten oder aufgrund von Aktualisierungen der vorhandenen Daten seit der letzten Sicherung geändert haben, bei nachfolgenden Backups kopiert.

Wenn Sie Timestream für die LiveAnalytics Konsole verwenden, werden die für alle Ressourcen im Konto erstellten Backups auf der Registerkarte Backups aufgeführt. Darüber hinaus sind die Backups auch in den Tabellendetails aufgeführt.

Wiederherstellen

Sie können eine Tabelle aus dem Timestream für LiveAnalytics Konsole oder AWS Backup KonsoleSDK, oder AWS CLI wiederherstellen. Sie können entweder die gesamten Daten aus Ihrem Backup wiederherstellen oder die Tabellenaufbewahrungseinstellungen so konfigurieren, dass ausgewählte Daten wiederhergestellt werden. Wenn Sie eine Wiederherstellung initiieren, können Sie die folgenden Tabelleneinstellungen konfigurieren.

- Datenbank-Name
- Tabellenname
- Aufbewahrung des Speichers
- Aufbewahrung im Magnetspeicher
- Schreibvorgänge im Magnetspeicher aktivieren
- Speicherort der S3-Fehlerprotokolle (optional)
- IAMRolle, AWS Backup die beim Wiederherstellen des Backups übernommen wird

Die vorherigen Konfigurationen sind unabhängig von der Quelltabelle. Um alle Daten in Ihrer Sicherung wiederherzustellen, empfehlen wir Ihnen, die neuen Tabelleneinstellungen so zu konfigurieren, dass die Summe aus Aufbewahrungsdauer des Speicherspeichers und Aufbewahrungsdauer des Magnetspeichers größer ist als die Differenz zwischen dem ältesten

Zeitstempel und dem aktuellen Zeitstempel. Wenn Sie ein Backup auswählen, das inkrementell wiederhergestellt werden soll, werden alle Daten (inkrementell und zugrundeliegende vollständige Daten) wiederhergestellt. Nach erfolgreicher Wiederherstellung befindet sich die Tabelle im aktiven Zustand, und Sie können Aufnahme- und/oder Abfrageoperationen für die wiederhergestellte Tabelle ausführen. Sie können diese Vorgänge jedoch nicht ausführen, während die Wiederherstellung läuft. Nach der Wiederherstellung ähnelt die Tabelle jeder anderen Tabelle in Ihrem Konto.

Example Stellen Sie alle Daten aus einem Backup wieder her

Dieses Beispiel basiert auf den folgenden Annahmen.

Ältester Zeitstempel — `August 1, 2021 0:00:00`

- Jetzt — `November 9, 2022 0:00:00`

Um alle Daten aus einem Backup wiederherzustellen, geben Sie Werte wie folgt ein und vergleichen Sie sie.

1. Geben Sie Memory Store Retention und Magnetic Store Retention ein. Gehen Sie beispielsweise von diesen Werten aus.
   - Aufbewahrung des Speichers —12 Stunden
   - Aufbewahrung im Magnetspeicher —500 Tage
2. Ermitteln Sie die Summe aus Speicherretention und Magnetic Store-Aufbewahrung.

   ```
   12 hours + (500 * 24 hours) =
   12 hours + 12,000 hours =
   12,012 hours
   ```

3. Finden Sie den Unterschied zwischen dem ältesten Zeitstempel und dem aktuellen Zeitstempel heraus.

   ```
   November 9, 2022 0:00:00 - August 1, 2021 0:00:00 =
   465 days =
   465 * 24 hours =
   11,160 hours
   ```

4. Stellen Sie sicher, dass die Summe der Aufbewahrungswerte im zweiten Schritt größer ist als die Differenz der Zeiten im dritten Schritt. Passen Sie bei Bedarf die Aufbewahrungszeiten an.

   ```
   12,012 > 11,160
   ```

```
true
```

Example Ausgewählte Daten aus einem Backup wiederherstellen

In diesem Beispiel wird von der folgenden Annahme ausgegangen.

- Jetzt — `November 9, 2022 0:00:00`

Um nur ausgewählte Daten aus einem Backup wiederherzustellen, geben Sie Werte wie folgt ein und vergleichen Sie sie.

1. Ermitteln Sie den frühesten erforderlichen Zeitstempel. Nehmen wir zum Beispiel an`December 4, 2021 0:00:00`.
2. Finden Sie den Unterschied zwischen dem frühesten benötigten Zeitstempel und dem aktuellen Zeitstempel heraus.

   ```
   November 9, 2022 0:00:00 - December 4, 2021 0:00:00 =
   340 days =
   340 * 24 hours =
   8,160 hours
   ```
3. Geben Sie den gewünschten Wert für die Aufbewahrung des Speichers ein. Geben Sie beispielsweise 12 Stunden ein.
4. Subtrahieren Sie den Wert von der Differenz im zweiten Schritt.

   ```
   8,160 hours - 12 hours =
   8148 hours
   ```
5. Geben Sie diesen Wert für Magnetic Store Retention ein.

Sie können eine Sicherungskopie Ihrer Timestream-Daten für LiveAnalytics Tabellen in eine andere AWS Region kopieren und sie dann in dieser neuen Region wiederherstellen. Sie können Backups zwischen AWS kommerziellen Regionen und Regionen AWS GovCloud (USA) kopieren und dann wiederherstellen. Sie zahlen nur für die Daten, die Sie aus der Quellregion kopieren, und Daten, die Sie in eine neue Tabelle in der Zielregion wieder herstellen.

Sobald die Tabelle wiederhergestellt ist, müssen Sie für die wiederhergestellte Tabelle manuell Folgendes einrichten.

- AWS Richtlinien für Identity and Access Management (IAM)
- Tags
- Geplante Abfragen

Die Wiederherstellungszeiten stehen in direktem Zusammenhang mit der Konfiguration Ihrer Tabellen. Dazu gehören die Größe Ihrer Tabellen, die Anzahl der zugrunde liegenden Partitionen, die Menge der im Speicher wiederhergestellten Daten und andere Variablen. Eine bewährte Methode bei der Planung der Notfallwiederherstellung besteht darin, die durchschnittlichen Wiederherstellungszeiten regelmäßig zu dokumentieren und festzustellen, wie sich diese Zeiten auf Ihr allgemeines Wiederherstellungszeitziel (RTO) auswirken.

Alle Konsolen und API Aktionen zur Sicherung und Wiederherstellung werden erfasst und AWS CloudTrail zur Protokollierung, kontinuierlichen Überwachung und Prüfung aufgezeichnet.

Erstellen von Backups von Amazon Timestream Timestream-Tabellen

In diesem Abschnitt wird beschrieben, wie Sie On-Demand-Backups und geplante Backups für Amazon Timestream aktivieren AWS Backup und erstellen.

Themen

Aktivierung AWS Backup zum Schutz von Timestream für Daten LiveAnalytics

Sie müssen es aktivieren AWS Backup , um es mit Timestream for verwenden zu können. LiveAnalytics

Gehen Sie zur Aktivierung AWS Backup im Timestream für die LiveAnalytics Konsole wie folgt vor.

1. Melden Sie sich bei der AWS Management Console an.
2. Oben auf Ihrer Timestream for LiveAnalytics Dashboard-Seite erscheint ein Popup-Banner, mit dem Sie Timestream for LiveAnalytics Data unterstützen AWS Backup können. Andernfalls wählen Sie im Navigationsbereich Backups aus.
3. Im Backup-Fenster sehen Sie das Banner zum Aktivieren AWS Backup. Wählen Sie Enable (Aktivieren) aus.

Data Protection through AWS Backup ist jetzt für Ihren Timestream für LiveAnalytics Tabellen verfügbar.

Informationen zur Aktivierung über AWS Backup finden Sie in der AWS Backup Dokumentation zur Aktivierung über die Konsole und programmgesteuert.

Wenn Sie Ihren Timestream für LiveAnalytics Daten deaktivieren AWS Backup möchten, nachdem diese aktiviert wurden, melden Sie sich über die AWS Backup Konsole an und bewegen Sie den Schalter nach links.

Wenn Sie die AWS Backup Funktionen nicht aktivieren oder deaktivieren können, muss Ihr AWS Administrator diese Aktionen möglicherweise durchführen.

Erstellen eines On-Demand-Backups

Gehen Sie wie folgt vor, um ein On-Demand-Backup eines Timestreams für eine LiveAnalytics Tabelle zu erstellen.

1. Melden Sie sich bei der AWS Management Console an.
2. Klicken Sie im Navigationsbereich auf der linken Seite der Konsole auf Backups (Backups).
3. Wählen Sie On-Demand-Backup erstellen.
4. Fahren Sie mit der Auswahl der Einstellungen im Backup-Fenster fort.
5. Sie können entweder jetzt ein Backup erstellen, sofort ein Backup starten oder ein Backup-Fenster auswählen, um das Backup zu starten.
6. Wählen Sie die Lifecycle-Management-Richtlinie für Ihr Backup aus. Sie können Ihre Backup-Daten in einen Cold Storage verschieben, wo Sie das Backup mindestens 90 Tage lang aufbewahren müssen. Sie können den erforderlichen Aufbewahrungszeitraum für Ihr Backup festlegen. Sie können entweder einen vorhandenen Tresor auswählen oder Neuen Backup-Tresor erstellen auswählen, um zur AWS Backup Konsole zu navigieren und einen neuen Backup-Tresor zu erstellen **<documentation link on creating a new backup vault here>**
7. Wählen Sie die entsprechende IAM Rolle aus.
8. Wenn Sie Ihrer On-Demand-Sicherung einen oder mehrere Tags zuweisen möchten, geben Sie einen key (Schlüssel) und optional einen value (Wert) ein und wählen Sie Add tag (Tag hinzufügen).
9. Wählen Sie, ob Sie ein Backup auf Abruf erstellen möchten. Dadurch gelangen Sie zur Backup-Seite, auf der Sie eine Liste von Jobs sehen.

10. Wählen Sie die Backup job ID (Sicherungsaufgaben-ID) für die Ressource aus, die Sie für die Sicherung ausgewählt haben, um die Details der Aufgabe anzuzeigen.

Geplante Backups

Informationen zum Planen eines Backups finden Sie unter Ein geplantes Backup erstellen.

Wiederherstellung einer Sicherung einer Amazon Timestream Timestream-Tabelle

In diesem Abschnitt wird beschrieben, wie Sie eine Sicherung einer Amazon Timestream Timestream-Tabelle wiederherstellen.

Themen

- Wiederherstellen eines Timestreams für eine Tabelle von LiveAnalytics AWS Backup
- Einen Timestream für eine LiveAnalytics Tabelle in einer anderen Region oder einem anderen Konto wiederherstellen

Wiederherstellen eines Timestreams für eine Tabelle von LiveAnalytics AWS Backup

Gehen Sie wie folgt vor, um Ihren Timestream für die LiveAnalytics Tabelle AWS Backup mithilfe von Timestream für die LiveAnalytics Konsole wiederherzustellen.

1. Melden Sie sich bei der AWS Management Console an.
2. Klicken Sie im Navigationsbereich auf der linken Seite der Konsole auf Backups (Backups).
3. Um eine Ressource wiederherzustellen, wählen Sie das Optionsfeld neben der Wiederherstellungspunkt-ID der Ressource. Wählen Sie in der oberen rechten Ecke des Bereichs die Option Wiederherstellen.
4. Geben Sie die Tabellenkonfigurationseinstellungen ein, nämlich Datenbankname und Tabellenname. Bitte beachten Sie, dass sich der Name der wiederhergestellten Tabelle vom Namen der ursprünglichen Quelltabelle unterscheiden sollte.
5. Konfigurieren Sie die Einstellungen für den Speicher und die Aufbewahrung des Magnetspeichers.
6. Wählen Sie unter Rolle wiederherstellen die IAM Rolle aus, die für diese Wiederherstellung übernommen AWS Backup werden soll.
7. Wählen Sie Restore backup aus. Eine Meldung am Anfang der Seite enthält Informationen zu dem Wiederherstellungsauftrag.

> **Note**
>
> Die Wiederherstellung des gesamten Backups wird Ihnen unabhängig von den konfigurierten Aufbewahrungsfristen für Speicher und Magnetspeicher in Rechnung gestellt. Sobald die Wiederherstellung abgeschlossen ist, enthält Ihre wiederhergestellte Tabelle jedoch nur die Daten innerhalb der konfigurierten Aufbewahrungsfristen.

Einen Timestream für eine LiveAnalytics Tabelle in einer anderen Region oder einem anderen Konto wiederherstellen

Um einen Timestream für eine LiveAnalytics Tabelle in einer anderen Region oder einem anderen Konto wiederherzustellen, müssen Sie zuerst das Backup in diese neue Region oder dieses neue Konto kopieren. Um auf ein anderes Konto zu kopieren, muss dieses Konto Ihnen zuerst die Erlaubnis erteilen. Nachdem Sie Ihren Timestream zur LiveAnalytics Sicherung in die neue Region oder das neue Konto kopiert haben, kann er mit dem im vorherigen Abschnitt beschriebenen Vorgang wiederhergestellt werden.

Kopieren einer Sicherung einer Amazon Timestream Timestream-Tabelle

Sie können eine Kopie eines aktuellen Backups erstellen. Sie können Backups bei Bedarf oder automatisch im Rahmen eines geplanten Sicherungsplans auf mehrere AWS Konten oder AWS Regionen kopieren. Regionsübergreifende Replikation ist insbesondere wertvoll, wenn Sie über Betriebskontinuität oder Compliance-Anforderungen verfügen, um Backups in einem Mindestabstand von Ihren Produktionsdaten zu speichern.

Kontoübergreifende Backups sind nützlich, um Ihre Backups sicher in eine oder mehrere AWS -Konten in Ihrer Organisation aus betrieblichen oder Sicherheitsgründen zu kopieren. Wenn Ihr ursprüngliches Backup versehentlich gelöscht wird, können Sie das Backup von seinem Zielkonto in das Quellkonto kopieren und dann die Wiederherstellung starten. Bevor Sie dies tun können, müssen Sie über zwei Konten verfügen, die derselben Organisation im Organizations-Service gehören, und über die erforderlichen Berechtigungen für die Konten verfügen. Wenn Sie ein inkrementelles Backup in ein anderes Konto oder eine andere Region kopieren, wird das zugehörige vollständige Backup ebenfalls kopiert.

Kopien übernehmen die Konfiguration des Quell-Backups, sofern Sie nichts anderes angeben. Es gibt eine Ausnahme. Wenn Sie angeben, dass Ihre neue Kopie „Niemals“ abläuft. Mit dieser Einstellung erbt die neue Kopie weiterhin ihr Ablaufdatum von der Quelle. Wenn Sie möchten, dass

Ihr neues Backup dauerhaft ist, legen Sie entweder fest, dass Ihre Quellbackups niemals ablaufen, oder geben Sie an, dass Ihre neue Kopie 100 Jahre nach ihrer Erstellung abläuft.

Gehen Sie wie folgt vor, um ein Backup von der Timestream-Konsole zu kopieren.

1. Melden Sie sich bei der AWS Management Console an.
2. Klicken Sie im Navigationsbereich auf der linken Seite der Konsole auf Backups (Backups).
3. Wählen Sie das Optionsfeld neben der Wiederherstellungspunkt-ID der Ressource. Wählen Sie in der oberen rechten Ecke des Bereichs Aktionen und dann Kopieren aus.
4. Wählen Sie Weiter zur AWS Sicherung und folgen Sie den Schritten für kontoübergreifendes Backup.

Das Kopieren von On-Demand-Backups und geplanten Backups zwischen Konten und Regionen wird derzeit nicht nativ in der LiveAnalytics Timestream-Konsole unterstützt, und Sie müssen zu navigieren, AWS Backup um den Vorgang durchzuführen.

Löschen eines Backups

In diesem Abschnitt wird beschrieben, wie Sie ein Backup eines Timestreams für eine Tabelle löschen. LiveAnalytics

Gehen Sie folgendermaßen vor, um ein Backup aus der Timestream-Konsole zu löschen.

1. Melden Sie sich bei der AWS Management Console an.
2. Klicken Sie im Navigationsbereich auf der linken Seite der Konsole auf Backups (Backups).
3. Wählen Sie das Optionsfeld neben der Wiederherstellungspunkt-ID der Ressource. Wählen Sie in der oberen rechten Ecke des Bereichs Aktionen und dann Löschen aus.
4. Wählen Sie Weiter zum AWS Backup und folgen Sie den Schritten zum Löschen von Backups unter Backups löschen.

> **Note**
>
> Wenn Sie ein inkrementelles Backup löschen, wird nur das inkrementelle Backup gelöscht und das zugrunde liegende vollständige Backup wird nicht gelöscht.

Kontingent und Grenzwerte

AWS Backup beschränkt die Backups auf ein gleichzeitiges Backup pro Ressource. Daher werden zusätzliche geplante oder auf Abruf benötigte Backup-Anfragen für die Ressource in die Warteschlange gestellt und erst gestartet, nachdem der bestehende Backup-Job abgeschlossen ist. Wenn der Backup-Job nicht innerhalb des Backup-Fensters gestartet oder abgeschlossen wird, schlägt die Anforderung fehl. Weitere Informationen zu AWS Backup Limits finden Sie unter AWS Backup Limits im AWS Backup Developer Guide.

Wenn Sie ein Backup erstellen, können Sie bis zu vier Backups gleichzeitig pro Konto ausführen. In ähnlicher Weise können Sie pro Konto eine gleichzeitige Wiederherstellung ausführen. Wenn Sie mehr als vier Backup-Jobs gleichzeitig initiieren, werden nur vier Backup-Jobs initiiert, und die verbleibenden Jobs werden regelmäßig wiederholt. Wenn der Backup-Job nach der Initiierung nicht innerhalb der konfigurierten Dauer des Backup-Fensters abgeschlossen wird, schlägt der Backup-Job fehl. Wenn es sich bei dem fehlgeschlagenen Backup-Job um ein On-Demand-Backup handelt, können Sie das Backup erneut versuchen. Bei geplanten Backups wird der Job nach dem folgenden Zeitplan ausgeführt.

Kundendefinierte Partitionsschlüssel

Amazon Timestream für LiveAnalytics kundenspezifische Partitionsschlüssel ist eine Funktion in Timestream für LiveAnalytics , mit der Kunden ihre eigenen Partitionsschlüssel für ihre Tabellen definieren können. Partitionierung ist eine Technik, die verwendet wird, um Daten auf mehrere physische Speichereinheiten zu verteilen und so einen schnelleren und effizienteren Datenabruf zu ermöglichen. Mit kundenspezifischen Partitionsschlüsseln können Kunden ein Partitionierungsschema erstellen, das besser zu ihren Abfragemustern und Anwendungsfällen passt.

Mit Timestream für LiveAnalytics kundenspezifische Partitionsschlüssel können Kunden einen Dimensionsnamen als Partitionsschlüssel für ihre Tabellen wählen. Dies ermöglicht mehr Flexibilität bei der Definition des Partitionierungsschemas für ihre Daten. Durch die Auswahl des richtigen Partitionsschlüssels können Kunden ihr Datenmodell optimieren, ihre Abfrageleistung verbessern und die Abfragelatenz reduzieren.

Themen

- Verwendung von kundenspezifischen Partitionsschlüsseln
- Erste Schritte mit kundendefinierten Partitionsschlüsseln
- Die Konfiguration des Partitionierungsschemas wird überprüft

- Die Konfiguration des Partitionierungsschemas wird aktualisiert
- Vorteile von kundendefinierten Partitionsschlüsseln
- Einschränkungen kundendefinierter Partitionsschlüssel
- Kundendefinierte Partitionsschlüssel und Dimensionen mit geringer Kardinalität
- Partitionsschlüssel für bestehende Tabellen erstellen
- Timestream für die LiveAnalytics Schemavalidierung mit benutzerdefinierten zusammengesetzten Partitionsschlüsseln

Verwendung von kundenspezifischen Partitionsschlüsseln

Wenn Sie über ein klar definiertes Abfragemuster mit hohen Kardinalitätsdimensionen verfügen und eine geringe Abfragelatenz benötigen, kann ein Timestream für LiveAnalytics kundenspezifische Partitionsschlüssel ein nützliches Tool zur Verbesserung Ihres Datenmodells sein. Wenn Sie beispielsweise ein Einzelhandelsunternehmen sind, das Kundeninteraktionen auf Ihrer Website verfolgt, sind die wichtigsten Zugriffsmuster wahrscheinlich nach Kunden-ID und Zeitstempel geordnet. Durch die Definition der Kunden-ID als Partitionsschlüssel können Ihre Daten gleichmäßig verteilt werden, wodurch die Latenz reduziert und letztendlich die Benutzererfahrung verbessert wird.

Ein anderes Beispiel ist das Gesundheitswesen, wo tragbare Geräte Sensordaten sammeln, um die Vitalparameter von Patienten zu verfolgen. Das Hauptzugriffsmuster würde anhand der Geräte-ID und des Zeitstempels erfolgen, wobei in beiden Dimensionen eine hohe Kardinalität herrschen würde. Durch die Definition der Geräte-ID als Partitionsschlüssel können Sie Ihre Abfrageausführung optimieren und eine nachhaltige langfristige Abfrageleistung sicherstellen.

Zusammenfassend lässt sich sagen, dass Timestream für LiveAnalytics kundenspezifische Partitionsschlüssel am nützlichsten ist, wenn Sie über ein klares Abfragemuster und hohe Kardinalitätsdimensionen verfügen und eine geringe Latenz für Ihre Abfragen benötigen. Durch die Definition eines Partitionsschlüssels, der auf Ihr Abfragemuster abgestimmt ist, können Sie Ihre Abfrageausführung optimieren und eine dauerhafte Leistung der Abfragen auf lange Sicht sicherstellen.

Erste Schritte mit kundendefinierten Partitionsschlüsseln

Wählen Sie in der Konsole Tabellen aus und erstellen Sie eine neue Tabelle. Sie können auch eine verwendenSDK, um auf die `CreateTable` Aktion zuzugreifen, um neue Tabellen zu erstellen, die einen benutzerdefinierten Partitionsschlüssel enthalten können.

Erstellen Sie eine Tabelle mit einem Partitionsschlüssel vom Typ Dimension

Sie können die folgenden Codefragmente verwenden, um eine Tabelle mit einem Partitionsschlüssel vom Typ Dimension zu erstellen.

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)
```

Die Konfiguration des Partitionierungsschemas wird überprüft

Sie können auf verschiedene Arten überprüfen, wie eine Tabellenkonfiguration für das Partitionierungsschema funktioniert. Wählen Sie in der Konsole Datenbanken und wählen Sie die Tabelle aus, die Sie überprüfen möchten. Sie können auch eine verwendenSDK, um auf die `DescribeTable` Aktion zuzugreifen.

Beschreiben Sie eine Tabelle mit einem Partitionsschlüssel

Sie können die folgenden Codefragmente verwenden, um eine Tabelle mit einem Partitionsschlüssel zu beschreiben.

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;
       }
   }
```

Im Folgenden finden Sie eine Beispielausgabe.

1. Die Tabelle hat den Dimensionstyp Partitionsschlüssel

```
[{Type: DIMENSION,Name: hostId,EnforcementInRecord: OPTIONAL}]
```

2. Die Tabelle hat den Typ „Kennzahlname“ und einen Partitionsschlüssel

```
[{Type: MEASURE,}]
```

3. Abrufen des zusammengesetzten Partitionsschlüssels aus einer Tabelle, die ohne Angabe des zusammengesetzten Partitionsschlüssels erstellt wurde

```
[{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;
       }
   }
```

Im Folgenden finden Sie eine Beispielausgabe.

1. Die Tabelle hat den Dimensionstyp Partitionsschlüssel

```
[PartitionKey(Type=DIMENSION, Name=hostId, EnforcementInRecord=OPTIONAL)]
```

2. Die Tabelle hat den Typ „Kennzahlname“ und einen Partitionsschlüssel

```
[PartitionKey(Type=MEASURE)]
```

3. Beim Abrufen des zusammengesetzten Partitionsschlüssels aus einer Tabelle, die ohne Angabe des zusammengesetzten Partitionsschlüssels erstellt wurde, wird zurückgegeben

```
[PartitionKey(Type=MEASURE)]
```

Go v1

```
    <tablistentry>
     <tabname> Go </tabname>
     <tabcontent>
      <programlisting language="go"></programlisting>
     </tabcontent>
    </tablistentry>
```

Im Folgenden finden Sie eine Beispielausgabe.

```
{
  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
   }
```

Im Folgenden finden Sie eine Beispielausgabe.

```
{
  "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)
```

Im Folgenden finden Sie eine Beispielausgabe.

1. Die Tabelle hat den Dimensionstyp Partitionsschlüssel

```
[{'CompositePartitionKey': [{'Type': 'DIMENSION', 'Name': 'hostId',
 'EnforcementInRecord': 'OPTIONAL'}]}]
```

2. Die Tabelle hat den Typ „Kennzahlname“ und einen Partitionsschlüssel

```
[{'CompositePartitionKey': [{'Type': 'MEASURE'}]}]
```

3. Abrufen des zusammengesetzten Partitionsschlüssels aus einer Tabelle, die ohne Angabe des zusammengesetzten Partitionsschlüssels erstellt wurde

```
[{'CompositePartitionKey': [{'Type': 'MEASURE'}]}]
```

Die Konfiguration des Partitionierungsschemas wird aktualisiert

Sie können die Tabellenkonfiguration für das Partitionierungsschema mit der Aktion SDK with access the `UpdateTable` aktualisieren.

Aktualisieren Sie eine Tabelle mit einem Partitionsschlüssel

Sie können die folgenden Codefragmente verwenden, um eine Tabelle mit einem Partitionsschlüssel zu aktualisieren.

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)
```

Vorteile von kundendefinierten Partitionsschlüsseln

Verbesserte Abfrageleistung: Mit kundenspezifischen Partitionsschlüsseln können Sie Ihre Abfrageausführung optimieren und die allgemeine Abfrageleistung verbessern. Durch die Definition von Partitionsschlüsseln, die Ihren Abfragemustern entsprechen, können Sie das Scannen von Daten minimieren und die Datenbereinigung optimieren, was zu einer geringeren Abfragelatenz führt.

Bessere langfristige Vorhersagbarkeit der Leistung: Kundendefinierte Partitionsschlüssel ermöglichen es Kunden, Daten gleichmäßig auf Partitionen zu verteilen, wodurch die Effizienz der Datenverwaltung verbessert wird. Dadurch wird sichergestellt, dass Ihre Abfrageleistung stabil bleibt, auch wenn Ihre gespeicherten Daten im Laufe der Zeit skaliert werden.

Einschränkungen kundendefinierter Partitionsschlüssel

Als Timestream für LiveAnalytics Benutzer ist es wichtig, die Einschränkungen im Zusammenhang mit einem kundenspezifischen Partitionsschlüssel zu berücksichtigen. Erstens erfordert es ein gutes Verständnis Ihrer Arbeitslast und Ihrer Abfragemuster. Das bedeutet, dass Sie eine klare Vorstellung davon haben sollten, welche Dimensionen am häufigsten als Hauptfilterbedingungen in Abfragen verwendet werden, und eine hohe Kardinalität haben sollten, um Partitionsschlüssel am effektivsten zu nutzen.

Zweitens müssen Partitionsschlüssel zum Zeitpunkt der Tabellenerstellung definiert werden und können nicht zu vorhandenen Tabellen hinzugefügt werden. Das bedeutet, dass Sie Ihre Partitionierungsstrategie sorgfältig abwägen sollten, bevor Sie eine Tabelle erstellen, um sicherzustellen, dass sie Ihren Geschäftsanforderungen entspricht.

Schließlich ist es wichtig zu beachten, dass Sie den Partitionsschlüssel nach der Erstellung der Tabelle nicht mehr ändern können. Das bedeutet, dass Sie Ihre Partitionierungsstrategie gründlich testen und bewerten sollten, bevor Sie sich darauf festlegen. Angesichts dieser Einschränkungen kann der vom Kunden definierte Partitionsschlüssel von Timestream die Abfrageleistung und die langfristige Zufriedenheit erheblich verbessern.

Kundendefinierte Partitionsschlüssel und Dimensionen mit geringer Kardinalität

Wenn Sie sich für die Verwendung eines Partitionsschlüssels mit sehr niedriger Kardinalität entscheiden, z. B. für eine bestimmte Region oder ein bestimmtes Bundesland, müssen Sie beachten, dass die Daten für andere Entitäten wie `customerIDProductCategory`, und andere möglicherweise auf zu viele Partitionen verteilt werden, in denen manchmal nur wenige oder gar keine Daten vorhanden sind. Dies kann zu ineffizienter Abfrageausführung und verminderter Leistung führen.

Um dies zu vermeiden, empfehlen wir Ihnen, Dimensionen zu wählen, die nicht nur Teil Ihrer Schlüsselfilterbedingung sind, sondern auch eine höhere Kardinalität aufweisen. Dadurch wird sichergestellt, dass die Daten gleichmäßig auf die Partitionen verteilt werden, und die Abfrageleistung verbessert.

Partitionsschlüssel für bestehende Tabellen erstellen

Wenn Sie bereits Tabellen in Timestream für haben LiveAnalytics und kundenspezifische Partitionsschlüssel verwenden möchten, müssen Sie Ihre Daten in eine neue Tabelle mit der gewünschten Partitionierungsschemadefinition migrieren. Dies kann mithilfe von Export nach S3 und Batch-Load zusammen erfolgen. Dazu müssen die Daten aus der vorhandenen Tabelle nach S3 exportiert, die Daten so geändert werden, dass sie den Partitionsschlüssel enthalten (falls erforderlich) und Ihren CSV Dateien Header hinzugefügt werden. Anschließend werden die Daten in eine neue Tabelle importiert, in der das gewünschte Partitionierungsschema definiert ist. Beachten Sie, dass diese Methode zeitaufwändig und kostspielig sein kann, insbesondere bei großen Tabellen.

Alternativ können Sie geplante Abfragen verwenden, um Ihre Daten in eine neue Tabelle mit dem gewünschten Partitionierungsschema zu migrieren. Bei dieser Methode wird eine geplante Abfrage erstellt, die aus der vorhandenen Tabelle liest und in die neue Tabelle schreibt. Die geplante Abfrage kann so eingerichtet werden, dass sie regelmäßig ausgeführt wird, bis alle Daten migriert wurden. Beachten Sie, dass Ihnen das Lesen und Schreiben der Daten während des Migrationsprozesses in Rechnung gestellt wird.

Timestream für die LiveAnalytics Schemavalidierung mit benutzerdefinierten zusammengesetzten Partitionsschlüsseln

Die Schemavalidierung in Timestream for LiveAnalytics hilft sicherzustellen, dass die in die Datenbank aufgenommenen Daten dem angegebenen Schema entsprechen. Dadurch werden Aufnahmefehler minimiert und die Datenqualität verbessert. Die Schemavalidierung ist besonders nützlich, wenn Sie einen vom Kunden definierten Partitionsschlüssel verwenden, um Ihre Abfrageleistung zu optimieren.

Was ist Timestream für die LiveAnalytics Schemavalidierung mit kundendefinierten Partitionsschlüsseln?

Timestream für die LiveAnalytics Schemavalidierung ist eine Funktion, die Daten, die in eine LiveAnalytics Timestream-For-Tabelle aufgenommen werden, auf der Grundlage eines vordefinierten Schemas validiert. Dieses Schema definiert das Datenmodell, einschließlich Partitionsschlüssel, Datentypen und Einschränkungen für die einzufügenden Datensätze.

Bei Verwendung eines vom Kunden definierten Partitionsschlüssels wird die Schemavalidierung noch wichtiger. Mit Partitionsschlüsseln können Sie einen Partitionsschlüssel angeben, der bestimmt, wie Ihre Daten in Timestream gespeichert werden. LiveAnalytics Indem Sie die eingehenden Daten

anhand des Schemas mit einem benutzerdefinierten Partitionsschlüssel validieren, können Sie die Datenkonsistenz sicherstellen, Fehler frühzeitig erkennen und die Gesamtqualität der in Timestream for gespeicherten Daten verbessern. LiveAnalytics

So verwenden Sie Timestream für die LiveAnalytics Schemavalidierung mit benutzerdefinierten zusammengesetzten Partitionsschlüsseln

Gehen Sie folgendermaßen vor, um Timestream für die LiveAnalytics Schemavalidierung mit benutzerdefinierten zusammengesetzten Partitionsschlüsseln zu verwenden:

Denken Sie darüber nach, wie Ihre Abfragemuster aussehen werden: Um das Schema für Ihren Timestream für die LiveAnalytics Tabelle richtig auszuwählen und zu definieren, sollten Sie mit Ihren Abfrageanforderungen beginnen.

Geben Sie benutzerdefinierte zusammengesetzte Partitionsschlüssel an: Geben Sie beim Erstellen der Tabelle einen benutzerdefinierten Partitionsschlüssel an. Dieser Schlüssel bestimmt das Attribut, das zur Partitionierung der Tabellendaten verwendet wird. Sie können zwischen Dimensionsschlüsseln und Kennzahlschlüsseln für die Partitionierung wählen. Ein Dimensionsschlüssel partitioniert Daten auf der Grundlage eines Dimensionsnamens, während ein Kennzahlschlüssel Daten auf der Grundlage des Kennzahlnamens partitioniert.

Erzwingungsstufen festlegen: Um eine korrekte Datenpartitionierung und die damit verbundenen Vorteile sicherzustellen, LiveAnalytics ermöglicht Ihnen Amazon Timestream for, Durchsetzungsstufen für jeden Partitionsschlüssel in Ihrem Schema festzulegen. Die Durchsetzungsstufe bestimmt, ob die Partitionsschlüsseldimension bei der Erfassung von Datensätzen erforderlich oder optional ist. Sie können zwischen zwei Optionen wählen:`REQUIRED`, was bedeutet, dass der Partitionsschlüssel im aufgenommenen Datensatz vorhanden sein muss, und, was bedeutet`OPTIONAL`, dass der Partitionsschlüssel nicht vorhanden sein muss. Es wird empfohlen, bei der Verwendung einer benutzerdefinierten Partition die `REQUIRED` Erzwingungsstufe zu verwenden, um sicherzustellen, dass Ihre Daten ordnungsgemäß partitioniert sind und Sie alle Vorteile dieser Funktion nutzen können. Darüber hinaus können Sie die Konfiguration der Durchsetzungsstufe jederzeit nach der Schemaerstellung ändern, um sie an Ihre Datenaufnahmeanforderungen anzupassen.

Daten aufnehmen: Bei der Aufnahme von Daten in den Timestream für die LiveAnalytics Tabelle überprüft der Schemavalidierungsprozess die Datensätze anhand des definierten Schemas mit benutzerdefinierten zusammengesetzten Partitionsschlüsseln. Wenn die Datensätze nicht dem Schema entsprechen, gibt Timestream for LiveAnalytics einen Validierungsfehler zurück.

Behandlung von Validierungsfehlern: Im Falle von Validierungsfehlern gibt Timestream for LiveAnalytics je nach Art des `RejectedRecordsException` Fehlers a `ValidationException` oder a zurück. Stellen Sie sicher, dass Sie diese Ausnahmen in Ihrer Anwendung behandeln und entsprechende Maßnahmen ergreifen, z. B. die falschen Datensätze korrigieren und die Aufnahme erneut versuchen.

Erzwingungsstufen aktualisieren: Falls erforderlich, können Sie die Durchsetzungsstufe von Partitionsschlüsseln nach der Tabellenerstellung mithilfe der Aktion aktualisieren. `UpdateTable` Es ist jedoch wichtig zu beachten, dass einige Aspekte der Konfiguration des Partitionsschlüssels, wie Name und Typ, nach der Tabellenerstellung nicht geändert werden können. Wenn Sie die Erzwingungsebene von `REQUIRED` auf ändern`OPTIONAL`, werden alle Datensätze akzeptiert, unabhängig davon, ob das als benutzerdefinierter Partitionsschlüssel ausgewählte Attribut vorhanden ist. Wenn Sie dagegen die Durchsetzungsstufe von `OPTIONAL` auf ändern`REQUIRED`, werden möglicherweise 4xx-Schreibfehler für Datensätze angezeigt, die diese Bedingung nicht erfüllen. Daher ist es wichtig, dass Sie bei der Erstellung Ihrer Tabelle auf der Grundlage der Partitionierungsanforderungen Ihrer Daten die für Ihren Anwendungsfall geeignete Durchsetzungsstufe auswählen.

Wann sollte Timestream für die LiveAnalytics Schemavalidierung mit benutzerdefinierten zusammengesetzten Partitionsschlüsseln verwendet werden

Timestream für die LiveAnalytics Schemavalidierung mit benutzerdefinierten zusammengesetzten Partitionsschlüsseln sollte in Szenarien verwendet werden, in denen Datenkonsistenz, Qualität und optimierte Partitionierung entscheidend sind. Indem Sie bei der Datenaufnahme ein Schema durchsetzen, können Sie Fehler und Inkonsistenzen verhindern, die zu falschen Analysen oder zum Verlust wertvoller Erkenntnisse führen könnten.

Interaktion mit Batch-Load-Jobs

Beim Einrichten eines Batch-Load-Jobs zum Importieren von Daten in eine Tabelle mit einem vom Kunden definierten Partitionsschlüssel gibt es einige Szenarien, die den Prozess beeinflussen könnten:

1. Wenn die Erzwingungsstufe auf eingestellt ist`OPTIONAL`, wird während des Erstellungsvorgangs eine Warnung auf der Konsole angezeigt, falls der Partitionsschlüssel bei der Jobkonfiguration nicht zugeordnet wurde. Diese Warnung wird nicht angezeigt, wenn Sie das API oder CLI verwenden.

2. Wenn die Erzwingungsebene auf eingestellt ist`REQUIRED`, wird die Auftragserstellung abgelehnt, sofern der Partitionsschlüssel nicht einer Quelldatenspalte zugeordnet ist.
3. Wenn die Erzwingungsstufe `REQUIRED` nach der Erstellung des Jobs auf die Erzwingungsstufe geändert wird, wird der Job zwar weiterhin ausgeführt, aber alle Datensätze, die nicht die richtige Zuordnung für den Partitionsschlüssel haben, werden mit einem 4xx-Fehler zurückgewiesen.

Interaktion mit einer geplanten Abfrage

Bei der Einrichtung eines geplanten Abfrageauftrags zur Berechnung und Speicherung von Aggregaten, Rollups und anderen Formen vorverarbeiteter Daten in einer Tabelle mit einem benutzerdefinierten Partitionsschlüssel gibt es einige Szenarien, die den Prozess beeinflussen könnten:

1. Wenn die Erzwingungsstufe auf eingestellt ist`OPTIONAL`, wird eine Warnung angezeigt, wenn der Partitionsschlüssel bei der Jobkonfiguration nicht zugewiesen wurde. Diese Warnung wird nicht angezeigt, wenn Sie die Option API oder CLI verwenden.
2. Wenn die Erzwingungsebene auf eingestellt ist`REQUIRED`, wird die Auftragserstellung abgelehnt, sofern der Partitionsschlüssel nicht einer Quelldatenspalte zugeordnet ist.
3. Wenn die Erzwingungsstufe `REQUIRED` nach der Erstellung des Jobs auf die Erzwingungsstufe geändert wird und die geplanten Abfrageergebnisse die Partitionsschlüsseldimension nicht enthalten, schlagen alle nächsten Iterationen des Jobs fehl.

Hinzufügen von Tags und Etiketten zu Ressourcen

Sie können Amazon Timestream mithilfe von Tags für LiveAnalytics Ressourcen kennzeichnen. Mithilfe von Tags können Sie Ihre Ressourcen auf unterschiedliche Weise kategorisieren, z. B. nach Zweck, Eigentümer, Umgebung oder anderen Kriterien. Tags können Sie bei Folgendem unterstützen:

- Eine Ressource basierend auf den Tags, die Sie ihr zugeordnet haben, schnell zu erkennen.
- Sehen Sie sich AWS Rechnungen an, die nach Schlagwörtern aufgeschlüsselt sind.

Tagging wird von AWS Diensten wie Amazon Elastic Compute Cloud (AmazonEC2), Amazon Simple Storage Service (Amazon S3), Timestream for LiveAnalytics und mehr unterstützt. Effizientes Tagging kann Kosteneinblicke bereistellen, mit denen Sie Berichte für Services erstellen können, die ein spezifisches Tag aufweisen.

Gehen Sie wie folgt vor, um sich mit dem Taggen vertraut zu machen::

1. Machen Sie sich mit den Tagging-Einschränkungen vertraut.
2. Erstellen Sie Tags mithilfe von Tagging-Vorgängen.

Schließlich wird empfohlen, optimale Tagging-Strategien zu befolgen. Weitere Informationen finden Sie unter AWS -Markierungsstrategien.

Tagging-Einschränkungen

Jedes Tag besteht aus einem Schlüssel und einem Wert, die Sie beide selbst definieren können. Beachten Sie die folgenden Einschränkungen:

- Jeder Timestream für eine LiveAnalytics Tabelle kann nur ein Tag mit demselben Schlüssel haben. Wenn Sie versuchen, ein vorhandenes Tag hinzuzufügen, wird der vorhandene Tag-Wert auf den neuen Wert aktualisiert.
- Ein Wert fungiert als Deskriptor innerhalb einer Tag-Kategorie. In Timestream darf LiveAnalytics der Wert nicht leer oder Null sein.
- Bei Tag-Schlüsseln und -Werten muss die Groß- und Kleinschreibung beachtet werden.
- Die maximale Schlüssellänge beträgt 128 Unicode-Zeichen.
- Die maximale Wertlänge beträgt 256 Unicode-Zeichen.
- Namen dürfen Buchstaben, Leerzeichen und Zahlen sowie die folgenden Sonderzeichen enthalten: `+ - = . _ : /`
- Die maximale Anzahl an Tags pro Ressource beträgt 50.
- AWS-zugewiesenen Tagnamen und Werten wird automatisch das `aws:` Präfix zugewiesen, das Sie nicht zuweisen können. AWS-zugewiesene Tagnamen werden nicht auf das Tag-Limit von 50 angerechnet. Von Benutzern zugewiesene Tag-Namen haben das Präfix `user:` im Kostenzuordnungsbericht.
- Sie können die Anwendung eines Tags nicht rückdatieren.

Tagging-Operationen

Sie können Tags für Datenbanken und Tabellen hinzufügen, auflisten, bearbeiten oder löschen, indem Sie Amazon Timestream für LiveAnalytics Konsole, Abfragesprache oder AWS Command Line Interface (AWS CLI) verwenden.

Themen

Hinzufügen von Tags zu neuen oder vorhandenen Datenbanken und Tabellen mithilfe der Konsole

Sie können Timestream for LiveAnalytics console verwenden, um Tags zu neuen Datenbanken, Tabellen und geplanten Abfragen hinzuzufügen, wenn Sie diese erstellen. Sie können auch Tags für bestehende Tabellen hinzufügen, bearbeiten oder löschen.

Um Datenbanken bei ihrer Erstellung zu taggen (Konsole)

1. Öffnen Sie die Timestream-Konsole unter https://console.aws.amazon.com/timestream.
2. Wählen Sie im Navigationsbereich Datenbanken und dann Datenbank erstellen aus.
3. Geben Sie auf der Seite Datenbank erstellen einen Namen für die Datenbank ein. Geben Sie einen Schlüssel und einen Wert für das Tag ein und wählen Sie dann Neues Tag hinzufügen.
4. Wählen Sie Datenbank erstellen aus.

Um Tabellen bei ihrer Erstellung zu taggen (Konsole)

1. Öffnen Sie die Timestream-Konsole unter https://console.aws.amazon.com/timestream.
2. Wählen Sie im Navigationsbereich Tables (Tabellen) und anschließend Create table (Tabelle erstellen) aus.
3. Geben Sie auf der Seite „Timestream für LiveAnalytics Tabelle erstellen“ einen Namen für die Tabelle ein. Geben Sie einen Schlüssel und einen Wert für das Tag ein und wählen Sie Neues Tag hinzufügen aus.
4. Wählen Sie Create table (Tabelle erstellen) aus.

Um geplante Abfragen bei deren Erstellung zu taggen (Konsole)

1. Öffnen Sie die Timestream-Konsole unter https://console.aws.amazon.com/timestream.
2. Wählen Sie im Navigationsbereich Geplante Abfragen und dann Geplante Abfrage erstellen aus.
3. Im Schritt 3. Wählen Sie auf der Seite „Abfrageeinstellungen konfigurieren“ die Option Neues Tag hinzufügen aus. Geben Sie einen Schlüssel und Wert für den Tag ein. Wählen Sie Neues Tag hinzufügen, um weitere Tags hinzuzufügen.

4. Wählen Sie Weiter.

Markieren vorhandener Ressourcen (Konsole)

1. Öffnen Sie die Timestream-Konsole unter https://console.aws.amazon.com/timestream.
2. Wählen Sie im Navigationsbereich Datenbanken, Tabellen oder Geplante Abfragen aus.
3. Wählen Sie eine Datenbank oder Tabelle in der Liste aus. Wählen Sie dann Tags verwalten, um Ihre Tags hinzuzufügen, zu bearbeiten oder zu löschen.

Informationen zur Tag-Struktur finden Sie unter Tagging-Einschränkungen.

Sicherheit in Timestream für LiveAnalytics

Cloud-Sicherheit hat AWS höchste Priorität. Als AWS Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame Verantwortung von Ihnen AWS und Ihnen. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, die AWS Dienste in der AWS Cloud ausführt. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Die Wirksamkeit unserer Sicherheitsfunktionen wird regelmäßig von externen Prüfern im Rahmen des AWS -Compliance-Programms getestet und überprüft. Weitere Informationen zu den Compliance-Programmen, die für Timestream gelten LiveAnalytics, finden Sie unter AWS Services im Umfang nach Compliance-Programmen.
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS Dienst, den Sie nutzen. In Ihre Verantwortung fallen außerdem weitere Faktoren, wie z. B. die Vertraulichkeit der Daten, die Anforderungen Ihrer Organisation sowie geltende Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung anwenden können, wenn Sie Timestream for LiveAnalytics verwenden. In den folgenden Themen erfahren Sie, wie Sie Timestream konfigurieren LiveAnalytics , um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Sie erfahren auch, wie Sie andere AWS Dienste nutzen können, die Ihnen bei der Überwachung und Sicherung Ihrer Timestream-Ressourcen helfen können. LiveAnalytics

Themen

Datenschutz in Timestream für LiveAnalytics

Das AWS Modell der gilt für den Datenschutz in Amazon Timestream Live Analytics. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der die AWS Cloud gesamte Infrastruktur läuft. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie im Abschnitt Datenschutz FAQ. Informationen zum Datenschutz in Europa finden Sie im AWS Shared Responsibility Model und im GDPR Blogbeitrag im AWS Security Blog.

Aus Datenschutzgründen empfehlen wir, dass Sie Ihre AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto eine Multi-Faktor-Authentifizierung (MFA).
- Verwenden SieSSL/TLS, um mit AWS Ressourcen zu kommunizieren. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Einrichtung API und Protokollierung von Benutzeraktivitäten mit AWS CloudTrail. Informationen zur Verwendung von CloudTrail Pfaden zur Erfassung von AWS Aktivitäten finden Sie unter Arbeiten mit CloudTrail Pfaden im AWS CloudTrail Benutzerhandbuch.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.

- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie FIPS 140-3 validierte kryptografische Module für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine benötigenAPI, verwenden Sie einen Endpunkt. FIPS Weitere Informationen zu den verfügbaren FIPS Endpunkten finden Sie unter Federal Information Processing Standard () 140-3. FIPS

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit Timestream Live Analytics oder anderen Geräten AWS-Services über die Konsole arbeiten,, API oder. AWS CLI AWS SDKs Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie einem externen Server eine URL zur Verfügung stellen, empfehlen wir dringend, dass Sie keine Anmeldeinformationen angeben, URL um Ihre Anfrage an diesen Server zu validieren.

Für detailliertere Informationen zu Timestream zu LiveAnalytics Datenschutzthemen wie Verschlüsselung im Ruhezustand und Schlüsselverwaltung wählen Sie eines der folgenden verfügbaren Themen aus.

Themen

- Verschlüsselung im Ruhezustand
- Verschlüsselung während der Übertragung
- Schlüsselverwaltung

Verschlüsselung im Ruhezustand

Timestream for LiveAnalytics Encryption at Rest bietet verbesserte Sicherheit, indem alle Ihre Daten im Ruhezustand mit den in AWS Key Management Service () gespeicherten Verschlüsselungsschlüsseln verschlüsselt werden.AWS KMS Diese Funktionalität trägt zur Verringerung des Betriebsaufwands und der Komplexität bei, die mit dem Schutz sensibler Daten einhergeht. Mit der Verschlüsselung ruhender Daten können Sie sicherheitsrelevante Anwendungen erstellen, die eine strenge Einhaltung der Verschlüsselungsvorschriften und der gesetzlichen Bestimmungen erfordern.

- Die Verschlüsselung ist in Ihrer LiveAnalytics Timestream-Datenbank standardmäßig aktiviert und kann nicht deaktiviert werden. Der Industriestandard-Verschlüsselungsalgorithmus AES -256 ist der verwendete Standardverschlüsselungsalgorithmus.
- AWS KMS ist für die Verschlüsselung im Ruhezustand in Timestream for erforderlich. LiveAnalytics
- Sie können nicht nur eine Teilmenge von Elementen in einer Tabelle verschlüsseln.
- Sie müssen Ihre Datenbank-Client-Anwendungen nicht ändern, um Verschlüsselung anzuwenden.

Wenn Sie keinen Schlüssel angeben, LiveAnalytics erstellt und verwendet Timestream for einen AWS KMS Schlüssel, der `alias/aws/timestream` in Ihrem Konto angegeben ist.

Sie können Ihren eigenen, vom Kunden verwalteten Schlüssel verwendenKMS, um Ihren Timestream für Daten zu verschlüsseln. LiveAnalytics Weitere Informationen zu Schlüsseln in Timestream für LiveAnalytics finden Sie unter. Schlüsselverwaltung

Timestream for LiveAnalytics speichert Ihre Daten in zwei Speicherebenen: Speicherspeicher und Magnetspeicher. Speicherdaten werden mit einem Timestream als LiveAnalytics Serviceschlüssel verschlüsselt. Magnetische Speicherdaten werden mit Ihrem AWS KMS Schlüssel verschlüsselt.

Der Timestream Query-Dienst benötigt Anmeldeinformationen, um auf Ihre Daten zugreifen zu können. Diese Anmeldeinformationen werden mit Ihrem KMS Schlüssel verschlüsselt.

> **Note**
>
> Timestream for ruft LiveAnalytics nicht AWS KMS für jeden Decrypt-Vorgang auf. Stattdessen wird bei aktivem Datenverkehr für 5 Minuten ein lokaler Schlüsselcache verwaltet. Alle Berechtigungsänderungen werden über den Timestream für das LiveAnalytics System übertragen, wobei die Konsistenz innerhalb von höchstens 5 Minuten erreicht werden kann.

Verschlüsselung während der Übertragung

Alle Ihre Timestream Live Analytics-Daten werden bei der Übertragung verschlüsselt. Standardmäßig ist die gesamte Kommunikation zu und von Timestream for durch LiveAnalytics die Transport Layer Security (TLS) -Verschlüsselung geschützt.

Schlüsselverwaltung

Sie können Schlüssel für Amazon Timestream Live Analytics mit dem AWS Key Management Service (AWS KMS) verwalten. Timestream Live Analytics erfordert die Verwendung von, KMS um Ihre Daten

zu verschlüsseln. Sie haben die folgenden Optionen für die Schlüsselverwaltung, je nachdem, wie viel Kontrolle Sie über Ihre Schlüssel benötigen:

Datenbank- und Tabellenressourcen

- Von Timestream Live Analytics verwalteter Schlüssel: Wenn Sie keinen Schlüssel angeben, erstellt Timestream Live Analytics einen Schlüssel mithilfe von. `alias/aws/timestream` KMS
- Vom Kunden verwalteter Schlüssel: Vom KMS Kunden verwaltete Schlüssel werden unterstützt. Wählen Sie diese Option, wenn Sie mehr Kontrolle über die Berechtigungen und den Lebenszyklus Ihrer Schlüssel benötigen, einschließlich der Möglichkeit, diese auf jährlicher Basis automatisch wechseln zu lassen.

Ressource für geplante Abfragen

- Timestream Live Analytics-eigener Schlüssel: Wenn Sie keinen Schlüssel angeben, verwendet Timestream Live Analytics seinen eigenen Schlüssel, um die Abfrageressource zu verschlüsseln. Dieser KMS Schlüssel ist im Timestream-Konto vorhanden. Weitere Informationen finden Sie AWS im Entwicklerhandbuch unter Eigene Schlüssel. KMS
- Vom Kunden verwalteter Schlüssel: KMS Vom Kunden verwaltete Schlüssel werden unterstützt. Wählen Sie diese Option, wenn Sie mehr Kontrolle über die Berechtigungen und den Lebenszyklus Ihrer Schlüssel benötigen, einschließlich der Möglichkeit, diese auf jährlicher Basis automatisch wechseln zu lassen.

KMSSchlüssel in einem externen Schlüsselspeicher (XKS) werden nicht unterstützt.

Identitäts- und Zugriffsmanagement für Amazon Timestream für LiveAnalytics

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf Ressourcen sicher zu AWS kontrollieren. IAMAdministratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), Timestream für LiveAnalytics Ressourcen zu verwenden. IAMist eine AWS-Service , die Sie ohne zusätzliche Kosten nutzen können.

Themen

- Zielgruppe
- Authentifizierung mit Identitäten

- Verwalten des Zugriffs mit Richtlinien
- So LiveAnalytics funktioniert Amazon Timestream for mit IAM
- AWS verwaltete Richtlinien für Amazon Timestream Live Analytics
- Amazon Timestream für Beispiele für LiveAnalytics identitätsbasierte Richtlinien
- Fehlerbehebung bei Amazon Timestream für LiveAnalytics Identität und Zugriff

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von der Arbeit ab, für LiveAnalytics die Sie in Timestream arbeiten.

Dienstbenutzer — Wenn Sie Timestream for LiveAnalytics Service für Ihre Arbeit verwenden, stellt Ihnen Ihr Administrator die Anmeldeinformationen und Berechtigungen zur Verfügung, die Sie benötigen. Wenn Sie für Ihre Arbeit mehr LiveAnalytics Funktionen von Timestream for verwenden, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Fuktionsweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anzufordern müssen. Wenn Sie nicht auf eine Funktion in Timestream für zugreifen können LiveAnalytics, finden Sie weitere Informationen unter. Fehlerbehebung bei Amazon Timestream für LiveAnalytics Identität und Zugriff

Serviceadministrator — Wenn Sie in Ihrem Unternehmen für Timestream für LiveAnalytics Ressourcen verantwortlich sind, haben Sie wahrscheinlich vollen Zugriff auf Timestream for. LiveAnalytics Es ist Ihre Aufgabe, zu bestimmen, auf welche LiveAnalytics Timestream-Funktionen und Ressourcen Ihre Servicebenutzer zugreifen sollen. Anschließend müssen Sie Anfragen an Ihren IAM Administrator senden, um die Berechtigungen Ihrer Servicebenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die grundlegenden Konzepte von zu verstehenIAM. Weitere Informationen darüber, wie Ihr Unternehmen Timestream for nutzen IAM kann LiveAnalytics, finden Sie unterSo LiveAnalytics funktioniert Amazon Timestream for mit IAM.

IAMAdministrator — Wenn Sie ein IAM Administrator sind, möchten Sie vielleicht mehr darüber erfahren, wie Sie Richtlinien schreiben können, um den Zugriff auf Timestream für zu verwalten. LiveAnalytics Ein Beispiel für Timestream für LiveAnalytics identitätsbasierte Richtlinien, die Sie in verwenden können, finden Sie unter. IAM Amazon Timestream für Beispiele für LiveAnalytics identitätsbasierte Richtlinien

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen als IAM Benutzer authentifiziert (angemeldet AWS) sein oder eine IAM Rolle übernehmen. Root-Benutzer des AWS-Kontos

Sie können sich AWS als föderierte Identität anmelden, indem Sie Anmeldeinformationen verwenden, die über eine Identitätsquelle bereitgestellt wurden. AWS IAM Identity Center (IAMIdentity Center-) Nutzer, die Single-Sign-On-Authentifizierung Ihres Unternehmens und Ihre Google- oder Facebook-Anmeldeinformationen sind Beispiele für föderierte Identitäten. Wenn Sie sich als föderierte Identität anmelden, hat Ihr Administrator zuvor einen Identitätsverbund mithilfe von Rollen eingerichtet. IAM Wenn Sie AWS mithilfe eines Verbunds darauf zugreifen, übernehmen Sie indirekt eine Rolle.

Je nachdem, welcher Benutzertyp Sie sind, können Sie sich beim AWS Management Console oder beim AWS Zugangsportal anmelden. Weitere Informationen zur Anmeldung finden Sie AWS unter So melden Sie sich bei Ihrem an AWS-Konto im AWS-Anmeldung Benutzerhandbuch.

Wenn Sie AWS programmgesteuert darauf zugreifen, AWS stellt es ein Software Development Kit (SDK) und eine Befehlszeilenschnittstelle (CLI) bereit, mit der Sie Ihre Anfragen mithilfe Ihrer Anmeldeinformationen kryptografisch signieren können. Wenn Sie keine AWS Tools verwenden, müssen Sie Anfragen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode, um Anfragen selbst zu signieren, finden Sie im IAMBenutzerhandbuch unter AWS Signature Version 4 für API Anfragen.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen angeben. AWS Empfiehlt beispielsweise, die Multi-Faktor-Authentifizierung (MFA) zu verwenden, um die Sicherheit Ihres Kontos zu erhöhen. Weitere Informationen finden Sie unter Multi-Faktor-Authentifizierung im AWS IAM Identity Center Benutzerhandbuch und AWS Multi-Faktor-Authentifizierung IAM im IAM Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAMBenutzer ist eine Identität innerhalb von Ihnen AWS-Konto , die über spezifische Berechtigungen für eine einzelne Person oder Anwendung verfügt. Wir empfehlen, sich nach Möglichkeit auf temporäre Anmeldeinformationen zu verlassen, anstatt IAM Benutzer mit langfristigen Anmeldeinformationen wie Passwörtern und Zugriffsschlüsseln zu erstellen. Wenn Sie jedoch spezielle Anwendungsfälle haben, für die langfristige Anmeldeinformationen von IAM Benutzern erforderlich sind, empfehlen wir, die Zugriffsschlüssel abwechselnd zu verwenden. Weitere Informationen finden Sie im Benutzerhandbuch unter Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, für die IAM langfristige Anmeldeinformationen erforderlich sind.

Eine IAMGruppe ist eine Identität, die eine Sammlung von IAM Benutzern spezifiziert. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise eine Gruppe benennen IAMAdminsund dieser Gruppe Berechtigungen zur Verwaltung von IAM Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie im Benutzerhandbuch unter Anwendungsfälle für IAM IAM Benutzer.

IAMRollen

Eine IAMRolle ist eine Identität innerhalb von Ihnen AWS-Konto , für die bestimmte Berechtigungen gelten. Sie ähnelt einem IAM Benutzer, ist jedoch keiner bestimmten Person zugeordnet. Um vorübergehend eine IAM Rolle in der zu übernehmen AWS Management Console, können Sie von einem Benutzer zu einer IAM Rolle (Konsole) wechseln. Sie können eine Rolle übernehmen, indem Sie eine AWS CLI AWS API OR-Operation aufrufen oder eine benutzerdefinierte Operation verwendenURL. Weitere Informationen zu Methoden zur Verwendung von Rollen finden Sie unter Methoden zur Übernahme einer Rolle im IAMBenutzerhandbuch.

IAMRollen mit temporären Anmeldeinformationen sind in den folgenden Situationen nützlich:

- Verbundbenutzerzugriff – Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie im IAMBenutzerhandbuch unter Erstellen einer Rolle für einen externen Identitätsanbieter. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Um zu kontrollieren, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in. IAM Informationen zu Berechtigungssätzen finden Sie unter Berechtigungssätze im AWS IAM Identity Center -Benutzerhandbuch.
- Temporäre IAM Benutzerberechtigungen — Ein IAM Benutzer oder eine Rolle kann eine IAM Rolle übernehmen, um vorübergehend verschiedene Berechtigungen für eine bestimmte Aufgabe zu übernehmen.
- Kontoübergreifender Zugriff — Sie können eine IAM Rolle verwenden, um einer Person (einem vertrauenswürdigen Principal) in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto

zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. Bei einigen können Sie AWS-Services jedoch eine Richtlinie direkt an eine Ressource anhängen (anstatt eine Rolle als Proxy zu verwenden). Informationen zum Unterschied zwischen Rollen und ressourcenbasierten Richtlinien für den kontenübergreifenden Zugriff finden Sie IAMim Benutzerhandbuch unter Kontoübergreifender Ressourcenzugriff. IAM

- Serviceübergreifender Zugriff — Einige AWS-Services verwenden Funktionen in anderen. AWS-Services Wenn Sie beispielsweise in einem Service einen Anruf tätigen, ist es üblich, dass dieser Service Anwendungen in Amazon ausführt EC2 oder Objekte in Amazon S3 speichert. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
  - Zugriffssitzungen weiterleiten (FAS) — Wenn Sie einen IAM Benutzer oder eine Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FASverwendet die Berechtigungen des Prinzipals, der an aufruft AWS-Service, kombiniert mit der Anforderung, Anfragen AWS-Service an nachgelagerte Dienste zu stellen. FASAnfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren Abschluss Interaktionen mit anderen AWS-Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien beim Stellen von FAS Anfragen finden Sie unter Zugriffssitzungen weiterleiten.
  - Servicerolle — Eine Servicerolle ist eine IAMRolle, die ein Dienst übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM Administrator kann eine Servicerolle von innen heraus erstellen, ändern und löschenIAM. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter Erstellen einer Rolle zum Delegieren von Berechtigungen AWS-Service an eine.
  - Dienstbezogene Rolle — Eine dienstverknüpfte Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Servicebezogene Rollen erscheinen in Ihrem Dienst AWS-Konto und gehören dem Dienst. Ein IAM Administrator kann die Berechtigungen für dienstbezogene Rollen anzeigen, aber nicht bearbeiten.
- Auf Amazon ausgeführte Anwendungen EC2 — Sie können eine IAM Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2 Instance ausgeführt werden und AWS API Anfragen stellen AWS CLI . Dies ist dem Speichern von Zugriffsschlüsseln innerhalb der EC2 Instance vorzuziehen. Um einer EC2 Instanz eine AWS Rolle zuzuweisen und sie allen ihren Anwendungen zur Verfügung zu stellen, erstellen Sie ein Instanzprofil, das an die Instanz angehängt ist. Ein Instanzprofil enthält die Rolle und ermöglicht Programmen, die auf der EC2 Instanz ausgeführt werden, temporäre Anmeldeinformationen

abzurufen. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter Verwenden einer IAM Rolle zur Erteilung von Berechtigungen für Anwendungen, die auf EC2 Amazon-Instances ausgeführt werden.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie ist ein Objekt, AWS das, wenn es einer Identität oder Ressource zugeordnet ist, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anfrage stellt. Berechtigungen in den Richtlinien bestimmen, ob die Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden in AWS Form von JSON Dokumenten gespeichert. Weitere Informationen zur Struktur und zum Inhalt von JSON Richtliniendokumenten finden Sie im IAMBenutzerhandbuch unter Überblick über JSON Richtlinien.

Administratoren können mithilfe von AWS JSON Richtlinien festlegen, wer Zugriff auf was hat. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Um Benutzern die Erlaubnis zu erteilen, Aktionen mit den Ressourcen durchzuführen, die sie benötigen, kann ein IAM Administrator IAM Richtlinien erstellen. Der Administrator kann dann die IAM Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen übernehmen.

IAMRichtlinien definieren Berechtigungen für eine Aktion, unabhängig von der Methode, mit der Sie den Vorgang ausführen. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Rolleninformationen aus dem AWS Management Console AWS CLI, dem oder dem abrufen AWS API.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind Dokumente mit JSON Berechtigungsrichtlinien, die Sie an eine Identität anhängen können, z. B. an einen IAM Benutzer, eine Benutzergruppe oder eine Rolle. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen einer identitätsbasierten Richtlinie finden Sie im Benutzerhandbuch unter Definieren benutzerdefinierter IAM Berechtigungen mit vom Kunden verwalteten Richtlinien. IAM

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem System zuordnen können. AWS-Konto Zu den verwalteten Richtlinien gehören AWS verwaltete Richtlinien und vom Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie oder einer Inline-Richtlinie wählen können, finden Sie im IAMBenutzerhandbuch unter Wählen Sie zwischen verwalteten Richtlinien und Inline-Richtlinien.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON Richtliniendokumente, die Sie an eine Ressource anhängen. Beispiele für ressourcenbasierte Richtlinien sind IAM Rollenvertrauensrichtlinien und Amazon S3 S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien nicht IAM in einer ressourcenbasierten Richtlinie verwenden.

Zugriffskontrolllisten () ACLs

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLsähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON Richtliniendokumentformat.

Amazon S3 und AWS WAF Amazon VPC sind Beispiele für Dienste, die Unterstützung bietenACLs. Weitere Informationen finden Sie unter Übersicht über ACLs die Zugriffskontrollliste (ACL) im Amazon Simple Storage Service Developer Guide.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger verbreitete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen — Eine Berechtigungsgrenze ist eine erweiterte Funktion, mit der Sie die maximalen Berechtigungen festlegen, die eine identitätsbasierte Richtlinie einer IAM

Entität (IAMBenutzer oder Rolle) gewähren kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen zu Berechtigungsgrenzen finden Sie im IAMBenutzerhandbuch unter Berechtigungsgrenzen für IAM Entitäten.

- Dienststeuerungsrichtlinien (SCPs) — SCPs sind JSON Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in festlegen AWS Organizations. AWS Organizations ist ein Dienst zur Gruppierung und zentralen Verwaltung mehrerer Geräte AWS-Konten , die Ihrem Unternehmen gehören. Wenn Sie alle Funktionen in einer Organisation aktivieren, können Sie Richtlinien zur Servicesteuerung (SCPs) auf einige oder alle Ihre Konten anwenden. Das SCP schränkt die Berechtigungen für Entitäten in Mitgliedskonten ein, einschließlich der einzelnen Root-Benutzer des AWS-Kontos. Weitere Informationen zu Organizations und SCPs finden Sie unter Richtlinien zur Servicesteuerung im AWS Organizations Benutzerhandbuch.
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter Sitzungsrichtlinien.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie im IAMBenutzerhandbuch unter Bewertungslogik für Richtlinien.

So LiveAnalytics funktioniert Amazon Timestream for mit IAM

Bevor Sie IAM den Zugriff auf Timestream for verwalten LiveAnalytics, sollten Sie sich darüber im Klaren sein, für welche IAM Funktionen Timestream verfügbar ist. LiveAnalytics Einen allgemeinen Überblick darüber, wie Timestream for LiveAnalytics und andere AWS Services funktionierenIAM, finden Sie unter AWS Services That Work with IAM im IAM Benutzerhandbuch.

Themen

Timestream für identitätsbasierte Richtlinien LiveAnalytics

Mit IAM identitätsbasierten Richtlinien können Sie zulässige oder verweigerte Aktionen und Ressourcen sowie die Bedingungen angeben, unter denen Aktionen zugelassen oder verweigert werden. Timestream for LiveAnalytics unterstützt bestimmte Aktionen und Ressourcen sowie Bedingungsschlüssel. Weitere Informationen zu allen Elementen, die Sie in einer JSON Richtlinie verwenden, finden Sie im IAMBenutzerhandbuch unter IAMJSONPolicy Elements Reference.

Aktionen

Administratoren können mithilfe von AWS JSON Richtlinien angeben, wer Zugriff auf was hat. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das `Action` Element einer JSON Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Richtlinienaktionen haben normalerweise denselben Namen wie der zugehörige AWS API Vorgang. Es gibt einige Ausnahmen, z. B. Aktionen, für die nur eine Genehmigung erforderlich ist und für die es keinen entsprechenden Vorgang gibt. API Es gibt auch einige Operationen, die mehrere Aktionen in einer Richtlinie erfordern. Diese zusätzlichen Aktionen werden als abhängige Aktionen bezeichnet.

Schließen Sie Aktionen in eine Richtlinie ein, um Berechtigungen zur Durchführung der zugeordneten Operation zu erteilen.

Sie können die folgenden Aktionen im Aktionselement einer IAM Richtlinienerklärung angeben. Verwenden Sie Richtlinien, um Berechtigungen zur Ausführung eines Vorgangs in zu erteilenAWS. Wenn Sie eine Aktion in einer Richtlinie verwenden, gewähren oder verweigern Sie normalerweise den Zugriff auf die API Operation, den CLI Befehl oder den SQL Befehl mit demselben Namen.

In einigen Fällen steuert eine einzelne Aktion sowohl den Zugriff auf eine API Operation als auch auf einen SQL Befehl. Alternativ erfordern einige Vorgänge mehrere verschiedene Aktionen.

Eine Liste der unterstützten Timestream-Formulare LiveAnalytics `Action` finden Sie in der folgenden Tabelle:

> **Note**
>
> Für alle datenbankspezifischen Daten können Sie eine Datenbank angeben`Actions`, ARN um die Aktion auf eine bestimmte Datenbank zu beschränken.

Aktionen	Beschreibung	Zugriffsebene	Ressourcentypen (*erforderlich)
DescribeEndpoints	Gibt den Timestrea m-Endpunkt zurück, an den nachfolge nde Anfragen gestellt werden müssen.	Alle	*
Select	Führt Abfragen für Timestream aus, die Daten aus einer oder mehreren Tabellen auswählen. Eine ausführliche Erklärung finden Sie in diesem Hinweis	Lesen	Tabelle*
CancelQuery	Stornieren Sie eine Abfrage.	Lesen	*
ListTables	Ruft die Liste der Tabellen ab.	Auflisten	Datenbank*
ListDatabases	Holen Sie sich die Liste der Datenbank en.	Auflisten	*

Aktionen	Beschreibung	Zugriffsebene	Ressourcentypen (*erforderlich)
ListMeasures	Holen Sie sich die Liste der Maßnahmen .	Lesen	Tabelle*
DescribeTable	Holen Sie sich die Beschreibung der Tabelle.	Lesen	Tabelle*
DescribeDatabase	Holen Sie sich die Datenbankbeschreib ung.	Lesen	Datenbank*
SelectValues	Führen Sie Abfragen aus, für die keine bestimmte Ressource angegeben werden muss. Eine ausführli che Erklärung finden Sie in diesem Hinweis.	Lesen	*
WriteRecords	Fügen Sie Daten in Timestream ein.	Schreiben	Tabelle*
CreateTable	Erstellen Sie eine - Tabelle.	Schreiben	Datenbank*
CreateDatabase	Erstellen Sie eine Datenbank.	Schreiben	*
DeleteDatabase	Löscht eine Datenbank.	Schreiben	*
UpdateDatabase	Eine Datenbank aktualisieren.	Schreiben	*

Aktionen	Beschreibung	Zugriffsebene	Ressourcentypen (*erforderlich)
DeleteTable	Löscht eine Tabelle.	Schreiben	Datenbank*
UpdateTable	Aktualisieren Sie eine Tabelle.	Schreiben	Datenbank*

SelectValues im Vergleich zu wählen:

`SelectValues`ist ein`Action`, das für Abfragen verwendet wird, die keine Ressource benötigen. Ein Beispiel für eine Abfrage, die keine Ressource benötigt, lautet wie folgt:

```
SELECT 1
```

Beachten Sie, dass sich diese Abfrage nicht auf einen bestimmten Timestream für eine LiveAnalytics Ressource bezieht. Betrachten Sie ein anderes Beispiel:

```
SELECT now()
```

Diese Abfrage gibt mithilfe der `now()` Funktion den aktuellen Zeitstempel zurück, erfordert jedoch keine Angabe einer Ressource. `SelectValues`wird häufig zum Testen verwendet, sodass Timestream for Abfragen ohne LiveAnalytics Ressourcen ausführen kann. Stellen Sie sich nun eine `Select` Abfrage vor:

```
SELECT * FROM database.table
```

Für diesen Abfragetyp ist eine Ressource erforderlich, insbesondere ein Timestream für LiveAnalytics `table`, damit die angegebenen Daten aus der Tabelle abgerufen werden können.

Ressourcen

Administratoren können mithilfe von AWS JSON Richtlinien angeben, wer Zugriff auf was hat. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das `Resource` JSON Richtlinienelement gibt das Objekt oder die Objekte an, für die die Aktion gilt. Anweisungen müssen entweder ein – `Resource`oder ein `NotResource`-Element enthalten. Es hat

sich bewährt, eine Ressource mit ihrem Amazon-Ressourcennamen (ARN) anzugeben. Sie können dies für Aktionen tun, die einen bestimmten Ressourcentyp unterstützen, der als Berechtigungen auf Ressourcenebene bezeichnet wird.

Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, z. B. Auflistungsoperationen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

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

In Timestream können LiveAnalytics Datenbanken und Tabellen im `Resource` Element IAM Berechtigungen verwendet werden.

Der Timestream für die LiveAnalytics Datenbankressource hat Folgendes: ARN

```
arn:${Partition}:timestream:${Region}:${Account}:database/${DatabaseName}
```

Der Timestream für die LiveAnalytics Tabellenressource hat Folgendes: ARN

```
arn:${Partition}:timestream:${Region}:${Account}:database/${DatabaseName}/table/
${TableName}
```

Weitere Informationen zum Format von ARNs finden Sie unter Amazon Resource Names (ARNs) und AWS Service Namespaces.

Um beispielsweise den `database` Schlüsselraum in Ihrer Anweisung anzugeben, verwenden Sie Folgendes: ARN

```
"Resource": "arn:aws:timestream:us-east-1:123456789012:database/mydatabase"
```

Um alle Datenbanken anzugeben, die zu einem bestimmten Konto gehören, verwenden Sie den Platzhalter (*):

```
"Resource": "arn:aws:timestream:us-east-1:123456789012:database/*"
```

Manche LiveAnalytics Timestream-Aktionen, z. B. solche zum Erstellen von Ressourcen, können für eine bestimmte Ressource nicht ausgeführt werden. In diesen Fällen müssen Sie den Platzhalter (*) verwenden.

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

Bedingungsschlüssel

Timestream for LiveAnalytics stellt keine dienstspezifischen Bedingungsschlüssel bereit, unterstützt jedoch die Verwendung einiger globaler Bedingungsschlüssel. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter AWS Globale Bedingungskontextschlüssel im IAMBenutzerhandbuch.

Beispiele

Beispiele für Timestream für LiveAnalytics identitätsbasierte Richtlinien finden Sie unter. Amazon Timestream für Beispiele für LiveAnalytics identitätsbasierte Richtlinien

Timestream für ressourcenbasierte Richtlinien LiveAnalytics

Timestream for unterstützt LiveAnalytics keine ressourcenbasierten Richtlinien. Ein Beispiel für eine detaillierte Seite zu ressourcenbasierten Richtlinien finden Sie unter https://docs.aws.amazon.com/lambda/latest/dg/access-control-resource-based.html.

Autorisierung basiert auf Timestream für Tags LiveAnalytics

Sie können den Zugriff auf Ihren Timestream für LiveAnalytics Ressourcen mithilfe von Tags verwalten. Um den Ressourcenzugriff auf der Grundlage von Tags zu verwalten, geben Sie Tag-Informationen im Bedingungselement einer Richtlinie mithilfe der `aws:TagKeys` Bedingungsschlüssel `timestream:ResourceTag/`*`key-name`*`aws:RequestTag/`*`key-name`*, oder ein. Weitere Informationen zum Taggen von Timestream für LiveAnalytics Ressourcen finden Sie unter. the section called “Taggen von -Ressourcen”

Ein Beispiel für eine identitätsbasierte Richtlinie zur Einschränkung des Zugriffs auf eine Ressource auf der Grundlage der Markierungen dieser Ressource finden Sie unter Timestream für den LiveAnalytics Ressourcenzugriff auf der Grundlage von Tags.

Timestream für Rollen LiveAnalytics IAM

Eine IAMRolle ist eine Entität in Ihrem AWS Konto, die über bestimmte Berechtigungen verfügt.

Verwenden temporärer Anmeldeinformationen mit Timestream für LiveAnalytics

Sie können temporäre Anmeldeinformationen verwenden, um sich bei Federation anzumelden, eine IAM Rolle zu übernehmen oder eine kontoübergreifende Rolle anzunehmen. Sie erhalten

temporäre Sicherheitsanmeldedaten, indem Sie AWS STS API Operationen wie AssumeRoleoder GetFederationTokenaufrufen.

Service-verknüpfte Rollen

Timestream for unterstützt LiveAnalytics keine dienstbezogenen Rollen.

Servicerollen

Timestream for unterstützt LiveAnalytics keine Servicerollen.

AWS verwaltete Richtlinien für Amazon Timestream Live Analytics

Eine AWS verwaltete Richtlinie ist eine eigenständige Richtlinie, die von erstellt und verwaltet AWS wird. AWS Verwaltete Richtlinien dienen dazu, Berechtigungen für viele gängige Anwendungsfälle bereitzustellen, sodass Sie damit beginnen können, Benutzern, Gruppen und Rollen Berechtigungen zuzuweisen.

Beachten Sie, dass AWS verwaltete Richtlinien für Ihre speziellen Anwendungsfälle möglicherweise keine Berechtigungen mit den geringsten Rechten gewähren, da sie allen AWS Kunden zur Verfügung stehen. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie kundenverwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind.

Sie können die in AWS verwalteten Richtlinien definierten Berechtigungen nicht ändern. Wenn die in einer AWS verwalteten Richtlinie definierten Berechtigungen AWS aktualisiert werden, wirkt sich das Update auf alle Prinzidentitäten (Benutzer, Gruppen und Rollen) aus, denen die Richtlinie zugeordnet ist. AWS aktualisiert eine AWS verwaltete Richtlinie höchstwahrscheinlich, wenn eine neue Richtlinie eingeführt AWS-Service wird oder neue API Operationen für bestehende Dienste verfügbar werden.

Weitere Informationen finden Sie im IAMBenutzerhandbuch unter AWS Verwaltete Richtlinien.

Themen

- AWS verwaltete Richtlinie: AmazonTimestreamReadOnlyAccess
- AWS verwaltete Richtlinie: AmazonTimestreamConsoleFullAccess
- AWS verwaltete Richtlinie: AmazonTimestreamFullAccess
- Timestream Live Analytics aktualisiert AWS verwaltete Richtlinien

AWS verwaltete Richtlinie: AmazonTimestreamReadOnlyAccess

Sie können `AmazonTimestreamReadOnlyAccess` sie Ihren Benutzern, Gruppen und Rollen zuordnen. Die Richtlinie bietet nur Lesezugriff auf Amazon Timestream.

Berechtigungsdetails

Diese Richtlinie umfasst die folgende Berechtigung:

- `Amazon Timestream`— Bietet schreibgeschützten Zugriff auf Amazon Timestream. Diese Richtlinie gewährt auch die Erlaubnis, alle laufenden Abfragen abzubrechen.

Informationen zur JSON Formatierung dieser Richtlinie finden Sie unter AmazonTimestreamReadOnlyAccess.

AWS verwaltete Richtlinie: AmazonTimestreamConsoleFullAccess

Sie können `AmazonTimestreamConsoleFullAccess` sie Ihren Benutzern, Gruppen und Rollen zuordnen.

Die Richtlinie bietet vollen Zugriff auf die Verwaltung von Amazon Timestream mithilfe von. AWS Management Console Diese Richtlinie gewährt auch Berechtigungen für bestimmte AWS KMS Operationen und Operationen zur Verwaltung Ihrer gespeicherten Abfragen.

Berechtigungsdetails

Diese Richtlinie umfasst die folgenden Berechtigungen:

- `Amazon Timestream`— Gewährt Schulleitern vollen Zugriff auf Amazon Timestream.
- `AWS KMS`— Ermöglicht Prinzipalen, Aliase aufzulisten und Schlüssel zu beschreiben.
- `Amazon S3`— Ermöglicht Principals, alle Amazon S3 S3-Buckets aufzulisten.
- `Amazon SNS`— Ermöglicht Principals, SNS Amazon-Themen aufzulisten.
- `IAM`— Ermöglicht Prinzipalen das Auflisten IAM von Rollen.
- `DBQMS` – Ermöglicht es Prinzipalen, auf Abfragen zuzugreifen, Abfragen zu erstellen, zu löschen, zu beschreiben und zu aktualisieren. Der Database Query Metadata Service (dbqms) ist ein reiner interner Dienst. Es stellt Ihre letzten und gespeicherten Abfragen für den Abfrage-Editor AWS Management Console für mehrere AWS-Services, einschließlich Amazon Timestream, bereit.

Informationen zu dieser Richtlinie im JSON Format finden Sie unter AmazonTimestreamConsoleFullAccess.

AWS verwaltete Richtlinie: AmazonTimestreamFullAccess

Sie können `AmazonTimestreamFullAccess` sie Ihren Benutzern, Gruppen und Rollen zuordnen.

Die Richtlinie bietet vollen Zugriff auf Amazon Timestream. Diese Richtlinie gewährt auch Genehmigungen für bestimmte AWS KMS Operationen.

Berechtigungsdetails

Diese Richtlinie umfasst die folgenden Berechtigungen:

- `Amazon Timestream`— Gewährt Schulleitern vollen Zugriff auf Amazon Timestream.
- `AWS KMS`— Ermöglicht Prinzipalen, Aliase aufzulisten und Schlüssel zu beschreiben.
- `Amazon S3`— Ermöglicht Principals, alle Amazon S3 S3-Buckets aufzulisten.

Informationen zu dieser Richtlinie im JSON Format finden Sie unter. AmazonTimestreamFullAccess

Timestream Live Analytics aktualisiert AWS verwaltete Richtlinien

Sehen Sie sich Details zu Aktualisierungen der AWS verwalteten Richtlinien für Timestream Live Analytics an, seit dieser Dienst begonnen hat, diese Änderungen zu verfolgen. Abonnieren Sie den RSS Feed auf der Timestream Live Analytics-Dokumentverlaufsseite, um automatische Benachrichtigungen über Änderungen an dieser Seite zu erhalten.

Änderung	Beschreibung	Datum
AmazonTimestreamReadOnlyAccess – Aktualisierung auf eine bestehende Richtlinie	Die `timestream:DescribeAccountSettings` Aktion wurde der vorhandenen `AmazonTimestreamReadOnlyAccess` verwalteten Richtlinie hinzugefügt. Diese Aktion wird zur	03. Juni 2024

Änderung	Beschreibung	Datum
	Beschreibung von AWS-Konto Einstellungen verwendet. Timestream Live Analytics hat diese verwaltete Richtlini e ebenfalls aktualisiert, indem ein `Sid` Feld hinzugefügt wurde. Die Aktualisierung der Richtlini e hat keine Auswirkungen auf die Verwendung der `AmazonTimestreamRe adOnlyAccess` verwaltet en Richtlinie.	
AmazonTimestreamRe adOnlyAccess – Aktualisi erung auf eine bestehende Richtlinie	Die `timestream:ListBat chLoadTasks` Aktionen `timestream:Describ eBatchLoadTask` und wurden zur vorhandenen `AmazonTimestreamRe adOnlyAccess` verwaltet en Richtlinie hinzugefügt. Diese Aktionen werden beim Auflisten und Beschreiben von Batch-Load-Aufgaben verwendet. Die Aktualisierung der Richtlini e hat keine Auswirkungen auf die Verwendung der `AmazonTimestreamRe adOnlyAccess` verwaltet en Richtlinie.	24. Februar 2023

Änderung	Beschreibung	Datum
AmazonTimestreamReadOnlyAccess – Aktualisierung auf eine bestehende Richtlinie	Die `timestream:ListScheduledQueries` Aktionen `timestream:DescribeScheduledQuery` und wurden zur vorhandenen `AmazonTimestreamReadOnlyAccess` verwalteten Richtlinie hinzugefügt. Diese Aktionen werden verwendet, um bestehende geplante Abfragen aufzulisten und zu beschreiben. Die Aktualisierung der Richtlinie hat keine Auswirkungen auf die Verwendung der `AmazonTimestreamReadOnlyAccess` verwalteten Richtlinie.	29. November 2021

Änderung	Beschreibung	Datum
AmazonTimestreamConsoleFullAccess – Aktualisierung auf eine bestehende Richtlinie	Die `s3:ListAllMyBuckets` Aktion wurde der vorhandenen `AmazonTimestreamConsoleFullAccess` verwalteten Richtlinie hinzugefügt. Diese Aktion wird verwendet, wenn Sie einen Amazon S3 S3-Bucket für Timestream angeben, um Magnetspeicher-Schreibfehler zu protokollieren. Die Aktualisierung der Richtlinie hat keine Auswirkungen auf die Verwendung der `AmazonTimestreamConsoleFullAccess` verwalteten Richtlinie.	29. November 2021

Änderung	Beschreibung	Datum
AmazonTimestreamFullAccess – Aktualisierung auf eine bestehende Richtlinie	Die `s3:ListAllMyBuckets` Aktion wurde der vorhandenen `AmazonTimestreamFullAccess` verwalteten Richtlinie hinzugefügt. Diese Aktion wird verwendet, wenn Sie einen Amazon S3 S3-Bucket für Timestream angeben, um Magnetspeicher-Schreibfehler zu protokollieren. Die Aktualisierung der Richtlinie hat keine Auswirkungen auf die Verwendung der `AmazonTimestreamFullAccess` verwalteten Richtlinie.	29. November 2021

Änderung	Beschreibung	Datum
AmazonTimestreamConsoleFullAccess – Aktualisierung auf eine bestehende Richtlinie	Redundante Aktionen wurden aus der vorhandenen `AmazonTimestreamConsoleFullAccess` verwalteten Richtlinie entfernt. Bisher umfasste diese Richtlinie eine redundante Aktion `dbqms:DescribeQueryHistory`. Die aktualisierte Richtlinie entfernt die redundante Aktion. Die Aktualisierung der Richtlinie hat keine Auswirkungen auf die Verwendung der `AmazonTimestreamConsoleFullAccess` verwalteten Richtlinie.	23. April 2021
Timestream Live Analytics hat mit der Nachverfolgung von Änderungen begonnen	Timestream Live Analytics begann, Änderungen an den AWS verwalteten Richtlinien nachzuverfolgen.	21. April 2021

Amazon Timestream für Beispiele für LiveAnalytics identitätsbasierte Richtlinien

Standardmäßig sind IAM Benutzer und Rollen nicht berechtigt, Timestream für LiveAnalytics Ressourcen zu erstellen oder zu ändern. Sie können auch keine Aufgaben mit dem AWS Management Console, CQLSH AWS CLI, oder AWS API ausführen. Ein IAM Administrator muss IAM Richtlinien erstellen, die Benutzern und Rollen die Berechtigung gewähren, bestimmte API Operationen mit den angegebenen Ressourcen auszuführen, die sie benötigen. Der Administrator muss diese Richtlinien dann den IAM Benutzern oder Gruppen zuordnen, für die diese Berechtigungen erforderlich sind.

Informationen zum Erstellen einer IAM identitätsbasierten Richtlinie anhand dieser JSON Beispieldokumente finden Sie unter Richtlinien auf der JSON Registerkarte erstellen im IAMBenutzerhandbuch.

Themen

Bewährte Methoden für Richtlinien

Identitätsbasierte Richtlinien legen fest, ob jemand Timestream für LiveAnalytics Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Befolgen Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Anleitungen und Empfehlungen:

- Beginnen Sie mit AWS verwalteten Richtlinien und wechseln Sie zu Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um damit zu beginnen, Ihren Benutzern und Workloads Berechtigungen zu gewähren. Sie sind in Ihrem verfügbar. AWS-Konto Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie AWS im IAMBenutzerhandbuch unter AWS Verwaltete Richtlinien oder Verwaltete Richtlinien für Jobfunktionen.
- Berechtigungen mit den geringsten Rechten anwenden — Wenn Sie Berechtigungen mit IAM Richtlinien festlegen, gewähren Sie nur die Berechtigungen, die für die Ausführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung IAM zum Anwenden von Berechtigungen finden Sie IAMim Benutzerhandbuch unter Richtlinien und Berechtigungen. IAM
- Verwenden Sie Bedingungen in IAM Richtlinien, um den Zugriff weiter einzuschränken — Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen einzuschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um

anzugeben, dass alle Anfragen mit gesendet werden müssenSSL. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese über einen bestimmten Zweck verwendet werden AWS-Service, z. AWS CloudFormation B. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter IAMJSONRichtlinienelemente: Bedingung.

- Verwenden Sie IAM Access Analyzer, um Ihre IAM Richtlinien zu validieren, um sichere und funktionale Berechtigungen zu gewährleisten. IAM Access Analyzer validiert neue und bestehende Richtlinien, sodass die Richtlinien der IAM Richtliniensprache (JSON) und den IAM bewährten Methoden entsprechen. IAMAccess Analyzer bietet mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen, um Sie bei der Erstellung sicherer und funktionaler Richtlinien zu unterstützen. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter Überprüfen von Richtlinien mit IAM Access Analyzer.
- Multi-Faktor-Authentifizierung erforderlich (MFA) — Wenn Sie ein Szenario haben, in dem IAM Benutzer oder ein Root-Benutzer erforderlich sind AWS-Konto, aktivieren Sie die Option MFA für zusätzliche Sicherheit. Wenn Sie festlegen möchten, MFA wann API Operationen aufgerufen werden, fügen Sie MFA Bedingungen zu Ihren Richtlinien hinzu. Weitere Informationen finden Sie unter Sicherer API Zugriff mit MFA im IAMBenutzerhandbuch.

Weitere Informationen zu bewährten Methoden finden Sie unter Bewährte Sicherheitsmethoden IAM im IAM Benutzerhandbuch. IAM

Timestream für LiveAnalytics die Konsole verwenden

Timestream for benötigt LiveAnalytics keine speziellen Berechtigungen für den Zugriff auf die Amazon Timestream for LiveAnalytics Console. Sie benötigen mindestens Leseberechtigungen, um den Timestream für LiveAnalytics Ressourcen in Ihrem Konto aufzulisten und einzusehen. AWS Wenn Sie eine identitätsbasierte Richtlinie erstellen, die restriktiver ist als die erforderlichen Mindestberechtigungen, funktioniert die Konsole für Entitäten (IAMBenutzer oder Rollen) mit dieser Richtlinie nicht wie vorgesehen.

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die es IAM Benutzern ermöglicht, die Inline- und verwalteten Richtlinien einzusehen, die mit ihrer Benutzeridentität verknüpft sind. Diese Richtlinie umfasst Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder programmgesteuert mithilfe von oder. 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": "*"
        }
    ]
}
```

Allgemeine Operationen in Timestream für LiveAnalytics

Im Folgenden finden Sie IAM Beispielrichtlinien, die allgemeine Operationen im Timestream for LiveAnalytics Service ermöglichen.

Themen

- Zulassen aller Operationen
- Zulassen SELECT von Vorgängen
- Zulassen von SELECT Vorgängen auf mehreren Ressourcen
- Metadaten-Operationen zulassen

- Zulassen INSERT von Vorgängen
- CRUDOperationen zulassen
- Stornieren Sie Abfragen und wählen Sie Daten aus, ohne Ressourcen anzugeben
- Eine Datenbank erstellen, beschreiben, löschen und beschreiben
- Beschränken Sie die aufgelisteten Datenbanken nach Tag {"Owner": "${username}"}
- Alle Tabellen in einer Datenbank auflisten
- Eine Tabelle erstellen, beschreiben, löschen, aktualisieren und auswählen
- Beschränken Sie eine Abfrage nach einer Tabelle

Zulassen aller Operationen

Im Folgenden finden Sie ein Beispiel für eine Richtlinie, die alle Operationen in Timestream für LiveAnalytics zulässt.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Zulassen SELECT von Vorgängen

Die folgende Beispielrichtlinie erlaubt Abfragen `SELECT` im -stil für eine bestimmte Ressource.

> **Note**
>
> `<account_ID>`Ersetzen Sie es durch Ihre Amazon-Konto-ID.

```
{
```

```
    "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": "*"
        }
    ]
}
```

Zulassen von SELECT Vorgängen auf mehreren Ressourcen

Die folgende Beispielrichtlinie ermöglicht Abfragen SELECT im -stil für mehrere Ressourcen.

> **Note**
>
> <account_ID>Ersetzen Sie es durch Ihre Amazon-Konto-ID.

```
{
    "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": "*"
        }
    ]
}
```

Metadaten-Operationen zulassen

Die folgende Beispielrichtlinie ermöglicht es dem Benutzer, Metadatenabfragen durchzuführen, erlaubt dem Benutzer jedoch nicht, Vorgänge auszuführen, bei denen tatsächliche Daten in Timestream für LiveAnalytics gelesen oder geschrieben werden.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream:DescribeEndpoints",
                "timestream:DescribeTable",
                "timestream:ListMeasures",
                "timestream:SelectValues",
                "timestream:ListTables",
                "timestream:ListDatabases",
                "timestream:CancelQuery"
            ],
            "Resource": "*"
```

```
        }
    ]
}
```

Zulassen INSERT von Vorgängen

Die folgende Beispielrichtlinie ermöglicht es einem Benutzer, einen `INSERT` Vorgang für sein Konto auszuführen`<account_id>`. `database/sampleDB/table/DevOps`

> **Note**
>
> `<account_ID>`Ersetzen Sie es durch Ihre Amazon-Konto-ID.

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

CRUDOperationen zulassen

Die folgende Beispielrichtlinie ermöglicht es einem Benutzer, CRUD Operationen in Timestream für LiveAnalytics auszuführen.

```
{
    "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": "*"
        }
    ]
}
```

Stornieren Sie Abfragen und wählen Sie Daten aus, ohne Ressourcen anzugeben

Die folgende Beispielrichtlinie ermöglicht es einem Benutzer, Abfragen abzubrechen und Abfragen für Daten durchzuführen`Select`, für die keine Ressourcenspezifikation erforderlich ist:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream:SelectValues",
                "timestream:CancelQuery"
            ],
            "Resource": "*"
        }
    ]
}
```

Eine Datenbank erstellen, beschreiben, löschen und beschreiben

Die folgende Beispielrichtlinie ermöglicht es einem Benutzer, eine Datenbank zu erstellen, zu beschreiben, zu löschen und zu beschreiben`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"
        }
    ]
}
```

Beschränken Sie die aufgelisteten Datenbanken nach Tag **`{"Owner": "${username}"}`**

Die folgende Beispielrichtlinie ermöglicht es einem Benutzer, alle Datenbanken aufzulisten, die mit einem Schlüssel-Wert-Paar gekennzeichnet sind`{"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}"
                }
            }
        }
    ]
```

```
}
```

Alle Tabellen in einer Datenbank auflisten

Die folgende Beispielrichtlinie zum Auflisten aller Tabellen in der Datenbank`sampleDB`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream:ListTables"
            ],
            "Resource": "arn:aws:timestream:us-east-1:<account_ID>:database/sampleDB/"
        }
    ]
}
```

Eine Tabelle erstellen, beschreiben, löschen, aktualisieren und auswählen

Die folgende Beispielrichtlinie ermöglicht es einem Benutzer, Tabellen zu erstellen, Tabellen zu beschreiben, Tabellen zu löschen, Tabellen zu aktualisieren und `Select` Abfragen für Tabellen `DevOps` in der Datenbank durchzuführen`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"
        }
    ]
}
```

Beschränken Sie eine Abfrage nach einer Tabelle

Die folgende Beispielrichtlinie ermöglicht es einem Benutzer, alle Tabellen außer `DevOps` in der Datenbank abzufragen`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 für den LiveAnalytics Ressourcenzugriff auf der Grundlage von Tags

Sie können Bedingungen in Ihrer identitätsbasierten Richtlinie verwenden, um den Zugriff auf Timestream für LiveAnalytics Ressourcen auf der Grundlage von Tags zu steuern. In diesem Abschnitt finden Sie einige Beispiele.

Das folgende Beispiel zeigt, wie Sie eine Richtlinie erstellen können, die einem Benutzer Berechtigungen zum Anzeigen einer Tabelle gewährt, wenn die Tabelle den Wert des Benutzernamens dieses Benutzers `Owner` enthält.

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

Sie können diese Richtlinie den IAM Benutzern in Ihrem Konto zuordnen. Wenn ein benannter Benutzer `richard-roe` versucht, einen Timestream für eine LiveAnalytics Tabelle aufzurufen, muss die Tabelle mit `Owner=richard-roe` oder `owner=richard-roe` gekennzeichnet werden. Andernfalls wird der Zugriff abgelehnt. Der Tag-Schlüssel `Owner` der Bedingung stimmt sowohl mit `Owner` als auch mit `owner` überein, da die Namen von Bedingungsschlüsseln nicht zwischen Groß- und Kleinschreibung unterscheiden. Weitere Informationen finden Sie unter IAMJSONPolicy Elements: Condition im IAMBenutzerhandbuch.

Die folgende Richtlinie gewährt einem Benutzer die Erlaubnis, Tabellen mit Tags zu erstellen, wenn das in der Anfrage übergebene Tag einen Schlüssel `Owner` und einen Wert enthält`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}"
                }
            }
        }
    ]
```

```
}
```

Die folgende Richtlinie ermöglicht die Verwendung von in jeder Datenbank, deren env Tag entweder DescribeDatabase API auf dev oder gesetzt isttest:

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

Diese Richtlinie verwendet einen `Condition` Schlüssel, um zu ermöglichen, dass ein Tag, das den Schlüssel `env` und den Wert `testqa`, oder enthält, `dev` zu einer Ressource hinzugefügt werden kann.

Geplante Abfragen

Auflisten, Löschen, Aktualisieren, Ausführen ScheduledQuery

Die folgende Beispielrichtlinie ermöglicht es einem Benutzer, geplante Abfragen aufzulisten, zu löschen, zu aktualisieren und auszuführen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream:DeleteScheduledQuery",
                "timestream:ExecuteScheduledQuery",
                "timestream:UpdateScheduledQuery",
                "timestream:ListScheduledQueries",
                "timestream:DescribeEndpoints"
            ],
            "Resource": "*"
        }
    ]
}
```

CreateScheduledQuery mithilfe eines vom Kunden verwalteten KMS Schlüssels

Die folgende Beispielrichtlinie ermöglicht es einem Benutzer, eine geplante Abfrage zu erstellen, die mit einem vom Kunden verwalteten KMS Schlüssel verschlüsselt wird; *<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 unter Verwendung eines vom Kunden verwalteten KMS Schlüssels

Die folgende Beispielrichtlinie ermöglicht es einem Benutzer, eine geplante Abfrage zu beschreiben, die mit einem vom Kunden verwalteten KMS Schlüssel erstellt wurde; *<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"
        }
    ]
}
```

Berechtigungen für Ausführungsrollen (Verwendung eines vom Kunden verwalteten KMS Schlüssels für geplante Abfragen und SSE — KMS für Fehlerberichte)

Fügen Sie der im `ScheduledQueryExecutionRoleArn` Parameter angegebenen IAM Rolle die folgende Beispielrichtlinie bei, `CreateScheduledQuery` API die den vom Kunden verwalteten KMS Schlüssel für die Verschlüsselung von geplanten Abfragen und die `SSE-KMS` Verschlüsselung für Fehlerberichte verwendet.

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

Vertrauensverhältnis zwischen der Ausführungsrolle

Das Folgende ist die Vertrauensbeziehung für die IAM Rolle, die im `ScheduledQueryExecutionRoleArn` Parameter von angegeben ist `CreateScheduledQuery`API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "timestream.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Erlaubt den Zugriff auf alle geplanten Abfragen, die innerhalb eines Kontos erstellt wurden

Fügen Sie der im `ScheduledQueryExecutionRoleArn` Parameter von angegebenen IAM Rolle die folgende Beispielrichtlinie hinzu `CreateScheduledQuery`API, um den Zugriff auf alle geplanten Abfragen zu ermöglichen, die innerhalb eines Kontos erstellt wurden *`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/*"
```

```
                }
            }
        }
    ]
}
```

Erlaubt den Zugriff auf alle geplanten Abfragen mit einem bestimmten Namen

Fügen Sie der im `ScheduledQueryExecutionRoleArn` Parameter von angegebenen IAM Rolle die folgende Beispielrichtlinie hinzu `CreateScheduledQuery`API, um den Zugriff auf alle geplanten Abfragen zu ermöglichen, deren Name mit beginnt *`Scheduled_Query_Name`*, innerhalb des Kontos *`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*"
                }
            }
        }
    ]
}
```

Fehlerbehebung bei Amazon Timestream für LiveAnalytics Identität und Zugriff

Verwenden Sie die folgenden Informationen, um häufig auftretende Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit Timestream für LiveAnalytics und auftreten können. IAM

Themen

- Ich bin nicht berechtigt, eine Aktion in Timestream durchzuführen für LiveAnalytics

- Ich bin nicht berechtigt, iam auszuführen: PassRole
- Ich möchte Personen außerhalb meines AWS Kontos den Zugriff auf meinen Timestream für LiveAnalytics Ressourcen ermöglichen

Ich bin nicht berechtigt, eine Aktion in Timestream durchzuführen für LiveAnalytics

Wenn Ihnen AWS Management Console mitgeteilt wird, dass Sie nicht berechtigt sind, eine Aktion auszuführen, müssen Sie sich an Ihren Administrator wenden, um Unterstützung zu erhalten. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Der folgende Beispielfehler tritt auf, wenn der `mateojackson` IAM Benutzer versucht, die Konsole zu verwenden, um Details zu einem anzuzeigen *`table`* hat aber keine `timestream:`*`Select`* Berechtigungen für die Tabelle.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 timestream:Select on resource: mytable
```

In diesem Fall bittet Mateo seinen Administrator um die Aktualisierung seiner Richtlinien, um unter Verwendung der Aktion *`mytable`* auf die Ressource `timestream:`*`Select`* zugreifen zu können.

Ich bin nicht berechtigt, iam auszuführen: PassRole

Wenn Sie eine Fehlermeldung erhalten, dass Sie nicht berechtigt sind, die `iam:PassRole` Aktion durchzuführen, müssen Ihre Richtlinien aktualisiert werden, damit Sie eine Rolle an Timestream übergeben können. LiveAnalytics

Einige AWS-Services ermöglichen es Ihnen, eine bestehende Rolle an diesen Dienst zu übergeben, anstatt eine neue Servicerolle oder eine dienstverknüpfte Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in Timestream for auszuführen. LiveAnalytics Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb meines AWS Kontos den Zugriff auf meinen Timestream für LiveAnalytics Ressourcen ermöglichen

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Für Dienste, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (ACLs) unterstützen, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen darüber, ob Timestream for diese Funktionen LiveAnalytics unterstützt, finden Sie unter. So LiveAnalytics funktioniert Amazon Timestream for mit IAM
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen gewähren können, AWS-Konten die Ihnen gehören, finden Sie im IAM Benutzerhandbuch unter Gewähren des Zugriffs auf einen anderen IAMBenutzer AWS-Konto , der Ihnen gehört.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie AWS-Konten im IAMBenutzerhandbuch unter Gewähren des Zugriffs für Dritte.
- Informationen dazu, wie Sie Zugriff über einen Identitätsverbund gewähren, finden Sie im Benutzerhandbuch unter Zugriff für extern authentifizierte Benutzer (Identitätsverbund). IAM
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontenübergreifenden Zugriff finden Sie IAMim Benutzerhandbuch unter Kontoübergreifender Ressourcenzugriff. IAM

Protokollierung und Überwachung in Timestream für LiveAnalytics

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit und Leistung von Timestream for LiveAnalytics und Ihren AWS Lösungen. Sie sollten Überwachungsdaten aus allen Teilen Ihrer AWS Lösung sammeln, damit Sie einen etwaigen Ausfall an mehreren Stellen leichter debuggen können. Bevor Sie jedoch mit der Überwachung von Timestream beginnen LiveAnalytics, sollten Sie einen Überwachungsplan erstellen, der Antworten auf die folgenden Fragen enthält:

- Was sind Ihre Ziele bei der Überwachung?

- Welche Ressourcen werden überwacht?
- Wie oft werden diese Ressourcen überwacht?
- Welche Überwachungstools werden verwendet?
- Wer soll die Überwachungsaufgaben ausführen?
- Wer soll benachrichtigt werden, wenn Fehler auftreten?

Der nächste Schritt besteht darin, eine Ausgangsbasis für die normale LiveAnalytics Timestream-Leistung in Ihrer Umgebung festzulegen, indem Sie die Leistung zu verschiedenen Zeiten und unter verschiedenen Lastbedingungen messen. Speichern Sie bei der Überwachung von Timestream historische Überwachungsdaten LiveAnalytics, damit Sie sie mit aktuellen Leistungsdaten vergleichen, normale Leistungsmuster und Leistungsanomalien identifizieren und Methoden zur Behebung von Problemen entwickeln können.

Zur Festlegung eines Grundwertes sollten Sie mindestens die folgenden Elemente überwachen:

- Systemfehler, sodass Sie feststellen können, ob Anfragen zu einem Fehler geführt haben.

Themen

Überwachungstools

AWS stellt verschiedene Tools bereit, mit denen Sie Timestream überwachen können. LiveAnalytics Sie können einige dieser Tools so konfigurieren, dass diese die Überwachung für Sie übernehmen, während bei anderen Tools ein manuelles Eingreifen nötig ist. Wir empfehlen, dass Sie die Überwachungsaufgaben möglichst automatisieren.

Themen

Automatisierte Überwachungstools

Sie können die folgenden automatisierten Überwachungstools verwenden, um Timestream zu überwachen LiveAnalytics und zu melden, wenn etwas nicht stimmt:

- Amazon CloudWatch Alarms — Überwachen Sie eine einzelne Metrik über einen von Ihnen angegebenen Zeitraum und führen Sie eine oder mehrere Aktionen aus, die auf dem Wert der Metrik im Verhältnis zu einem bestimmten Schwellenwert über mehrere Zeiträume basieren. Die Aktion ist eine Benachrichtigung, die an ein Amazon Simple Notification Service (AmazonSNS) -Thema oder eine Amazon EC2 Auto Scaling Scaling-Richtlinie gesendet wird. CloudWatch Alarme lösen keine Aktionen aus, nur weil sie sich in einem bestimmten Zustand befinden. Der Status muss sich geändert haben und für eine bestimmte Anzahl von Zeiträumen beibehalten worden sein. Weitere Informationen finden Sie unter Überwachung mit Amazon CloudWatch.

Manuelle Überwachungstools

Ein weiterer wichtiger Teil der Timestream-Überwachung besteht in der LiveAnalytics manuellen Überwachung der Elemente, die von den CloudWatch Alarmen nicht abgedeckt werden. Der Timestream for LiveAnalytics, CloudWatch Trusted Advisor, und andere AWS Management Console Dashboards bieten einen at-a-glance Überblick über den Zustand Ihrer Umgebung. AWS

- Auf der CloudWatch Startseite wird Folgendes angezeigt:
  - Aktuelle Alarme und Status
  - Diagramme mit Alarmen und Ressourcen
  - Servicestatus

  Darüber hinaus können CloudWatch Sie Folgendes verwenden:
  - Erstellen angepasster Dashboards zur Überwachung der gewünschten Services.
  - Aufzeichnen von Metrikdaten, um Probleme zu beheben und Trends zu erkennen
  - Suchen und durchsuchen Sie alle Ihre AWS Ressourcenmetriken
  - Erstellen und Bearbeiten von Alarmen, um über Probleme benachrichtigt zu werden

Protokollierung des Timestreams für LiveAnalytics API Anrufe mit AWS CloudTrail

Timestream for LiveAnalytics ist in einen Dienst integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder einem AWS Dienst in Timestream

for ausgeführt wurden. LiveAnalytics CloudTrail erfasst API Aufrufe der Data Definition Language (DDL) für Timestream als Ereignisse. LiveAnalytics Zu den aufgezeichneten Aufrufen gehören Aufrufe aus dem Timestream für LiveAnalytics Konsolen und Code-Aufrufe aus dem Timestream für Operationen. LiveAnalytics API Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Übermittlung von CloudTrail Ereignissen an einen Amazon Simple Storage Service (Amazon S3) -Bucket aktivieren, einschließlich Ereignissen für Timestream for LiveAnalytics. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse trotzdem auf der CloudTrail Konsole im Ereignisverlauf anzeigen. Anhand der von CloudTrail gesammelten Informationen können Sie die Anfrage an Timestream LiveAnalytics, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Details ermitteln.

Weitere Informationen CloudTrail dazu finden Sie im AWS CloudTrail Benutzerhandbuch.

Timestream für LiveAnalytics Informationen in CloudTrail

CloudTrail ist in Ihrem AWS Konto aktiviert, wenn Sie das Konto erstellen. Wenn in Timestream für eine Aktivität auftritt LiveAnalytics, wird diese Aktivität zusammen mit anderen AWS Serviceereignissen in der CloudTrail Ereignishistorie in einem Ereignis aufgezeichnet. Sie können die neusten Ereignisse in Ihr AWS -Konto herunterladen und dort suchen und anzeigen. Weitere Informationen finden Sie unter Ereignisse mit CloudTrail Ereignisverlauf anzeigen.

> **Warning**
>
> Derzeit LiveAnalytics generiert Timestream for CloudTrail Ereignisse für alle Verwaltungs- und `Query` API Betriebsabläufe, generiert jedoch keine Ereignisse für `WriteRecords` und `DescribeEndpoints`APIs.

Für eine fortlaufende Aufzeichnung der Ereignisse in Ihrem AWS Konto, einschließlich der Ereignisse für Timestream für LiveAnalytics, erstellen Sie einen Trail. Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Wenn Sie einen Trail in der Konsole erstellen, gilt der Trail standardmäßig für alle AWS Regionen. Der Trail protokolliert Ereignisse aus allen Regionen der AWS Partition und übermittelt die Protokolldateien an den von Ihnen angegebenen Amazon S3 S3-Bucket. Darüber hinaus können Sie andere AWS Dienste konfigurieren, um die in den CloudTrail Protokollen gesammelten Ereignisdaten weiter zu analysieren und darauf zu reagieren.

Weitere Informationen finden Sie in folgenden Themen im AWS CloudTrail -Benutzerhandbuch:

- Übersicht zum Erstellen eines Trails
- CloudTrail Unterstützte Dienste und Integrationen
- Konfiguration von SNS Amazon-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail Protokolldateien aus mehreren Regionen
- Empfangen von CloudTrail Protokolldateien von mehreren Konten
- Protokollierung von Datenereignissen

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anfrage mit Root- oder AWS Identity and Access Management (IAM) Benutzeranmeldedaten gestellt wurde
- Ob die Anfrage mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen föderierten Benutzer ausgeführt wurde
- Ob die Anfrage von einem anderen AWS Dienst gestellt wurde

Weitere Informationen finden Sie im CloudTrail userIdentityElement.

Für `Query` API Veranstaltungen:

- Erstellen Sie einen Trail, der alle Ereignisse empfängt, oder wählen Sie Ereignisse mit Timestream als LiveAnalytics Ressourcentyp `AWS::Timestream::Database` oder `AWS::Timestream::Table` aus.
- `Query`APIAnfragen, die auf keine Datenbank oder Tabelle zugreifen oder die aufgrund einer falsch formatierten Abfragezeichenfolge zu einer Validierungsausnahme führen, werden CloudTrail mit einem Ressourcentyp `AWS::Timestream::Database` und einem ARN Wert von:

  ```
  arn:aws:timestream:(region):(accountId):database/NO_RESOURCE_ACCESSED
  ```

  Diese Ereignisse werden nur an Trails übermittelt, die Ereignisse mit dem Ressourcentyp `AWS::Timestream::Database` empfangen.

Ausfallsicherheit in Amazon Timestream Live Analytics

Die AWS globale Infrastruktur basiert auf AWS Regionen und Availability Zones. AWS Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Informationen zu den Datenschutzfunktionen von Timestream, die über verfügbar sind AWS Backup, finden Sie unterArbeiten mit AWS Backup.

Infrastruktursicherheit in Amazon Timestream Live Analytics

Als verwalteter Service ist Amazon Timestream Live Analytics durch die AWS globalen Netzwerksicherheitsverfahren geschützt, die im Whitepaper Amazon Web Services: Sicherheitsprozesse im Überblick beschrieben werden.

Sie verwenden AWS veröffentlichte API Aufrufe, um über das Netzwerk auf Timestream Live Analytics zuzugreifen. Clients müssen Transport Layer Security (TLS) 1.0 oder höher unterstützen. Wir empfehlen TLS 1.2 oder höher. Kunden müssen außerdem Cipher Suites mit Perfect Forward Secrecy (PFS) wie Ephemeral Diffie-Hellman () oder Elliptic Curve Ephemeral Diffie-Hellman (DHE) unterstützen. ECDHE Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Darüber hinaus müssen Anfragen mithilfe einer Zugriffsschlüssel-ID und eines geheimen Zugriffsschlüssels, der einem Prinzipal zugeordnet ist, signiert werden. IAM Alternativ können Sie mit AWS Security Token Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Timestream Live Analytics ist so konzipiert, dass Ihr Datenverkehr auf die spezifische AWS Region beschränkt ist, in der sich Ihre Timestream Live Analytics-Instanz befindet.

Konfiguration und Schwachstellenanalyse in Timestream

Konfiguration und IT-Kontrollen liegen in der gemeinsamen Verantwortung von AWS Ihnen, unserem Kunden. Weitere Informationen finden Sie im Modell der AWS gemeinsamen Verantwortung.

Zusätzlich zum Modell der gemeinsamen Verantwortung sollten LiveAnalytics Timestream-Benutzer Folgendes beachten:

- Es obliegt dem Kunden, seine Client-Anwendungen mit den relevanten kundenseitigen Abhängigkeiten zu patchen.
- Kunden sollten gegebenenfalls Penetrationstests in Betracht ziehen (siehe https://aws.amazon.com/security/Penetrationstesting/.)

Reaktion auf Vorfälle in Timestream für LiveAnalytics

Amazon Timestream for LiveAnalytics Service Incidents wird im Personal Health Dashboard gemeldet. Sie können mehr über das Dashboard und AWS Health hier erfahren.

Timestream for LiveAnalytics unterstützt die Berichterstattung mithilfe von AWS CloudTrail. Weitere Informationen finden Sie unter Protokollierung des Timestreams für LiveAnalytics API Anrufe mit AWS CloudTrail.

VPCEndpunkte ()AWS PrivateLink

Sie können eine private Verbindung zwischen Ihrem VPC und Amazon Timestream für einrichten, LiveAnalytics indem Sie einen VPCSchnittstellenendpunkt erstellen. Schnittstellenendpunkte werden mit einer Technologie betrieben AWS PrivateLink, mit der Sie privat auf Timestream zugreifen können, LiveAnalytics APIs ohne dass ein Internet-Gateway, ein NAT Gerät, eine Verbindung oder eine AWS Direct VPN Connect-Verbindung erforderlich ist. Instances in Ihrem VPC System benötigen keine öffentlichen IP-Adressen, um mit Timestream zu kommunizieren. LiveAnalytics APIs Der Datenverkehr zwischen Ihnen VPC und Timestream for LiveAnalytics verlässt das Amazon-Netzwerk nicht.

Jeder Schnittstellenendpunkt wird durch eine oder mehrere Elastic-Network-Schnittstellen in Ihren Subnetzen dargestellt. Weitere Informationen zu VPC Schnittstellenendpunkten finden Sie unter VPCSchnittstellenendpunkte (AWS PrivateLink) im VPCAmazon-Benutzerhandbuch.

Um mit Timestream for LiveAnalytics und VPC Endpoints zu beginnen, haben wir Informationen zu spezifischen Überlegungen für Timestream for LiveAnalytics mit Endpunkten, zur Erstellung eines VPC Schnittstellenendpunkts für Timestream for, zur Erstellung einer VPC Endpunktrichtlinie für Timestream for LiveAnalytics und zur Verwendung des Timestream-Clients (entweder für LiveAnalytics Write oder Query) mit VPC Endpunkten bereitgestellt. SDK VPC

Themen

VPCSo funktionieren Endgeräte mit Timestream

Wenn Sie einen VPC Endpunkt für den Zugriff auf Timestream Write oder Timestream Query erstellenSDK, werden alle Anfragen an Endpunkte innerhalb des Amazon-Netzwerks weitergeleitet und greifen nicht auf das öffentliche Internet zu. Genauer gesagt werden Ihre Anfragen an die Schreib- und Abfrageendpunkte der Zelle weitergeleitet, der Ihr Konto für eine bestimmte Region zugeordnet wurde. Weitere Informationen über die Mobilfunkarchitektur und die zellenspezifischen Endpunkte von Timestream finden Sie unter. Zellulare Architektur Nehmen wir zum Beispiel an, dass Ihr Konto mit `cell1` in `us-west-2` verknüpft wurde und Sie VPC Schnittstellenendpunkte für Schreibvorgänge () und Abfragen () `ingest-cell1.timestream.us-west-2.amazonaws.com` eingerichtet haben. `query-cell1.timestream.us-west-2.amazonaws.com` In diesem Fall bleiben alle Schreibanfragen, die über diese Endpunkte gesendet werden, vollständig im Amazon-Netzwerk und greifen nicht auf das öffentliche Internet zu.

Überlegungen zu Timestream-Endpunkten VPC

Beachten Sie beim Erstellen eines VPC Endpunkts für Timestream Folgendes:

- Bevor Sie einen VPC Schnittstellenendpunkt für Timestream einrichten, stellen Sie sicher LiveAnalytics, dass Sie die Eigenschaften und Einschränkungen der Schnittstellenendpunkte im VPCAmazon-Benutzerhandbuch lesen.
- Timestream for LiveAnalytics unterstützt das Aufrufen all seiner API Aktionen von Ihrem aus. VPC
- VPCEndpunktrichtlinien werden für Timestream unterstützt. LiveAnalytics Standardmäßig LiveAnalytics ist der vollständige Zugriff auf Timestream for über den Endpunkt zulässig. Weitere Informationen finden Sie unter Steuern des Zugriffs auf Dienste mit VPC Endpunkten im VPCAmazon-Benutzerhandbuch.
- Aufgrund der Architektur von Timestream erfordert der Zugriff auf Write- und Query-Aktionen die Erstellung von zwei VPC Schnittstellen-Endpunkten, jeweils einer für jeden. SDK Darüber hinaus müssen Sie einen Zellenendpunkt angeben (Sie können nur einen Endpunkt für die Timestream-Zelle erstellen, der Sie zugeordnet sind). Detaillierte Informationen finden Sie im LiveAnalytics Abschnitt Erstellen eines VPC Schnittstellenendpunkts für Timestream für dieses Handbuch.

Nachdem Sie nun verstanden haben, wie Timestream for mit VPC Endpunkten LiveAnalytics funktioniert, erstellen Sie einen VPC Schnittstellenendpunkt für Timestream for. LiveAnalytics

Erstellen Sie einen VPC Schnittstellenendpunkt für Timestream für LiveAnalytics

Sie können einen VPCSchnittstellenendpunkt für den Timestream for LiveAnalytics Service entweder mit der VPC Amazon-Konsole oder der AWS Command Line Interface (AWS CLI) erstellen. Um einen VPC Endpunkt für Timestream zu erstellen, führen Sie die unten beschriebenen TimeStream-spezifischen Schritte aus.

> **Note**
>
> Bevor Sie die folgenden Schritte ausführen, stellen Sie sicher, dass Sie die spezifischen Überlegungen zu Timestream-Endpunkten verstanden haben. VPC

Konstruieren eines VPC Endpunktdienstnamens mithilfe Ihrer Timestream-Zelle

Aufgrund der einzigartigen Architektur von Timestream müssen für jeden VPC Schnittstellenendpunkt SDK (Write und Query) separate Schnittstellenendpunkte erstellt werden. Darüber hinaus müssen Sie einen Timestream-Zellenendpunkt angeben (Sie können nur einen Endpunkt für die Timestream-Zelle erstellen, der Sie zugeordnet sind). Gehen Sie wie folgt vor, um Interface VPC Endpoints zu verwenden, um von Ihrem aus eine direkte Verbindung zu Timestream herzustellenVPC:

1. Suchen Sie zunächst nach einem verfügbaren Timestream-Zellenendpunkt. Um einen verfügbaren Zellenendpunkt zu finden, verwenden Sie die `DescribeEndpointsAktion` (verfügbar sowohl über Write als auch QueryAPIs), um die in Ihrem Timestream-Konto verfügbaren Zellenendpunkte aufzulisten. Weitere Informationen finden Sie im Beispiel.
2. Nachdem Sie einen zu verwendenden Zellenendpunkt ausgewählt haben, erstellen Sie eine VPC Schnittstellenendpunktzeichenfolge für Timestream Write oder QueryAPI:
   - Für den API Schreibvorgang:

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

   - Für die AbfrageAPI:

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

where *<region>* ist ein gültiger AWS Regionalcode und *<cell>* ist eine der Zellenendpunktadressen (wie `cell1` oder`cell2`), die von der DescribeEndpoints Aktion im Endpoints-Objekt zurückgegeben wurden. Weitere Informationen finden Sie im Beispiel.

3. Nachdem Sie einen Dienstnamen für den VPC Endpunkt erstellt haben, erstellen Sie einen Schnittstellenendpunkt. Wenn Sie aufgefordert werden, einen VPC Endpunktdienstnamen anzugeben, verwenden Sie den VPC Endpunktdienstnamen, den Sie in Schritt 2 erstellt haben.

Beispiel: Konstruieren Sie Ihren VPC Endpunkt-Servicenamen

Im folgenden Beispiel wird die `DescribeEndpoints` Aktion AWS CLI unter Verwendung von Write API in der `us-west-2` Region ausgeführt:

```
aws timestream-write describe-endpoints --region us-west-2
```

Dieser Befehl gibt die folgende Ausgabe zurück:

```
{
    "Endpoints": [
        {
            "Address": "ingest-cell1.timestream.us-west-2.amazonaws.com",
            "CachePeriodInMinutes": 1440
        }
    ]
}
```

In diesem Fall *cell1* ist der *<cell>* , und *us-west-2* ist der *<region>*. Der resultierende Name des VPC Endpunktdienstes sieht also wie folgt aus:

```
com.amazonaws.us-west-2.timestream.ingest-cell1
```

Nachdem Sie nun einen VPC Schnittstellenendpunkt für Timestream for erstellt haben LiveAnalytics, erstellen Sie eine VPC Endpunktrichtlinie für Timestream for. LiveAnalytics

Erstellen Sie eine VPC Endpunktrichtlinie für Timestream für LiveAnalytics

Sie können Ihrem VPC Endpunkt eine Endpunktrichtlinie hinzufügen, die den Zugriff auf Timestream für steuert. LiveAnalytics Die Richtlinie gibt die folgenden Informationen an:

- Prinzipal, der die Aktionen ausführen kann.
- Aktionen, die ausgeführt werden können
- Die Ressourcen, für die Aktionen ausgeführt werden können.

Weitere Informationen finden Sie unter Steuern des Zugriffs auf Dienste mit VPC Endpunkten im VPCAmazon-Benutzerhandbuch.

Beispiel: VPC Endpunktrichtlinie für Timestream für Aktionen LiveAnalytics

Im Folgenden finden Sie ein Beispiel für eine Endpunktrichtlinie für Timestream for. LiveAnalytics Wenn diese Richtlinie an einen Endpunkt angehängt ist, gewährt sie Zugriff auf den aufgelisteten Timestream für LiveAnalytics Aktionen (in diesem Fall `ListDatabases`) für alle Prinzipale auf allen Ressourcen.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "timestream:ListDatabases"
         ],
         "Resource":"*"
      }
   ]
}
```

Bewährte Sicherheitsmethoden für Amazon Timestream für LiveAnalytics

Amazon Timestream for LiveAnalytics bietet eine Reihe von Sicherheitsfunktionen, die Sie bei der Entwicklung und Implementierung Ihrer eigenen Sicherheitsrichtlinien berücksichtigen sollten. Die folgenden bewährten Methoden stellen allgemeine Richtlinien und keine vollständige Sicherheitslösung dar. Da diese bewährten Methoden für Ihre Umgebung möglicherweise nicht angemessen oder ausreichend sind, sollten Sie sie als hilfreiche Überlegungen und nicht als bindend ansehen.

Themen

- Timestream für bewährte Methoden zur LiveAnalytics präventiven Sicherheit

Timestream für bewährte Methoden zur LiveAnalytics präventiven Sicherheit

Die folgenden bewährten Methoden können Ihnen helfen, Sicherheitsvorfälle in Timestream for zu antizipieren und zu verhindern. LiveAnalytics

Verschlüsselung im Ruhezustand

- Timestream for LiveAnalytics verschlüsselt im Ruhezustand alle in Tabellen gespeicherten Benutzerdaten mithilfe von Verschlüsselungsschlüsseln, die in () gespeichert sind.AWS Key Management ServiceAWS KMS Dies bietet eine zusätzliche Datenschutzebene, indem Ihre Daten vor unbefugtem Zugriff auf den zugrunde liegenden Speicher geschützt werden.

  Timestream for LiveAnalytics verwendet einen einzigen Service-Standardschlüssel (AWS im BesitzCMK) für die Verschlüsselung all Ihrer Tabellen. Wenn dieser Schlüssel nicht existiert, wird er für Sie erstellt. Die Standardschlüssel des Dienstes können nicht deaktiviert werden. Weitere Informationen finden Sie unter Timestream for LiveAnalytics Encryption at Rest.

Verwenden Sie IAM Rollen, um den Zugriff auf Timestream zu authentifizieren für LiveAnalytics

- Damit Benutzer, Anwendungen und andere AWS Dienste auf Timestream zugreifen können LiveAnalytics, müssen sie gültige AWS Anmeldeinformationen in ihren Anfragen angeben. AWS API Sie sollten AWS Anmeldeinformationen nicht direkt in der Anwendung oder EC2 Instanz speichern. Dies sind langfristige Anmeldeinformationen, die nicht automatisch rotiert werden und daher erhebliche geschäftliche Auswirkungen haben können, wenn sie kompromittiert werden. Eine IAM Rolle ermöglicht es Ihnen, temporäre Zugriffsschlüssel zu erhalten, die für den Zugriff auf AWS Dienste und Ressourcen verwendet werden können.

  Weitere Informationen finden Sie unter IAM-Rollen.

Verwenden Sie IAM Richtlinien für Timestream für die LiveAnalytics Basisautorisierung

- Bei der Erteilung von Berechtigungen entscheiden Sie, wer sie erhält, für welchen Timestream LiveAnalytics APIs sie Berechtigungen erhalten und welche spezifischen Aktionen Sie für diese Ressourcen zulassen möchten. Die Implementierung der geringsten Rechte ist der Schlüssel zur Verringerung des Sicherheitsrisikos und der Auswirkungen, die sich aus Fehlern oder böswilligen Absichten ergeben können.

  Ordnen Sie Berechtigungsrichtlinien IAM Identitäten (d. h. Benutzern, Gruppen und Rollen) zu und gewähren Sie so Berechtigungen zur Ausführung von Vorgängen auf Timestream für Ressourcen. LiveAnalytics

  Sie können dies wie folgt tun:

- AWS verwaltete (vordefinierte) Richtlinien
- Kundenverwaltete Richtlinien
- Tag-basierte Autorisierung

Clientseitige Verschlüsselung in Betracht ziehen

Wenn Sie sensible oder vertrauliche Daten in Timestream for speichern LiveAnalytics, möchten Sie diese Daten möglicherweise so nah wie möglich an ihrem Ursprung verschlüsseln, damit Ihre Daten während ihres gesamten Lebenszyklus geschützt sind. Durch die Verschlüsselung Ihrer sensiblen Daten bei der Übertragung und Speicherung wird sichergestellt, dass Ihre Klartextdaten nicht für Dritte verfügbar sind.

Arbeiten mit anderen -Services

Amazon Timestream for LiveAnalytics lässt sich in eine Vielzahl von AWS Diensten und beliebten Tools von Drittanbietern integrieren. Derzeit LiveAnalytics unterstützt Timestream for Integrationen mit den folgenden Komponenten:

Themen

- Amazon-DynamoDB
- AWS Lambda
- AWS IoT Core
- Amazon Managed Service für Apache Flink
- Amazon Kinesis
- Amazon MQ
- Amazon MSK
- Amazon QuickSight
- Amazon SageMaker
- Amazon SQS
- Wird verwendetDBeaver, um mit Amazon Timestream zu arbeiten
- Grafana
- Wird verwendet SquaredUp , um mit Amazon Timestream zu arbeiten
- Open-Source-Telegraf
- JDBC

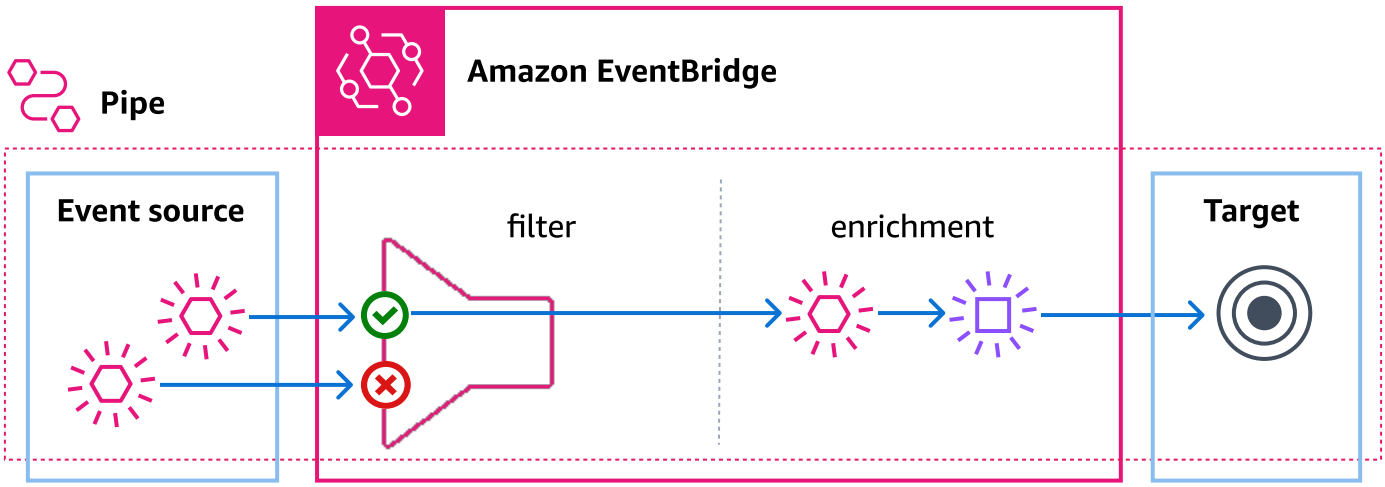
- ODBC
- VPCEndpunkte ()AWS PrivateLink

Amazon-DynamoDB

Verwenden von EventBridge Pipes zum Senden von DynamoDB-Daten an Timestream

Sie können EventBridge Pipes verwenden, um Daten von einem DynamoDB-Stream an einen Amazon Timestream für eine Tabelle zu senden. LiveAnalytics

Pipes sind für point-to-point Integrationen zwischen unterstützten Quellen und Zielen vorgesehen und unterstützen erweiterte Transformationen und Anreicherungen. Pipes reduzieren den Bedarf an Fachwissen und Integrationscode bei der Entwicklung ereignisgesteuerter Architekturen. Zum Einrichten einer Pipe wählen Sie die Quelle aus, fügen Sie optionale Filterung hinzu, definieren Sie die optionale Anreicherung und wählen Sie das Ziel für die Ereignisdaten.



Weitere Informationen zu EventBridge Pipes finden Sie im EventBridge Benutzerhandbuch unter EventBridge Pipes. Informationen zur Konfiguration einer Pipe für die Übermittlung von Ereignissen an eine Amazon Timestream LiveAnalytics Timestream-For-Tabelle finden Sie unter EventBridge Pipes-Zielspezifikationen.

AWS Lambda

Sie können Lambda-Funktionen erstellen, die mit Timestream for interagieren. LiveAnalytics Sie können beispielsweise eine Lambda-Funktion erstellen, die in regelmäßigen Abständen ausgeführt wird, um eine Abfrage in Timestream auszuführen und eine SNS Benachrichtigung zu senden, die

auf den Abfrageergebnissen basiert, die ein oder mehrere Kriterien erfüllen. Weitere Informationen zu Lambda finden Sie in der AWS Lambda-Dokumentation.

Themen

- Erstellen Sie AWS Lambda-Funktionen mit Amazon Timestream for LiveAnalytics mit Python
- Erstellen Sie AWS Lambda-Funktionen mit Amazon Timestream für mit LiveAnalytics JavaScript
- Erstellen Sie AWS Lambda-Funktionen mit Amazon Timestream for with Go LiveAnalytics
- Erstellen Sie AWS Lambda-Funktionen mit Amazon Timestream for LiveAnalytics mit C#

Erstellen Sie AWS Lambda-Funktionen mit Amazon Timestream for LiveAnalytics mit Python

Gehen Sie wie folgt vor, um AWS Lambda-Funktionen mit Amazon Timestream for LiveAnalytics mit Python zu erstellen.

1. Erstellen Sie eine IAM Rolle, von der Lambda annehmen soll, dass sie die erforderlichen Berechtigungen für den Zugriff auf den Timestream-Dienst gewährt, wie unter beschrieben. Stellen Sie Timestream für den Zugriff bereit LiveAnalytics
2. Bearbeiten Sie die Vertrauensstellung der IAM Rolle, um den Lambda-Dienst hinzuzufügen. Sie können die folgenden Befehle verwenden, um eine bestehende Rolle zu aktualisieren, sodass AWS Lambda sie übernehmen kann:

   a. Erstellen Sie das Dokument mit der Vertrauensrichtlinie:

```
cat > Lambda-Role-Trust-Policy.json << EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "lambda.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

```
EOF
```

b. Aktualisieren Sie die Rolle aus dem vorherigen Schritt mit dem Vertrauensdokument

```
aws iam update-assume-role-policy --role-name <name_of_the_role_from_step_1> --
policy-document file://Lambda-Role-Trust-Policy.json
```

Verwandte Referenzen finden Sie unter TimestreamWriteund TimestreamQuery.

Erstellen Sie AWS Lambda-Funktionen mit Amazon Timestream für mit LiveAnalytics JavaScript

Folgen Sie den hier beschriebenen Anweisungen, um AWS Lambda-Funktionen LiveAnalytics mit JavaScript Amazon Timestream for with zu erstellen.

Verwandte Referenzen finden Sie unter Timestream Write Client — AWS SDK für JavaScript v3 und Timestream Query Client — für v3. AWS SDK JavaScript

Erstellen Sie AWS Lambda-Funktionen mit Amazon Timestream for with Go LiveAnalytics

Folgen Sie den hier beschriebenen Anweisungen, um AWS Lambda-Funktionen LiveAnalytics mit Amazon Timestream for with Go zu erstellen.

Verwandte Referenzen finden Sie unter timestreamwrite und timestreamquery.

Erstellen Sie AWS Lambda-Funktionen mit Amazon Timestream for LiveAnalytics mit C#

Folgen Sie den hier beschriebenen Anweisungen, um AWS Lambda-Funktionen mithilfe von Amazon Timestream for LiveAnalytics mit C# zu erstellen.

Verwandte Referenzen finden Sie bei Amazon. TimestreamWriteund Amazon. TimestreamQuery.

AWS IoT Core

Sie können mithilfe von IoT Core Daten von AWS IoT-Geräten sammeln und die Daten über IoT Core-Regelaktionen an Amazon Timestream weiterleiten. AWS IoT-Regelaktionen geben an, was zu tun ist, wenn eine Regel ausgelöst wird. Sie können Aktionen definieren, um Daten an eine Amazon

Timestream Timestream-Tabelle, eine Amazon DynamoDB DynamoDB-Datenbank zu senden und eine Lambda-Funktion aufzurufen. AWS

Die Timestream-Aktion in IoT-Regeln wird verwendet, um Daten aus eingehenden Nachrichten direkt in Timestream einzufügen. Die Aktion analysiert die Ergebnisse der IoT SQL Core-Anweisung und speichert Daten in Timestream. Die Namen der Felder aus der zurückgegebenen SQL Ergebnismenge werden als Measure: :Name verwendet, und der Wert des Feldes ist Measure: :value.

Betrachten Sie zum Beispiel die SQL Anweisung und die Payload der Beispielnachricht:

```
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
}
```

Wenn mit der obigen SQL Anweisung eine IoT Core-Regelaktion für Timestream erstellt wird, werden zwei Datensätze mit den Messnamen Temperatur und Luftfeuchtigkeit und den Messwerten 24,04 bzw. 43,605 zu Timestream hinzugefügt.

Sie können den Messnamen eines Datensatzes, der zu Timestream hinzugefügt wird, ändern, indem Sie den AS-Operator in der Anweisung verwenden. SELECT Mit der folgenden SQL Anweisung wird ein Datensatz mit dem Nachrichtennamen temp anstelle von temperature erstellt.

Der Datentyp der Messgröße wird aus dem Datentyp des Werts der Nachrichtennutzlast abgeleitet. JSONDatentypen wie Integer, Double, Boolean und String werden jeweils den Timestream-Datentypen,, und zugeordnet. BIGINT DOUBLE BOOLEAN VARCHAR Mit der Funktion cast () können Daten auch bestimmten Datentypen zugewiesen werden. Sie können den Zeitstempel der

Kennzahl angeben. Wenn der Zeitstempel leer gelassen wird, wird die Zeit verwendet, zu der der Eintrag verarbeitet wurde.

Weitere Informationen finden Sie in der Dokumentation zur Aktion Timestream-Regeln

Gehen Sie wie folgt vor, um eine IoT Core-Regelaktion zum Speichern von Daten in Timestream zu erstellen:

Themen

Voraussetzungen

1. Erstellen Sie mithilfe der unter beschriebenen Anweisungen eine Datenbank in Amazon Timestream. Erstellen einer -Datenbank
2. Erstellen Sie eine Tabelle in Amazon Timestream anhand der unter beschriebenen Anweisungen. Erstellen einer Tabelle

Verwenden der Konsole

1. Verwenden Sie die AWS Managementkonsole für AWS IoT Core, um eine Regel zu erstellen, indem Sie auf Verwalten > Nachrichtenweiterleitung > Regeln und anschließend auf Regel erstellen klicken.
2. Geben Sie als Regelnamen einen Namen Ihrer Wahl und anschließend SQL den unten angezeigten Text ein

```
SELECT temperature as temp, humidity from 'iot/topic'
```

3. Wählen Sie Timestream aus der Aktionsliste
4. Geben Sie die Namen der Timestream-Datenbank, der Tabelle und der Dimension sowie die Rolle zum Schreiben von Daten in Timestream an. Wenn die Rolle nicht existiert, können Sie eine erstellen, indem Sie auf Rollen erstellen klicken

5. Folgen Sie den hier angezeigten Anweisungen, um die Regel zu testen.

Mit dem CLI

Wenn Sie die AWS Befehlszeilenschnittstelle (AWS CLI) nicht installiert haben, tun Sie dies von hier aus.

1. Speichern Sie die folgende Regelnutzlast in einer JSON Datei namens timestream_rule.json. Ersetzen *arn:aws:iam::123456789012:role/TimestreamRole* mit Ihrer Rolle arn, die AWS IoT-Zugriff zum Speichern von Daten in Amazon Timestream gewährt

```
{
    "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. Erstellen Sie mit dem folgenden Befehl eine Themenregel

```
aws iot create-topic-rule --rule-name timestream_test --topic-rule-payload file://
<path/to/timestream_rule.json> --region us-east-1
```

3. Rufen Sie mit dem folgenden Befehl Details zur Themenregel ab

```
aws iot get-topic-rule --rule-name timestream_test
```

4. Speichern Sie die folgende Nachrichten-Payload in einer Datei namens 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. Testen Sie die Regel mit dem folgenden Befehl

```
aws iot-data publish --topic 'iot/topic' --payload file://<path/to/
timestream_msg.json>
```

Beispielanwendung

Um Ihnen den Einstieg in die Verwendung von Timestream mit AWS IoT Core zu erleichtern, haben wir eine voll funktionsfähige Beispielanwendung erstellt, die die erforderlichen Artefakte in AWS IoT Core und Timestream für die Erstellung einer Themenregel und eine Beispielanwendung für die Veröffentlichung von Daten zum Thema erstellt.

1. Klonen Sie das GitHub Repository für die Beispielanwendung für die AWS IoT Core-Integration gemäß den Anweisungen von GitHub
2. Folgen Sie den Anweisungen im README, um mithilfe einer AWS CloudFormation Vorlage die erforderlichen Artefakte in Amazon Timestream und AWS IoT Core zu erstellen und Beispielnachrichten zum Thema zu veröffentlichen.

Video-Tutorial

In diesem Video wird erklärt, wie IoT Core mit Timestream funktioniert.

Amazon Managed Service für Apache Flink

Sie können Apache Flink verwenden, um Ihre Zeitreihendaten von Amazon Managed Service für Apache Flink, AmazonMSK, Apache Kafka und anderen Streaming-Technologien direkt in Amazon Timestream for zu übertragen. LiveAnalytics Wir haben einen Apache Flink-Beispieldaten-Connector für Timestream erstellt. Wir haben auch eine Beispielanwendung für das Senden von Daten an Amazon Kinesis erstellt, sodass die Daten von Kinesis zu Managed Service für Apache Flink und schließlich weiter zu Amazon Timestream fließen können. All diese Artefakte stehen Ihnen in zur Verfügung. GitHub In diesem Video-Tutorial wird das Setup beschrieben.

> **Note**
>
> Java 11 ist die empfohlene Version für die Verwendung der Managed Service for Apache Flink-Anwendung. Wenn Sie mehrere Java-Versionen haben, stellen Sie sicher, dass Sie Java 11 in Ihre HOME Umgebungsvariable JAVA _ exportieren.

Themen

- Beispielanwendung
- Video-Tutorial

Beispielanwendung

Gehen Sie zunächst wie folgt vor:

1. Erstellen Sie in Timestream eine Datenbank mit dem Namen gemäß den `kdaflink` Anweisungen unter Erstellen einer -Datenbank
2. Erstellen Sie in Timestream eine Tabelle mit dem Namen gemäß den `kinesisdata1` Anweisungen unter Erstellen einer Tabelle
3. Erstellen Sie einen Amazon Kinesis Data Stream mit dem Namen gemäß den `TimestreamTestStream` Anweisungen unter Stream erstellen.
4. Klonen Sie das GitHub Repository für den Apache Flink-Datenconnector für Timestream gemäß den Anweisungen von GitHub

5. Folgen Sie den Anweisungen im Apache Flink-Beispieldatenconnector, um die Beispielanwendung zu kompilieren, auszuführen und zu verwenden README
6. Kompilieren Sie die Managed Service for Apache Flink-Anwendung gemäß den Anweisungen zum Kompilieren des Anwendungscodes
7. Laden Sie die Binärdatei der Anwendung Managed Service for Apache Flink hoch und folgen Sie dabei den Anweisungen zum Hochladen des Apache Flink-Streaming-Codes

    a. Nachdem Sie auf Anwendung erstellen geklickt haben, klicken Sie auf den Link der IAM Rolle für die Anwendung

    b. Hängen Sie die IAM Richtlinien für AmazonKinesisReadOnlyAccessund an AmazonTimestreamFullAccess.

    > **Note**
    >
    > Die oben genannten IAM Richtlinien sind nicht auf bestimmte Ressourcen beschränkt und eignen sich nicht für den produktiven Einsatz. Erwägen Sie für ein Produktionssystem die Verwendung von Richtlinien, die den Zugriff auf bestimmte Ressourcen einschränken.

8. Klonen Sie das GitHub Repository für die Beispielanwendung, die Daten in Kinesis schreibt. Folgen Sie dabei den Anweisungen von GitHub
9. Folgen Sie den Anweisungen in README, um die Beispielanwendung zum Schreiben von Daten in Kinesis auszuführen
10. Führen Sie eine oder mehrere Abfragen in Timestream aus, um sicherzustellen, dass Daten von Kinesis an Managed Service for Apache Flink to Timestream gesendet werden. Folgen Sie dabei den Anweisungen Erstellen einer Tabelle

Video-Tutorial

Dieses Video erklärt, wie Timestream mit Managed Service für Apache Flink verwendet wird.

Amazon Kinesis

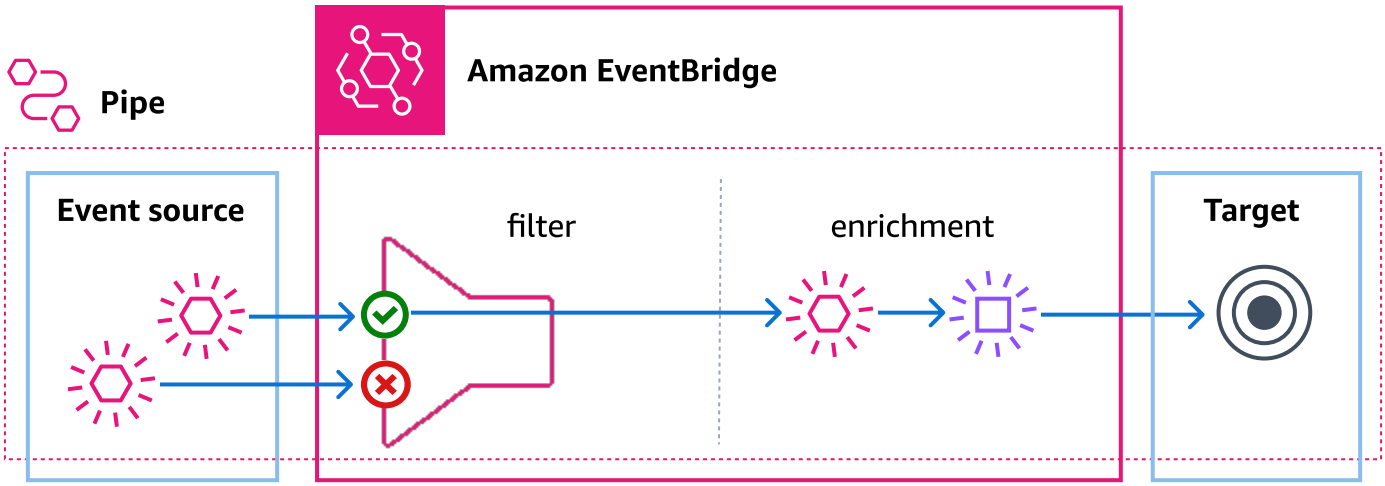
Verwenden Amazon Managed Service für Apache Flink

Sie können Daten von Kinesis Data Streams an Timestream senden, um den Timestream Beispieldatenconnector für Managed Service für Apache Flink zu LiveAnalytics verwenden. Weitere Informationen finden Sie Amazon Managed Service für Apache Flink unter Apache Flink.

Verwenden von EventBridge Pipes zum Senden von Kinesis-Daten an Timestream

Sie können EventBridge Pipes verwenden, um Daten von einem Kinesis-Stream an einen Amazon Timestream für LiveAnalytics eine Tabelle zu senden.

Pipes sind für point-to-point Integrationen zwischen unterstützten Quellen und Zielen vorgesehen und unterstützen erweiterte Transformationen und Anreicherungen. Pipes reduzieren den Bedarf an Fachwissen und Integrationscode bei der Entwicklung ereignisgesteuerter Architekturen. Zum Einrichten einer Pipe wählen Sie die Quelle aus, fügen Sie optionale Filterung hinzu, definieren Sie die optionale Anreicherung und wählen Sie das Ziel für die Ereignisdaten.



Diese Integration ermöglicht es Ihnen, die Leistungsfähigkeit der Funktionen zur Analyse Timestream von Zeitreihendaten zu nutzen und gleichzeitig Ihre Datenerfassungspipeline zu vereinfachen.

Die Verwendung von EventBridge Pipes with Timestream bietet die folgenden Vorteile:

- Datenaufnahme in Echtzeit: Streamen Sie Daten von Kinesis direkt zu Timestream für und ermöglichen so Analysen LiveAnalytics und Überwachung in Echtzeit.

- Nahtlose Integration: Verwenden Sie EventBridge Pipes, um den Datenfluss zu verwalten, ohne dass komplexe benutzerdefinierte Integrationen erforderlich sind.
- Verbesserte Filterung und Transformation: Filtern oder transformieren Sie Kinesis-Datensätze, bevor sie gespeichert werden Timestream , um Ihre spezifischen Datenverarbeitungsanforderungen zu erfüllen.
- Skalierbarkeit: Verarbeiten Sie Datenströme mit hohem Durchsatz und sorgen Sie für eine effiziente Datenverarbeitung mit integrierten Parallelismus- und Batching-Funktionen.

Konfiguration

Gehen Sie wie folgt vor, um eine EventBridge Pipe einzurichten Timestream, an die Daten von Kinesis gestreamt werden sollen:

1. Kinesis-Stream erstellen

   Stellen Sie sicher, dass Sie über einen aktiven Kinesis-Datenstream verfügen, aus dem Sie Daten aufnehmen möchten.
2. Erstellen Sie eine Timestream Datenbank und eine Tabelle

   Richten Sie Ihre Timestream Datenbank und Tabelle ein, in der die Daten gespeichert werden.
3. Konfigurieren Sie die EventBridge Pipe:
   - Quelle: Wählen Sie Ihren Kinesis-Stream als Quelle aus.
   - Ziel: Wählen Sie Timestream es als Ziel aus.
   - Batching-Einstellungen: Definieren Sie das Batch-Fenster und die Batchgröße, um die Datenverarbeitung zu optimieren und die Latenz zu reduzieren.

> **Important**
>
> Beim Einrichten einer Pipe empfehlen wir, die Richtigkeit aller Konfigurationen zu testen, indem Sie einige Datensätze aufnehmen. Bitte beachten Sie, dass die erfolgreiche Erstellung einer Pipe nicht garantiert, dass die Pipeline korrekt ist und die Daten fehlerfrei fließen. Nach dem Anwenden der Zuordnung können Laufzeitfehler auftreten, wie z. B. eine falsche Tabelle, ein falscher dynamischer Pfadparameter oder ein ungültiger Timestream Datensatz, die entdeckt werden, wenn die tatsächlichen Daten durch die Pipe fließen.

Die folgenden Konfigurationen bestimmen die Geschwindigkeit, mit der Daten aufgenommen werden:

- BatchSize: Die maximale Größe des Batches, für den Timestream gesendet wird. LiveAnalytics Bereich: 0 — 100. Es wird empfohlen, diesen Wert bei 100 zu belassen, um einen maximalen Durchsatz zu erzielen.
- MaximumBatchingWindowInSeconds: Die maximale Wartezeit, bis der Stapel gefüllt ist, batchSize bevor der Stapel als LiveAnalytics Ziel an Timestream gesendet wird. Abhängig von der Rate eingehender Ereignisse entscheidet diese Konfiguration über die Verzögerung der Aufnahme. Es wird empfohlen, diesen Wert auf < 10 Sekunden zu belassen, um die Daten nahezu in Echtzeit zu Timestream senden.
- ParallelizationFactor: Die Anzahl der Batches, die von jedem Shard gleichzeitig verarbeitet werden sollen. Es wird empfohlen, den Maximalwert 10 zu verwenden, um einen maximalen Durchsatz und eine Aufnahme nahezu in Echtzeit zu erzielen.

  Wenn Ihr Stream von mehreren Zielen gelesen wird, verwenden Sie den erweiterten Fan-Out, um einen speziellen Nutzer für Ihre Pipe bereitzustellen und so einen hohen Durchsatz zu erzielen. Weitere Informationen finden Sie im Kinesis Data Streams Benutzerhandbuch unter Entwickeln erweiterter Kinesis Data Streams API Fan-Out-Consumer mit dem.

> **Note**
>
> Der maximale Durchsatz, der erreicht werden kann, ist durch gleichzeitige Pipe-Ausführungen pro Konto begrenzt.

Die folgende Konfiguration stellt sicher, dass Datenverlust verhindert wird:

- DeadLetterConfig: Es wird empfohlen, die Konfiguration immer so DeadLetterConfig zu konfigurieren, dass Datenverlust vermieden wird, wenn Ereignisse LiveAnalytics aufgrund von Benutzerfehlern nicht in Timestream aufgenommen werden konnten.

Optimieren Sie die Leistung Ihrer Pipe mit den folgenden Konfigurationseinstellungen, um zu verhindern, dass Aufzeichnungen zu Verlangsamungen oder Blockaden führen.

- MaximumRecordAgeInSeconds: Ältere Datensätze werden nicht verarbeitet und direkt dorthin verschoben. DLQ Wir empfehlen, diesen Wert so einzustellen, dass er nicht höher ist als die konfigurierte Aufbewahrungsdauer für den Speicherspeicher der Timestream Zieltabelle.

- MaximumRetryAttempts: Die Anzahl der Wiederholungsversuche für einen Datensatz, bevor der Datensatz an DeadLetterQueue gesendet wird. Es wird empfohlen, diesen Wert auf 10 zu konfigurieren. Dies sollte helfen, vorübergehende Probleme zu beheben. Bei anhaltenden Problemen wird der Datensatz in den Rest des Streams verschoben DeadLetterQueue und dort entsperrt.
- OnPartialBatchItemFailure: Für Quellen, die eine teilweise Stapelverarbeitung unterstützen, empfehlen wir Ihnen, diese Option zu aktivieren und sie als AUTOMATIC _ BISECT zu konfigurieren, damit fehlgeschlagene Datensätze vor dem Ablegen/Senden an erneut versucht werden können. DLQ

Konfigurationsbeispiel

Hier ist ein Beispiel für die Konfiguration einer EventBridge Pipe zum Streamen von Daten aus einem Kinesis-Stream in eine Timestream Tabelle:

Example IAM Richtlinien-Updates für 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 Kinesis-Stream-Konfiguration

```
{
  "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 Zielkonfiguration

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

Transformation von Ereignissen

EventBridge Pipes ermöglichen es Ihnen, Daten zu transformieren, bevor sie ankommen. Timestream Sie können Transformationsregeln definieren, um die eingehenden Kinesis Datensätze zu ändern, z. B. um Feldnamen zu ändern.

Angenommen, Ihr Kinesis Stream enthält Temperatur- und Feuchtigkeitsdaten. Sie können diese Felder mithilfe einer EventBridge Transformation umbenennen, bevor Sie sie in Timestream einfügen.

Bewährte Methoden

Batching und Pufferung

- Konfigurieren Sie das Batchfenster und die Größe so, dass ein ausgewogenes Verhältnis zwischen Schreiblatenz und Verarbeitungseffizienz besteht.
- Verwenden Sie ein Batch-Fenster, um vor der Verarbeitung genügend Daten zu sammeln und so den Aufwand für häufige kleine Batches zu reduzieren.

Parallele Verarbeitung

Verwenden Sie diese ParallelizationFactorEinstellung, um die Parallelität zu erhöhen, insbesondere bei Streams mit hohem Durchsatz. Dadurch wird sichergestellt, dass mehrere Batches von jedem Shard gleichzeitig verarbeitet werden können.

Datentransformation

Nutzen Sie die Transformationsfunktionen von EventBridge Pipes, um Datensätze zu filtern und zu verbessern, bevor Sie sie speichern Timestream. Dies kann dazu beitragen, die Daten an Ihre analytischen Anforderungen anzupassen.

Sicherheit

- Stellen Sie sicher, dass die für EventBridge Pipes verwendeten IAM Rollen über die erforderlichen Lese Kinesis - und Schreibberechtigungen verfügen. Timestream
- Verwenden Sie Verschlüsselungs- und Zugriffskontrollmaßnahmen, um Daten während der Übertragung und Speicherung zu schützen.

Fehler beim Debuggen

- Automatische Deaktivierung von Pipes

  Pipes werden in etwa 2 Stunden automatisch deaktiviert, wenn das Ziel nicht existiert oder es Probleme mit den Berechtigungen gibt
- Drosselungen

  Pipes sind in der Lage, sich automatisch zurückzuziehen und es erneut zu versuchen, bis die Drosselung reduziert ist.
- Protokolle aktivieren

  Wir empfehlen Ihnen, Logs auf ERROR Ebene zu aktivieren und Ausführungsdaten einzubeziehen, um mehr Einblicke in Fehlschläge zu erhalten. Bei einem Fehler enthalten diese Protokolle request/

response sent/received von Timestream. Dies hilft Ihnen, den damit verbundenen Fehler zu verstehen und die Datensätze nach der Behebung gegebenenfalls erneut zu verarbeiten.

Überwachen

Wir empfehlen Ihnen, Alarme für Folgendes einzurichten, um Probleme mit dem Datenfluss zu erkennen:

- Maximales Alter des Datensatzes in der Quelle
  - `GetRecords.IteratorAgeMilliseconds`
- Messwerte für Fehler in Pipes
  - `ExecutionFailed`
  - `TargetStageFailed`
- Timestream APIFehler schreiben
  - `UserErrors`

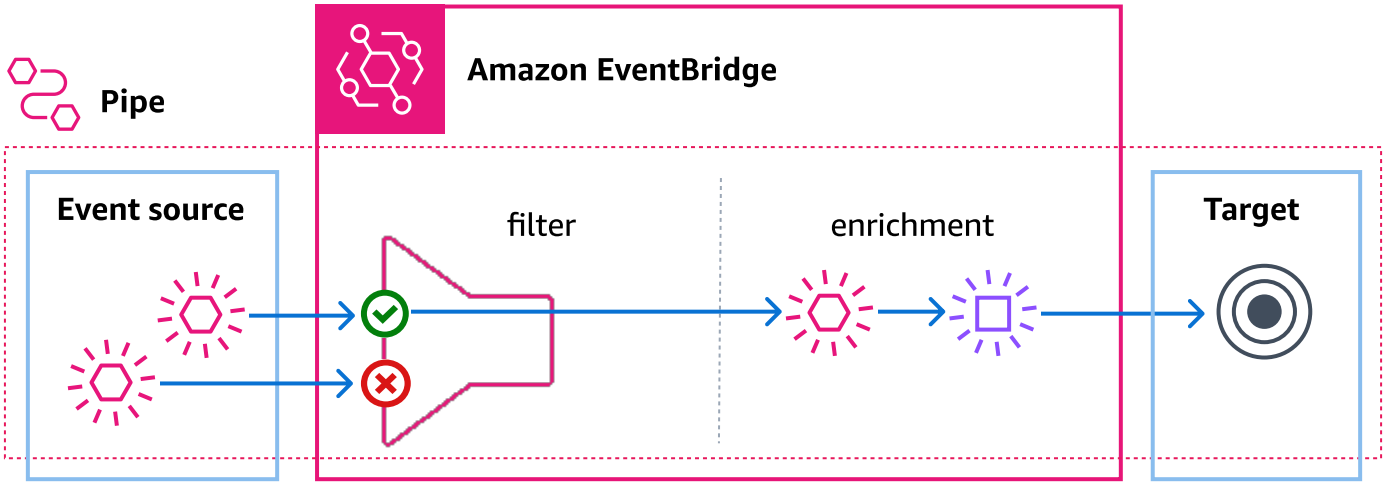
Weitere Monitoring-Metriken finden Sie unter Überwachung EventBridge im EventBridge Benutzerhandbuch.

Amazon MQ

Verwenden von EventBridge Pipes zum Senden von Amazon MQ MQ-Daten an Timestream

Sie können EventBridge Pipes verwenden, um Daten von einem Amazon MQ-Broker an einen Amazon Timestream für LiveAnalytics Tabellen zu senden.

Pipes sind für point-to-point Integrationen zwischen unterstützten Quellen und Zielen vorgesehen und unterstützen erweiterte Transformationen und Anreicherungen. Pipes reduzieren den Bedarf an Fachwissen und Integrationscode bei der Entwicklung ereignisgesteuerter Architekturen. Zum Einrichten einer Pipe wählen Sie die Quelle aus, fügen Sie optionale Filterung hinzu, definieren Sie die optionale Anreicherung und wählen Sie das Ziel für die Ereignisdaten.



Weitere Informationen zu EventBridge Pipes finden Sie im EventBridge Benutzerhandbuch unter EventBridge Pipes. Informationen zur Konfiguration einer Pipe für die Übermittlung von Ereignissen an einen Amazon Timestream für eine LiveAnalytics Tabelle finden Sie unter EventBridge Pipes-Zielspezifikationen.

Amazon MSK

Verwenden von Managed Service for Apache Flink zum Senden von Amazon MSK Daten an Timestream für LiveAnalytics

Sie können Daten von Amazon MSK nach senden, Timestream indem Sie einen Datenkonnektor erstellen, der dem Timestream Beispieldatenkonnektor für Managed Service für Apache Flink ähnelt. Weitere Informationen finden Sie unter Amazon Managed Service für Apache Flink.

Verwenden von Kafka Connect zum Senden von MSK Amazon-Daten an Timestream für LiveAnalytics

Sie können Kafka Connect verwenden, um Ihre Zeitreihendaten Amazon MSK direkt in Timestream for aufzunehmen. LiveAnalytics

Wir haben ein Beispiel für einen Kafka Sink Connector erstellt. Timestream Wir haben auch ein Beispiel für einen Apache jMeter Testplan für die Veröffentlichung von Daten zu einem Kafka-Thema erstellt, sodass die Daten vom Thema über den Timestream Kafka Sink Connector zu einem Timestream für eine Tabelle fließen können. LiveAnalytics All diese Artefakte sind auf verfügbar. GitHub

> **Note**
>
> Java 11 ist die empfohlene Version für die Verwendung des Timestream Kafka Sink Connectors. Wenn Sie mehrere Java-Versionen haben, stellen Sie sicher, dass Sie Java 11 in Ihre HOME Umgebungsvariable JAVA _ exportieren.

Eine Beispielanwendung erstellen

Gehen Sie zunächst wie folgt vor.

1. Erstellen Sie in Timestream for LiveAnalytics eine Datenbank mit dem Namen`kafkastream`.

   Ausführliche Anweisungen finden Sie im ??? Verfahren.
2. Erstellen Sie in Timestream for LiveAnalytics eine Tabelle mit dem Namen`purchase_history`.

   Ausführliche Anweisungen finden Sie im ??? Verfahren.
3. Folgen Sie den Anweisungen in, um Folgendes zu erstellen:, und.

   - Ein Amazon MSK Cluster
   - Eine Amazon EC2 Instanz, die als Kafka-Producer-Client-Computer konfiguriert ist
   - Ein Kafka-Thema

   Eine ausführliche Anleitung finden Sie in den Voraussetzungen für das Projekt kafka_ingestor.
4. Klonen Sie das Timestream Kafka Sink Connector-Repository.

   Eine ausführliche Anleitung finden Sie unter Ein Repository klonen GitHub auf.
5. Kompilieren Sie den Plugin-Code.

   Ausführliche Anweisungen finden Sie unter Connector — GitHub Aus dem Quellcode erstellen.
6. Laden Sie die folgenden Dateien in einen S3-Bucket hoch: Folgen Sie den Anweisungen unter.

   - Die JAR-Datei (kafka-connector-timestream-> VERSION <- jar-with-dependencies .jar) aus dem Verzeichnis `/target`
   - Die Beispiel-JSON-Schemadatei,`purchase_history.json`.

Ausführliche Anweisungen finden Sie im Amazon S3 Benutzerhandbuch unter Hochladen von Objekten.

7. Erstellen Sie zwei VPC Endpunkte. Diese Endpunkte würden vom MSK Connector verwendet, um auf die Ressourcen zuzugreifen, die verwendet werden. AWS PrivateLink

   - Einer für den Zugriff auf den Bucket Amazon S3
   - Eine für den Zugriff auf den Timestream für die LiveAnalytics Tabelle.

   Ausführliche VPCAnweisungen finden Sie unter Endpoints.

8. Erstellen Sie ein benutzerdefiniertes Plugin mit der hochgeladenen JAR-Datei.

   Ausführliche Anweisungen finden Sie unter Plugins im Amazon MSK Entwicklerhandbuch.

9. Erstellen Sie eine benutzerdefinierte Worker-Konfiguration mit den unter Worker-Konfigurationsparametern beschriebenen JSON Inhalten. Folgen Sie den Anweisungen unter

   Ausführliche Anweisungen finden Sie im Amazon MSK Entwicklerhandbuch unter Erstellen einer benutzerdefinierten Worker-Konfiguration.

10. Erstellen Sie eine IAM Dienstausführungsrolle.

    Ausführliche Anweisungen finden Sie unter IAM Servicerolle.

11. Erstellen Sie einen Amazon MSK Connector mit dem benutzerdefinierten Plugin, der benutzerdefinierten Worker-Konfiguration und der IAM Dienstausführungsrolle, die in den vorherigen Schritten erstellt wurden, sowie mit der Beispiel-Connector-Konfiguration.

    Ausführliche Anweisungen finden Sie im Amazon MSK Entwicklerhandbuch unter Einen Connector erstellen.

    Achten Sie darauf, die Werte der folgenden Konfigurationsparameter mit den entsprechenden Werten zu aktualisieren. Einzelheiten finden Sie unter Konnektor-Konfigurationsparameter.

    - `aws.region`
    - `timestream.schema.s3.bucket.name`
    - `timestream.ingestion.endpoint`

Die Erstellung des Konnektors dauert 5—10 Minuten. Die Pipeline ist bereit, wenn ihr Status auf geändert wird. `Running`

12. Veröffentlichen Sie einen kontinuierlichen Nachrichtenstrom zum Schreiben von Daten zu dem erstellten Kafka-Thema.

    Ausführliche Anweisungen finden Sie unter So verwenden Sie es.

13. Führen Sie eine oder mehrere Abfragen aus, um sicherzustellen, dass die Daten von an MSK Connect Amazon MSK to the Timestream for LiveAnalytics table gesendet werden.

    Ausführliche Anweisungen finden Sie im ??? Verfahren.

Weitere Ressourcen

Der Blog Serverless Data Ingestion from your Kafka clusters into Timestream for LiveAnalytics using Kafka Connect erklärt die Einrichtung einer end-to-end Pipeline mit dem Timestream for Kafka Sink Connector, angefangen von einem LiveAnalytics Kafka-Producer-Client-Computer, der den jMeter Apache-Testplan verwendet, um Tausende von Beispielnachrichten zu einem Kafka-Thema zu veröffentlichen, bis hin zur Überprüfung der aufgenommenen Datensätze in einem Timestream for table. LiveAnalytics

Amazon QuickSight

Sie können Amazon verwenden QuickSight , um Daten-Dashboards zu analysieren und zu veröffentlichen, die Ihre Amazon Timestream Timestream-Daten enthalten. In diesem Abschnitt wird beschrieben, wie Sie eine neue QuickSight Datenquellenverbindung erstellen, Berechtigungen ändern, neue Datensätze erstellen und eine Analyse durchführen können. Dieses Video-Tutorial beschreibt, wie Sie mit Timestream und Amazon QuickSight arbeiten.

> **Note**
>
> Alle Datensätze in Amazon QuickSight sind schreibgeschützt. Sie können keine Änderungen an Ihren tatsächlichen Daten in Timestream vornehmen, indem Sie Amazon verwenden, QuickSight um die Datenquelle, den Datensatz oder die Felder zu entfernen.

Themen

- Zugriff auf Amazon Timestream von QuickSight
- Erstellen Sie eine neue QuickSight Datenquellenverbindung für Timestream
- Bearbeiten Sie die Berechtigungen für die QuickSight Datenquellenverbindung für Timestream
- Erstellen Sie einen neuen QuickSight Datensatz für Timestream
- Erstellen Sie eine neue Analyse für Timestream
- Videotutorial

Zugriff auf Amazon Timestream von QuickSight

Bevor Sie fortfahren können, QuickSight muss Amazon autorisiert sein, eine Verbindung zu Amazon Timestream herzustellen. Wenn Verbindungen nicht aktiviert sind, erhalten Sie eine Fehlermeldung, wenn Sie versuchen, eine Verbindung herzustellen. Ein QuickSight Administrator kann Verbindungen zu AWS Ressourcen autorisieren. Um eine Verbindung von zu Timestream QuickSight zu autorisieren, folgen Sie den Anweisungen unter Andere AWS Dienste verwenden: Zugriff einschränken und wählen Sie in Schritt 5 Amazon Timestream aus.

Erstellen Sie eine neue QuickSight Datenquellenverbindung für Timestream

> **Note**
>
> Die Verbindung zwischen Amazon QuickSight und Amazon Timestream wird bei der Übertragung mit SSL (TLS1.2) verschlüsselt. Sie können keine unverschlüsselte Verbindung herstellen.

1. Stellen Sie sicher, dass Sie die entsprechenden Berechtigungen für Amazon für den QuickSight Zugriff auf Amazon Timestream konfiguriert haben, wie unter beschrieben Zugriff auf Amazon Timestream von QuickSight.
2. Beginnen Sie mit der Erstellung eines neuen Datensatzes. Wählen Sie im Navigationsbereich Datensätze und anschließend Neuer Datensatz aus.
3. Wählen Sie die Timestream-Datenquellenkarte aus.
4. Geben Sie unter Datenquellenname beispielsweise einen Namen für Ihre Timestream-Datenquellenverbindung ein. `US Timestream Data`

> **Note**
>
> Da Sie über eine Verbindung zu Timestream viele Datensätze erstellen können, empfiehlt es sich, den Namen einfach zu halten.

5. Wählen Sie Verbindung validieren, um zu überprüfen, ob Sie erfolgreich eine Verbindung zum Timestream herstellen können.

> **Note**
>
> Mit der Option Verbindung validieren wird nur überprüft, ob Sie eine Verbindung herstellen können. Eine bestimmte Tabelle oder Abfrage wird jedoch nicht validiert.

6. Wählen Sie Datenquelle erstellen, um fortzufahren.
7. Wählen Sie für Datenbank die Option Auswählen... um die Liste der verfügbaren Optionen anzuzeigen. Wählen Sie die aus, die Sie verwenden möchten.
8. Wählen Sie Auswählen, um fortzufahren.
9. Wählen Sie eine der folgenden Optionen aus:
   - Um Ihre Daten in die QuickSight In-Memory-Engine (genanntSPICE) zu importieren, wählen Sie Import in SPICE für schnellere Analysen.
   - Damit QuickSight Sie bei jeder Aktualisierung des Datensatzes oder bei jeder Verwendung der Analyse oder des Dashboards eine Abfrage Ihrer Daten ausführen können, wählen Sie Ihre Daten direkt abfragen.
10. Wählen Sie Bearbeiten/Vorschau und dann Speichern, um Ihren Datensatz zu speichern und zu schließen.

Bearbeiten Sie die Berechtigungen für die QuickSight Datenquellenverbindung für Timestream

Das folgende Verfahren beschreibt, wie Sie Berechtigungen für andere QuickSight Benutzer anzeigen, hinzufügen und entziehen, sodass diese auf dieselbe Timestream-Datenquelle zugreifen können. Die Personen müssen aktive Benutzer sein, QuickSight bevor Sie sie hinzufügen können.

> **Note**
>
> QuickSightIn haben Datenquellen zwei Berechtigungsstufen: Benutzer und Besitzer.
>
> - Wählen Sie den Benutzer aus, dem Lesezugriff gewährt werden soll.
> - Wählen Sie den Besitzer aus, um diesem Benutzer zu erlauben, diese QuickSight Datenquelle zu bearbeiten, zu teilen oder zu löschen.

1. Stellen Sie sicher, dass Sie die entsprechenden Berechtigungen für Amazon für den QuickSight Zugriff auf Amazon Timestream konfiguriert haben, wie unter beschrieben Zugriff auf Amazon Timestream von QuickSight.
2. Wählen Sie auf der linken Seite Datensätze und scrollen Sie dann nach unten, um die Datenquellenkarte für Ihre Timestream-Verbindung zu finden. Zum Beispiel `US Timestream Data`.
3. Wählen Sie die `Timestream` Datenquellenkarte aus.
4. Wählen Sie `Share data source`. Eine Liste der aktuellen Berechtigungen wird angezeigt.
5. (Optional) Um Berechtigungen zu bearbeiten, können Sie `user` oder wählen`owner`.
6. (Optional) Um Berechtigungen zu widerrufen, wählen Sie`Revoke access`. Personen, die Sie widerrufen, können aus dieser Datenquelle keine neuen Datensätze erstellen. Ihre vorhandenen Datensätze haben jedoch weiterhin Zugriff auf diese Datenquelle.
7. Um Berechtigungen hinzuzufügen`Invite users`, wählen Sie und folgen Sie dann diesen Schritten, um einen Benutzer hinzuzufügen:

   a. Fügen Sie Personen hinzu, damit sie dieselbe Datenquelle verwenden können.

   b. Wählen Sie jeweils die aus, für `Permission` die Sie sich bewerben möchten.
8. Wählen Sie danach `Close` aus.

Erstellen Sie einen neuen QuickSight Datensatz für Timestream

1. Stellen Sie sicher, dass Sie die entsprechenden Berechtigungen für Amazon für den QuickSight Zugriff auf Amazon Timestream konfiguriert haben, wie unter beschrieben Zugriff auf Amazon Timestream von QuickSight.
2. Wählen Sie auf der linken Seite Datensätze und scrollen Sie dann nach unten, um die Datenquellenkarte für Ihre Timestream-Verbindung zu finden. Wenn Sie über viele Datenquellen

verfügen, können Sie die Suchleiste oben auf der Seite verwenden, um sie zu finden, deren Name teilweise übereinstimmt.

3. Wählen Sie die Timestream-Datenquellenkarte. Wählen Sie dann Datensatz erstellen.
4. Wählen Sie für Datenbank die Option Auswählen aus, um die Liste der verfügbaren Optionen anzuzeigen. Wählen Sie die Datenbank aus, die Sie verwenden möchten.
5. Wählen Sie für Tabellen die Tabelle aus, die Sie verwenden möchten.
6. Wählen Sie Bearbeiten/Vorschau aus.
7. (Optional) Um weitere Daten hinzuzufügen, wählen Sie oben rechts Daten hinzufügen aus.
   a. Wählen Sie Datenquelle wechseln und wählen Sie eine andere Datenquelle aus.
   b. Folgen Sie den Anweisungen der Benutzeroberfläche, um das Hinzufügen von Daten abzuschließen.
   c. Nachdem Sie demselben Datensatz neue Daten hinzugefügt haben, wählen Sie Diese Verknüpfung konfigurieren (die beiden roten Punkte) aus. Richten Sie für jede weitere Tabelle eine Verknüpfung ein.
   d. Wenn Sie Kalkulationsfelder hinzufügen möchten, wählen Sie Kalkulationsfeld hinzufügen aus.
   e. Um Sagemaker zu verwenden, wählen Sie „Erweitern mit“. SageMaker Diese Option ist nur in QuickSight der Enterprise Edition verfügbar.
   f. Deaktivieren Sie alle Felder, die Sie auslassen möchten.
   g. Aktualisieren Sie alle Datentypen, die Sie ändern möchten.
8. Wenn Sie fertig sind, wählen Sie Speichern, um den Datensatz zu speichern und zu schließen.

Erstellen Sie eine neue Analyse für Timestream

1. Stellen Sie sicher, dass Sie die entsprechenden Berechtigungen für Amazon für den QuickSight Zugriff auf Amazon Timestream konfiguriert haben, wie unter beschrieben Zugriff auf Amazon Timestream von QuickSight.
2. Wählen Sie links Analysen aus.
3. Wählen Sie eine der folgenden Optionen aus:
   - Um eine neue Analyse zu erstellen, wählen Sie auf der rechten Seite Neue Analyse aus.

- Um den Timestream-Datensatz zu einer vorhandenen Analyse hinzuzufügen, öffnen Sie die Analyse, die Sie bearbeiten möchten. Wählen Sie das Stiftsymbol oben links und dann Datensatz hinzufügen.

4. Starten Sie die erste Datenvisualisierung, indem Sie Felder auf der linken Seite auswählen.
5. Weitere Informationen finden Sie unter Arbeiten mit Analysen — Amazon QuickSight

Videotutorial

In diesem Video wird erklärt, wie Amazon mit Timestream QuickSight zusammenarbeitet.

Amazon SageMaker

Sie können Amazon SageMaker Notebooks verwenden, um Ihre Machine-Learning-Modelle in Amazon Timestream zu integrieren. Um Ihnen den Einstieg zu erleichtern, haben wir ein SageMaker Beispiel-Notebook erstellt, das Daten aus Timestream verarbeitet. Die Daten werden von einer Python-Anwendung mit mehreren Threads in Timestream eingefügt, die kontinuierlich Daten sendet. Der Quellcode für das SageMaker Beispiel-Notebook und die Python-Beispielanwendung sind verfügbar in GitHub.

1. Erstellen Sie eine Datenbank und eine Tabelle gemäß den unter Erstellen einer -Datenbank und beschriebenen Anweisungen Erstellen einer Tabelle
2. Klonen Sie das GitHub Repository für die Python-Beispielanwendung mit mehreren Threads gemäß den Anweisungen von GitHub
3. Klonen Sie das GitHub Repository für das SageMaker Timestream-Beispiel-Notebook gemäß den Anweisungen von. GitHub
4. Führen Sie die Anwendung aus, um kontinuierlich Daten in Timestream aufzunehmen. Folgen Sie dabei den Anweisungen in README
5. Folgen Sie den Anweisungen, um einen Amazon S3 S3-Bucket für Amazon zu erstellen, SageMaker wie hier beschrieben.
6. Erstellen Sie eine SageMaker Amazon-Instance mit der neuesten Version von boto3: Gehen Sie zusätzlich zu den hier beschriebenen Anweisungen wie folgt vor:

    a. Klicken Sie auf der Seite Notebook-Instance erstellen auf Zusätzliche Konfiguration

    b. Klicken Sie auf Lebenszykluskonfiguration — optional und wählen Sie Neue Lebenszykluskonfiguration erstellen

    c. Gehen Sie im Feld Assistent zum Erstellen einer Lebenszykluskonfiguration wie folgt vor:

i. Geben Sie einen gewünschten Namen für die Konfiguration ein, z. B. `on-start`

ii. Kopieren Sie unter Notebook-Skript starten den Skriptinhalt von Github und fügen Sie ihn ein

iii. Ersetzen Sie `PACKAGE=scipy` durch `PACKAGE=boto3` im eingefügten Skript.

7. Klicken Sie auf Konfiguration erstellen
8. Rufen Sie den IAM Dienst in der AWS Management Console auf und suchen Sie die neu erstellte SageMaker Ausführungsrolle für die Notebook-Instanz.
9. Hängen Sie die IAM Richtlinie für `AmazonTimestreamFullAccess` an die Ausführungsrolle an.

> **Note**
>
> Die `AmazonTimestreamFullAccess` IAM Richtlinie ist nicht auf bestimmte Ressourcen beschränkt und für den produktiven Einsatz ungeeignet. Erwägen Sie für ein Produktionssystem die Verwendung von Richtlinien, die den Zugriff auf bestimmte Ressourcen einschränken.

10. Wenn der Status der Notebook-Instanz lautet InService, wählen Sie Open Jupyter, um ein SageMaker Notebook für die Instance zu starten
11. Laden Sie die Dateien hoch **`timestreamquery.py`** und **`Timestream_SageMaker_Demo.ipynb`** in das Notebook, indem Sie auf die Schaltfläche Hochladen klicken
12. Wählen Sie `Timestream_SageMaker_Demo.ipynb` aus.

> **Note**
>
> Wenn Sie ein Popup-Fenster mit der Meldung „Kernel nicht gefunden“ sehen, wählen Sie conda_python3 und klicken Sie auf Kernel festlegen.

13. Ändern Sie`DB_NAME`,`TABLE_NAME`, und so`bucket`, `ENDPOINT` dass sie dem Datenbanknamen, dem Tabellennamen, dem S3-Bucket-Namen und der Region für die Trainingsmodelle entsprechen.
14. Wählen Sie das Play-Symbol, um die einzelnen Zellen auszuführen
15. Wenn Sie zu der Zelle gelangen`Leverage Timestream to find hosts with average CPU utilization across the fleet`, stellen Sie sicher, dass die Ausgabe mindestens 2 Hostnamen zurückgibt.

> **Note**
>
> Wenn die Ausgabe weniger als 2 Hostnamen enthält, müssen Sie möglicherweise die Python-Beispielanwendung erneut ausführen, die Daten mit einer größeren Anzahl von Threads und auf Host-Skala in Timestream aufnimmt.

16. Wenn Sie zu der Zelle kommen`Train a Random Cut Forest (RCF) model using the CPU utilization history`, ändern Sie sie `train_instance_type` je nach den Ressourcenanforderungen für Ihre Ausbildungsstelle
17. Wenn Sie zu der Zelle gelangen`Deploy the model for inference`, ändern Sie die `instance_type` anhand des Ressourcenbedarfs für Ihren Inferenzjob

> **Note**
>
> Das Trainieren des Modells kann einige Minuten dauern. Wenn das Training abgeschlossen ist, wird in der Ausgabe der Zelle die Meldung Abgeschlossen — Trainingsjob abgeschlossen angezeigt.

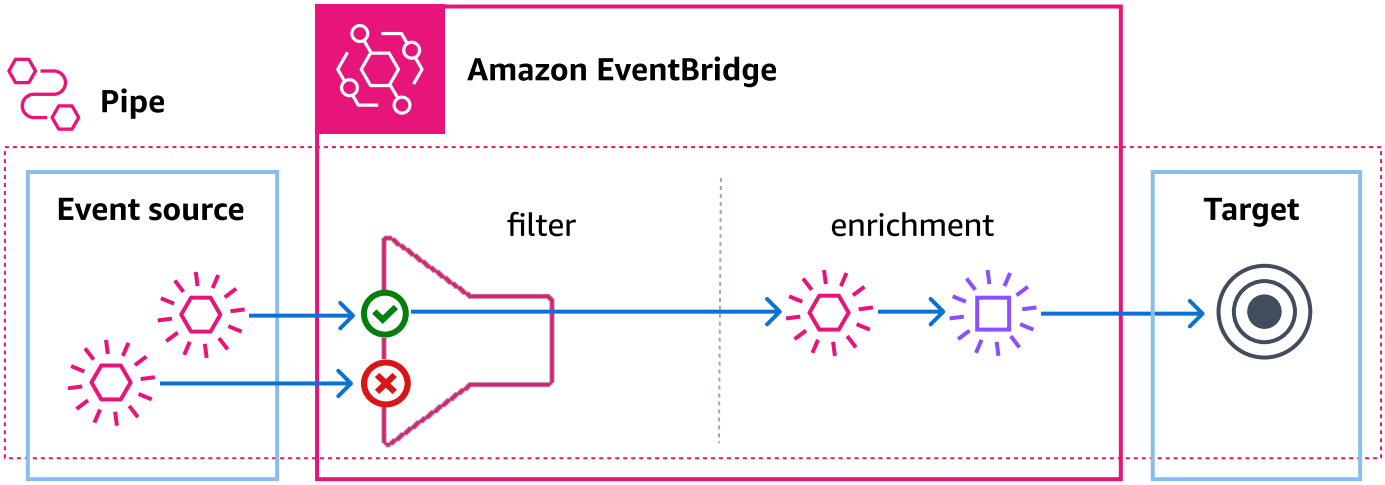
18. Führen Sie die Zelle aus`Stop and delete the endpoint`, um die Ressourcen zu bereinigen. Sie können die Instanz auch von der SageMaker Konsole aus beenden und löschen

Amazon SQS

Verwenden von EventBridge Pipes zum Senden von SQS Amazon-Daten an Timestream

Sie können EventBridge Pipes verwenden, um Daten aus einer SQS Amazon-Warteschlange an einen Amazon Timestream für eine LiveAnalytics Tabelle zu senden.

Pipes sind für point-to-point Integrationen zwischen unterstützten Quellen und Zielen vorgesehen und unterstützen erweiterte Transformationen und Anreicherungen. Pipes reduzieren den Bedarf an Fachwissen und Integrationscode bei der Entwicklung ereignisgesteuerter Architekturen. Zum Einrichten einer Pipe wählen Sie die Quelle aus, fügen Sie optionale Filterung hinzu, definieren Sie die optionale Anreicherung und wählen Sie das Ziel für die Ereignisdaten.



Weitere Informationen zu EventBridge Pipes finden Sie im EventBridge Benutzerhandbuch unter EventBridge Pipes. Informationen zur Konfiguration einer Pipe für die Übermittlung von Ereignissen an einen Amazon Timestream für eine LiveAnalytics Tabelle finden Sie unter EventBridge Pipes-Zielspezifikationen.

Wird verwendetDBeaver, um mit Amazon Timestream zu arbeiten

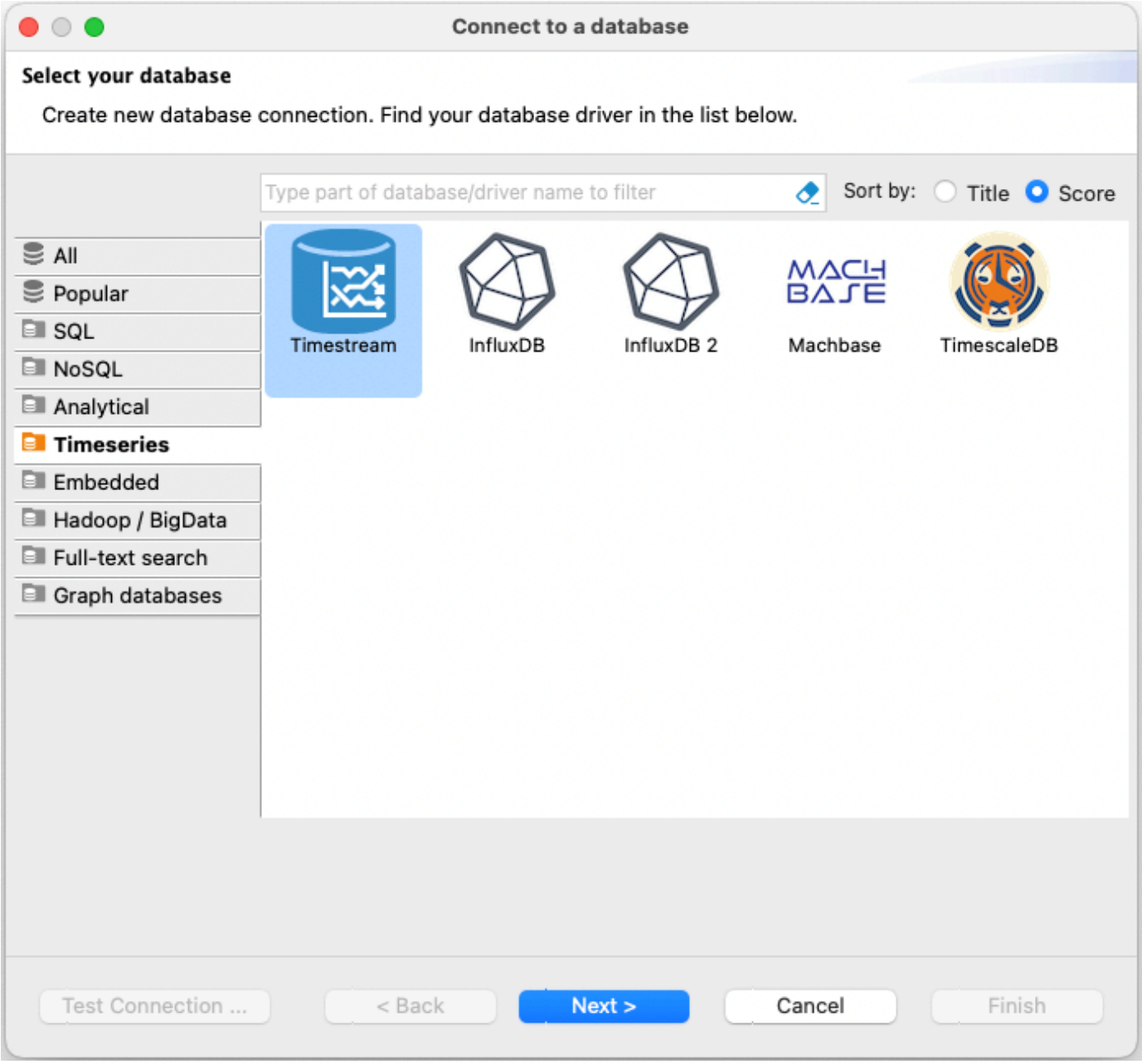
DBeaverist ein kostenloser SQL Universalclient, mit dem jede Datenbank verwaltet werden kann, die über einen JDBC Treiber verfügt. Aufgrund seiner robusten Funktionen zum Anzeigen, Bearbeiten und Verwalten von Daten wird er häufig von Entwicklern und Datenbankadministratoren verwendet.

Mithilfe DBeaver der Cloud-Konnektivitätsoptionen können Sie eine native Verbindung DBeaver zu Amazon Timestream herstellen. DBeaverbietet eine umfassende und intuitive Oberfläche für die Arbeit mit Zeitreihendaten direkt aus einer DBeaver Anwendung heraus. Mit Ihren Anmeldeinformationen erhalten Sie außerdem vollen Zugriff auf alle Abfragen, die Sie von einer anderen Abfrageschnittstelle aus ausführen könnten. Sie können damit sogar Diagramme erstellen, um die Abfrageergebnisse besser zu verstehen und zu visualisieren.

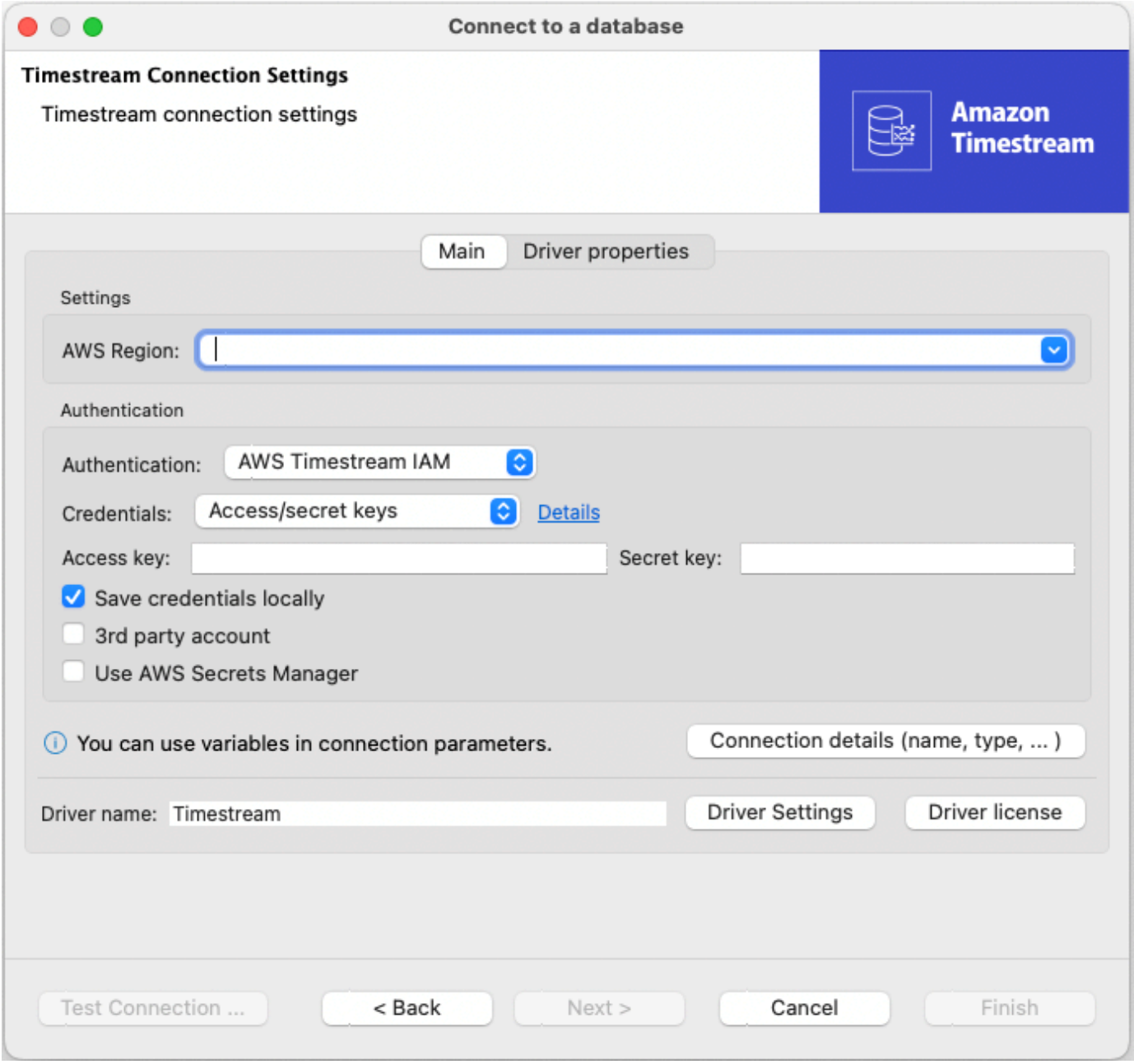
Einrichtung für DBeaver die Arbeit mit Timestream

Gehen Sie wie folgt vor, um die Arbeit mit Timestream einzurichtenDBeaver:

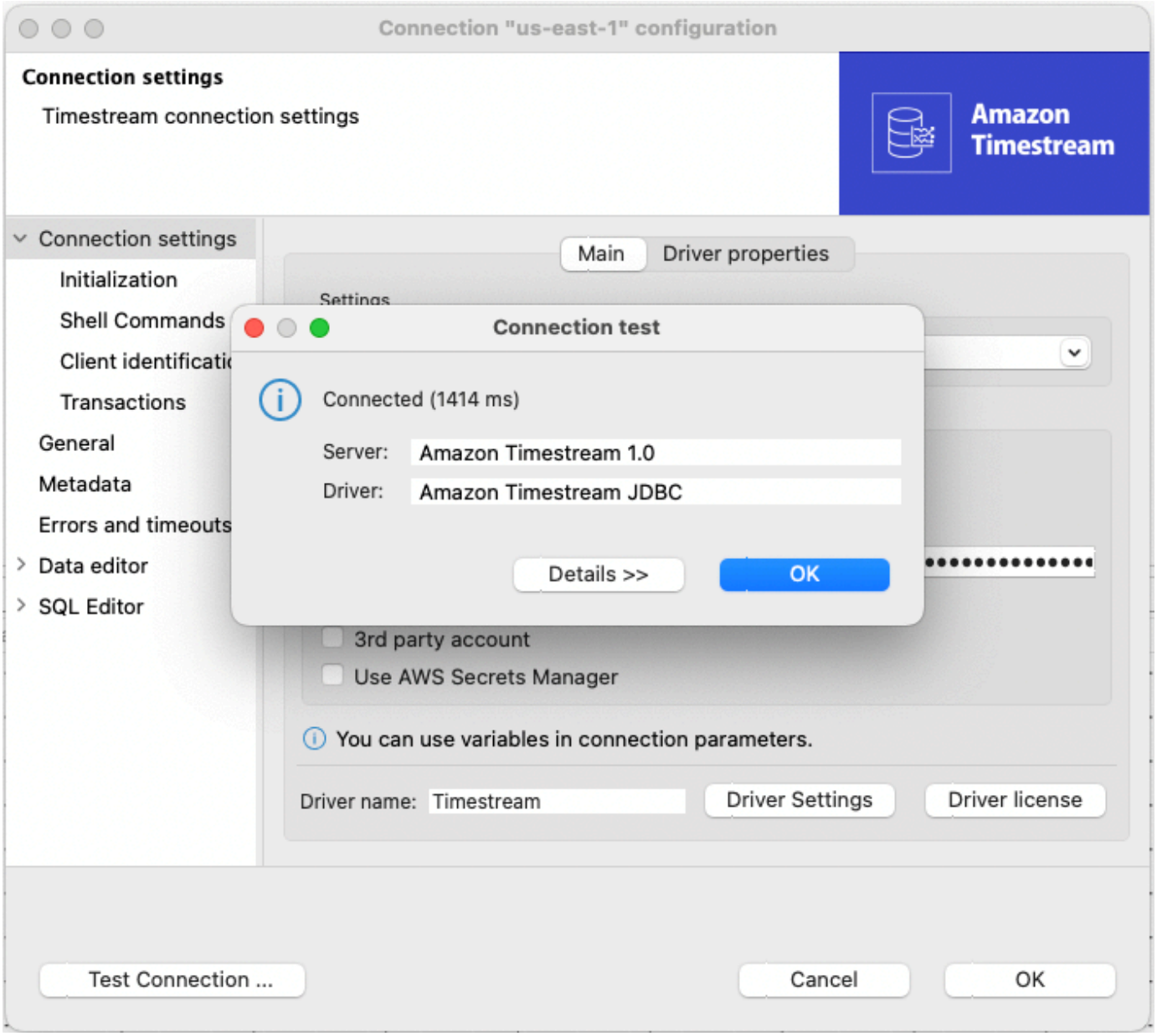
1. Laden Sie es herunter und installieren Sie es DBeaver auf Ihrem lokalen Computer.
2. Starten SieDBeaver, navigieren Sie zum Datenbankauswahlbereich, wählen Sie im linken Bereich Timeseries und dann im rechten Bereich das Timestream-Symbol aus:



3. Geben Sie im Fenster Timestream-Verbindungseinstellungen alle Informationen ein, die für die Verbindung mit Ihrer Amazon Timestream Timestream-Datenbank erforderlich sind. Bitte stellen Sie sicher, dass die von Ihnen eingegebenen Benutzerschlüssel über die erforderlichen Berechtigungen für den Zugriff auf Ihre Timestream-Datenbank verfügen. Stellen Sie außerdem sicher, dass die von Ihnen eingegebenen Informationen und Schlüssel DBeaver sicher und vertraulich behandelt werden, wie bei allen vertraulichen Informationen.



4. Testen Sie die Verbindung, um sicherzustellen, dass alles korrekt eingerichtet ist:



5. Wenn der Verbindungstest erfolgreich ist, können Sie jetzt mit Ihrer Amazon Timestream Timestream-Datenbank genauso interagieren, wie Sie es mit jeder anderen Datenbank tun würden. DBeaver Sie können beispielsweise zum SQL Editor oder zur ER-Diagrammansicht navigieren, um Abfragen auszuführen:

6. DBeaverbietet auch leistungsstarke Tools zur Datenvisualisierung. Um sie zu verwenden, führen Sie Ihre Abfrage aus und wählen Sie dann das Grafiksymbol aus, um den Ergebnissatz zu visualisieren. Das Grafiktool kann Ihnen helfen, Datentrends im Zeitverlauf besser zu verstehen.

Durch die Kopplung mit Amazon Timestream DBeaver entsteht eine effektive Umgebung für die Verwaltung von Zeitreihendaten. Sie können es nahtlos in Ihren bestehenden Arbeitsablauf integrieren, um Produktivität und Effizienz zu steigern.

Grafana

Mit Grafana können Sie Ihre Zeitreihendaten visualisieren und Benachrichtigungen erstellen. Um Ihnen den Einstieg in die Datenvisualisierung zu erleichtern, haben wir in Grafana ein Beispiel-Dashboard erstellt, das Daten visualisiert, die von einer Python-Anwendung an Timestream gesendet wurden, sowie ein Video-Tutorial, das die Einrichtung beschreibt.

Themen

- Beispielanwendung
- Video-Tutorial

Beispielanwendung

1. Erstellen Sie eine Datenbank und eine Tabelle in Timestream, indem Sie den Anweisungen folgen, die unter Weitere Informationen beschrieben sind. Erstellen einer -Datenbank

   > **Note**
   >
   > Der Standarddatenbankname und der Tabellenname für das Grafana-Dashboard sind auf GrafanaDB bzw. festgelegt. grafanaTable Verwenden Sie diese Namen, um die Einrichtung zu minimieren.

2. Installieren Sie Python 3.7 oder höher
3. Installieren und konfigurieren Sie Timestream Python SDK
4. Klonen Sie das GitHub Repository für die Multithread-Python-Anwendung, die kontinuierlich Daten in Timestream aufnimmt, indem Sie den Anweisungen von GitHub
5. Führen Sie die Anwendung aus, um kontinuierlich Daten in Timestream aufzunehmen, und folgen Sie dabei den Anweisungen in README
6. Schließen Sie Getting started with Amazon Managed Grafana ab oder schließen Sie Install Grafana ab.
7. Wenn Sie Grafana installieren, anstatt Amazon Managed Grafana zu verwenden, schließen Sie die Installation des Timestream-Plug-ins für Grafana ab.

8. Öffnen Sie das Grafana-Dashboard mit einem Browser Ihrer Wahl. Wenn Sie Grafana lokal installiert haben, können Sie den in der Grafana-Dokumentation beschriebenen Anweisungen folgen, um sich anzumelden
9. Gehen Sie nach dem Start von Grafana zu Datenquellen, klicken Sie auf Datenquelle hinzufügen, suchen Sie nach Timestream und wählen Sie die Timestream-Datenquelle aus
10. Konfigurieren Sie den Auth Provider und die Region und klicken Sie auf Speichern und Testen
11. Legen Sie die Standardmakros fest
    a. Setzen Sie $__database auf den Namen Ihrer Timestream-Datenbank (z. B. GrafanaDB)
    b. Setzen Sie $__table auf den Namen Ihrer Timestream-Tabelle (z. B.) grafanaTable
    c. Setzen Sie $__measure auf die am häufigsten verwendete Kennzahl aus der Tabelle
12. Klicken Sie auf Speichern und testen
13. Klicken Sie auf den Tab Dashboards
14. Klicken Sie auf Import, um das Dashboard zu importieren
15. Doppelklicken Sie auf das Beispielanwendungs-Dashboard
16. Klicken Sie auf die Dashboard-Einstellungen
17. Wählen Sie Variablen
18. Ändern Sie dbName die Namen der Timestream-Datenbank und -Tabelle und passen Sie sie an tableName
19. Klicken Sie auf Speichern
20. Aktualisieren Sie das Dashboard
21. Um Benachrichtigungen zu erstellen, folgen Sie den Anweisungen in der Grafana-Dokumentation zum Erstellen einer verwalteten Grafana-Warnregel
22. Um Warnmeldungen zu beheben, folgen Sie den Anweisungen in der Grafana-Dokumentation zur Fehlerbehebung
23. Weitere Informationen finden Sie in der Grafana-Dokumentation

Video-Tutorial

Dieses Video erklärt, wie Grafana mit Timestream arbeitet.

Wird verwendet SquaredUp , um mit Amazon Timestream zu arbeiten

SquaredUpist eine Observability-Plattform, die in Amazon Timestream integriert ist. Sie können den intuitiven Dashboard-Designer verwenden SquaredUp, um Ihre Zeitreihendaten zu visualisieren, zu analysieren und zu überwachen. Dashboards können öffentlich oder privat geteilt werden, und es können Benachrichtigungskanäle eingerichtet werden, um Sie zu benachrichtigen, wenn sich der Zustand eines Monitors ändert.

Verwendung SquaredUp mit Amazon Timestream

1. Melden Sie sich an SquaredUpund legen Sie kostenlos los.
2. Fügen Sie eine AWS Datenquelle hinzu.
3. Erstellen Sie eine Dashboard-Kachel, die den Timestream Query-Datenstream verwendet.
4. Aktivieren Sie optional die Überwachung für die Kachel, erstellen Sie einen Benachrichtigungskanal oder teilen Sie das Dashboard öffentlich oder privat.
5. Erstellen Sie optional weitere Kacheln, um Ihre Timestream-Daten zusammen mit Daten aus Ihren anderen Überwachungs- und Beobachtungstools zu sehen.

Open-Source-Telegraf

Sie können das Timestream for LiveAnalytics Output-Plugin für Telegraf verwenden, um Metriken direkt aus dem Open-Source-Telegraf in Timestream zu LiveAnalytics schreiben.

In diesem Abschnitt wird erklärt, wie Telegraf mit dem Timestream for Output-Plugin installiert wird, wie Telegraf mit dem Timestream for LiveAnalytics Output-Plugin ausgeführt wird und wie Open Source Telegraf mit Timestream for funktioniert LiveAnalytics . LiveAnalytics

Themen

LiveAnalyticsInstallation von Telegraf mit dem Timestream for Output-Plugin

Ab Version 1.16 ist das Timestream for LiveAnalytics Output-Plugin in der offiziellen Telegraf-Version verfügbar. Um das Output-Plugin auf den meisten gängigen Betriebssystemen zu installieren,

folgen Sie den in der Telegraf-Dokumentation beschriebenen Schritten. InfluxData Folgen Sie zur Installation auf dem Amazon Linux 2-Betriebssystem den nachstehenden Anweisungen.

Installation von Telegraf mit dem Timestream for LiveAnalytics Output-Plugin auf Amazon Linux 2

Gehen Sie wie folgt vor, um Telegraf mit dem Timestream Output Plugin auf Amazon Linux 2 zu installieren.

1. Installieren Sie Telegraf mit dem Paketmanager. yum

```
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. Führen Sie den folgenden Befehl aus.

```
sudo sed -i "s/\$releasever/$(rpm -E %{rhel})/g" /etc/yum.repos.d/influxdb.repo
```

3. Installieren und starten Sie Telegraf.

```
sudo yum install telegraf
sudo service telegraf start
```

Telegraf mit dem Timestream for Output-Plugin ausführen LiveAnalytics

Sie können den nachstehenden Anweisungen folgen, um Telegraf mit dem Timestream for Plugin auszuführen. LiveAnalytics

1. Generieren Sie eine Beispielkonfiguration mit Telegraf.

```
telegraf --section-filter agent:inputs:outputs --input-filter cpu:mem --output-
filter timestream config > example.config
```

2. Erstellen Sie mit der Managementkonsole eine Datenbank in Timestream,, CLIoder. SDKs

3. Fügen Sie in der `example.config` Datei Ihren Datenbanknamen hinzu, indem Sie den folgenden Schlüssel im `[[outputs.timestream]]` Abschnitt bearbeiten.

```
database_name = "yourDatabaseNameHere"
```

4. Standardmäßig erstellt Telegraf eine Tabelle. Wenn Sie eine Tabelle manuell erstellen möchten, stellen Sie `create_table_if_not_exists` diese Option ein `false` und folgen Sie den Anweisungen zum Erstellen einer Tabelle mithilfe der Managementkonsole CLI, oder. SDKs

5. Konfigurieren Sie in der Datei example.config die Anmeldeinformationen im `[[outputs.timestream]]` Abschnitt. Die Anmeldeinformationen sollten die folgenden Operationen ermöglichen.

```
timestream:DescribeEndpoints
timestream:WriteRecords
```

> **Note**
>
> Wenn Sie die `create_table_if_not_exists` Einstellung auf belassen`true`, schließen Sie Folgendes ein:
>
> ```
> timestream:CreateTable
> ```

> **Note**
>
> Wenn Sie `describe_database_on_start` auf einstellen`true`, schließen Sie Folgendes ein.
>
> ```
> timestream:DescribeDatabase
> ```

6. Sie können den Rest der Konfiguration nach Ihren Wünschen bearbeiten.

7. Wenn Sie mit der Bearbeitung der Konfigurationsdatei fertig sind, führen Sie Telegraf wie folgt aus.

```
./telegraf --config example.config
```

8. Die Metriken sollten je nach Ihrer Agentenkonfiguration innerhalb weniger Sekunden angezeigt werden. Sie sollten auch die neuen Tabellen, cpu und mem, in der Timestream-Konsole sehen.

Zuordnung von Telegraf/InfluxDB-Metriken zum Timestream-Modell LiveAnalytics

Beim Schreiben von Daten von Telegraf nach Timestream for werden die Daten wie LiveAnalytics folgt zugeordnet.

- Der Zeitstempel wird als Zeitfeld geschrieben.
- Tags werden als Dimensionen geschrieben.
- Felder werden als Kennzahlen geschrieben.
- Messungen werden meistens als Tabellennamen geschrieben (mehr dazu weiter unten).

Das Timestream for LiveAnalytics Output-Plugin für Telegraf bietet mehrere Optionen zum Organisieren und Speichern von Daten in Timestream for. LiveAnalytics Dies kann anhand eines Beispiels beschrieben werden, das mit den Daten im Zeilenprotokollformat beginnt.

```
weather,location=us-midwest,season=summer temperature=82,humidity=71
1465839830100400200 airquality,location=us-west no2=5,pm25=16
1465839830100400200
```

Im Folgenden werden die Daten beschrieben.

- Die Namen der Messungen lauten `weather` und`airquality`.
- Die Tags sind `location` und`season`.
- Die Felder sind `temperature`und`humidity`,`no2`, und`pm25`.

Themen

- Speichern der Daten in mehreren Tabellen
- Speichern der Daten in einer einzigen Tabelle

Speichern der Daten in mehreren Tabellen

Sie können sich dafür entscheiden, pro Messung eine separate Tabelle zu erstellen und jedes Feld in einer eigenen Zeile pro Tabelle zu speichern.

Die Konfiguration ist`mapping_mode = "multi-table"`.

- Der Timestream for LiveAnalytics Adapter erstellt zwei Tabellen, nämlich `weather` und`airquality`.
- Jede Tabellenzeile wird nur ein einzelnes Feld enthalten.

Der resultierende Timestream für LiveAnalytics Tabellen, `weather` und`airquality`, wird wie folgt aussehen.

`weather`

time	location	Jahreszeit	measure_name	Messwert::bigint
2016-06-13 17:43:50	US-Mittlerer Westen	Sommer	temperature	82
2016-06-13 17:43:50	US-Mittlerer Westen	Sommer	feuchtigkeit	71

`airquality`

time	location	measure_name	Messwert::bigint
2016-06-13 17:43:50	US-Mittlerer Westen	Nr. 2	5
2016-06-13 17:43:50	US-Mittlerer Westen	pm 25	16

Speichern der Daten in einer einzigen Tabelle

Sie können wählen, ob Sie alle Messungen in einer einzigen Tabelle und jedes Feld in einer separaten Tabellenzeile speichern möchten.

Die Konfiguration ist`mapping_mode = "single-table"`. Bei der Verwendung von`single-table`, `single_table_name` und gibt es zwei zusätzliche Konfigurationen`single_table_dimension_name_for_telegraf_measurement_name`.

- Das Timestream for LiveAnalytics Output-Plugin erstellt eine einzelne Tabelle mit dem Namen *`<single_table_name>`* das beinhaltet ein *`<single_table_dimension_name_for_telegraf_measurement_name>`* Spalte.
- Die Tabelle kann mehrere Felder in einer einzigen Tabellenzeile enthalten.

Der resultierende Timestream für die LiveAnalytics Tabelle wird wie folgt aussehen.

`weather`

time	location	Jahreszeit	*<single_table_dimension_name_for_telegraf_measurement_name>*	measure_name	Messwert::bigint
2016-06-13 17:43:50	US-Mittlerer Westen	Sommer	Wetter	temperature	82
2016-06-13 17:43:50	US-Mittlerer Westen	Sommer	Wetter	feuchtigkeit	71
2016-06-13 17:43:50	US-Mittlerer Westen	Sommer	Luftqualität	Nr. 2	5
2016-06-13 17:43:50	US-Mittlerer Westen	Sommer	Wetter	pm25	16

JDBC

Sie können eine Java Database Connectivity (JDBC) -Verbindung verwenden, um Timestream for mit Ihren Business Intelligence-Tools und anderen Anwendungen wie SQLWorkbench LiveAnalytics zu verbinden. Der Timestream for LiveAnalytics JDBC Treiber unterstützt SSO derzeit Okta und Microsoft Azure AD.

Themen

- Konfiguration des JDBC Treibers für Timestream für LiveAnalytics
- Eigenschaften der Verbindung
- JDBCURLBeispiele
- Timestream für die LiveAnalytics JDBC Single Sign-On-Authentifizierung mit Okta einrichten

- Timestream für die LiveAnalytics JDBC Single Sign-On-Authentifizierung mit Microsoft Azure AD einrichten

Konfiguration des JDBC Treibers für Timestream für LiveAnalytics

Gehen Sie wie folgt vor, um den JDBC Treiber zu konfigurieren.

Themen

- Timestream für den Treiber LiveAnalytics JDBC JARs
- Timestream für LiveAnalytics JDBC Treiberklasse und Format URL
- Beispielanwendung

Timestream für den Treiber LiveAnalytics JDBC JARs

Sie können den Timestream für den LiveAnalytics JDBC Treiber direkt herunterladen oder indem Sie den Treiber als Maven-Abhängigkeit hinzufügen.

- Als direkter Download:. Gehen Sie wie folgt vor, um den Timestream for LiveAnalytics JDBC Driver direkt herunterzuladen:

  1. Navigieren Sie zu /releases https://github.com/awslabs/ amazon-timestream-driver-jdbc
  2. Sie können es `amazon-timestream-jdbc-1.0.1-shaded.jar` direkt mit Ihren Business Intelligence-Tools und -Anwendungen verwenden
  3. Laden `amazon-timestream-jdbc-1.0.1-javadoc.jar` Sie es in ein Verzeichnis Ihrer Wahl herunter.
  4. Führen Sie in dem Verzeichnis, in das Sie heruntergeladen haben`amazon-timestream-jdbc-1.0.1-javadoc.jar`, den folgenden Befehl aus, um die Javadoc-Dateien zu extrahieren: HTML

     ```
     jar -xvf amazon-timestream-jdbc-1.0.1-javadoc.jar
     ```

- Als Maven-Abhängigkeit: Gehen Sie wie folgt vor, um den Timestream for LiveAnalytics JDBC Driver als Maven-Abhängigkeit hinzuzufügen:

  1. Navigieren Sie zu der `pom.xml` Datei Ihrer Anwendung und öffnen Sie sie in einem Editor Ihrer Wahl.
  2. Fügen Sie den JDBC Treiber als Abhängigkeit zur `pom.xml` Datei Ihrer Anwendung hinzu:

```
<!-- 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 für LiveAnalytics JDBC Treiberklasse und Format URL

Die Treiberklasse für Timestream for LiveAnalytics JDBC Driver lautet:

```
software.amazon.timestream.jdbc.TimestreamDriver
```

Der JDBC Timestream-Treiber benötigt das folgende Format: JDBC URL

```
jdbc:timestream:
```

Verwenden Sie das folgende URL Format JDBCURL, um Datenbankeigenschaften über die anzugeben:

```
jdbc:timestream://
```

Beispielanwendung

Um Ihnen den Einstieg in die Nutzung von Timestream für LiveAnalytics mit zu erleichternJDBC, haben wir in eine voll funktionsfähige Beispielanwendung erstellt. GitHub

1. Erstellen Sie eine Datenbank mit Beispieldaten, indem Sie den hier beschriebenen Anweisungen folgen.
2. Klonen Sie das GitHub Repository für die Beispielanwendung, um den Anweisungen von zu JDBC folgen GitHub.
3. Folgen Sie den Anweisungen in README, um mit der Beispielanwendung zu beginnen.

Eigenschaften der Verbindung

Der Timestream for LiveAnalytics JDBC Treiber unterstützt die folgenden Optionen:

Themen

> **Note**
>
> Wenn keine der Eigenschaften angegeben wird, verwendet der Timestream for LiveAnalytics JDBC -Treiber die Kette der Standardanmeldedaten, um die Anmeldeinformationen zu laden.

> **Note**
>
> Bei allen Eigenschaftsschlüsseln wird zwischen Groß- und Kleinschreibung unterschieden.

Grundlegende Authentifizierungsoptionen

In der folgenden Tabelle werden die verfügbaren Standardauthentifizierungsoptionen beschrieben.

Option	Beschreibung	Standard
AccessKeyId	Die ID des AWS Benutzerz ugriffsschlüssels.	NONE
SecretAccessKey	Der geheime Zugriffsschlüssel des AWS Benutzers.	NONE
SessionToken	Das temporäre Sitzungstoken, das für den Zugriff auf eine Datenbank mit aktivierter	NONE

Option	Beschreibung	Standard
	Multi-Faktor-Authentifizierung (MFA) erforderlich ist.	

Standardoption für Client-Informationen

In der folgenden Tabelle wird die Option Standard Client Info beschrieben.

Option	Beschreibung	Standard
ApplicationName	Der Name der Anwendung, die derzeit die Verbindung verwendet. `ApplicationName` wird für Debugging-Zwecke verwendet und nicht zur LiveAnalytics Wartung an den Timestream übermittelt.	Der vom Treiber erkannte Anwendungsname.

Option zur Treiberkonfiguration

In der folgenden Tabelle wird die Treiberkonfigurationsoption beschrieben.

Option	Beschreibung	Standard
EnableMetaDataPreparedStatement	Aktiviert Timestream für den LiveAnalytics JDBC Treiber, Metadaten für zurückzugeben`PreparedStatements`, was LiveAnalytics bei Timestream jedoch mit zusätzlichen Kosten für das Abrufen der Metadaten verbunden ist.	FALSE
Region	Die Region der Datenbank.	us-east-1

SDKOption

In der folgenden Tabelle wird die SDK Option beschrieben.

Option	Beschreibung	Standard
RequestTimeout	Die Zeit in Millisekunden, die AWS SDK auf eine Abfragean forderung warten, bevor das Timeout eintritt. Ein nicht positiver Wert deaktiviert das Zeitlimit für Anfragen.	0
SocketTimeout	Die Zeit in Millisekunden, die auf die AWS SDK Übertragu ng von Daten über eine offene Verbindung warten, bevor das Timeout eintritt. Der Wert darf nicht negativ sein. Der Wert `0` deaktiviert das Socket-Ti meout.	50000
MaxRetryCountClient	Die maximale Anzahl von Wiederholungsversuchen bei Fehlern, die wiederholt werden können, mit 5XX Fehlercod es in der. SDK Der Wert darf nicht negativ sein.	NONE
MaxConnections	Die maximale Anzahl gleichzei tig geöffneter HTTP Verbindun gen zum Timestream für den Service. LiveAnalytics Der Wert muss positiv sein.	50

Option zur Konfiguration des Endpunkts

In der folgenden Tabelle wird die Endpunktkonfigurationsoption beschrieben.

Option	Beschreibung	Standard
Endpunkt	Der Endpunkt für den Timestream for LiveAnalytics Service.	NONE

Optionen des Anmeldeinformationsanbieters

In der folgenden Tabelle werden die verfügbaren Credential Provider-Optionen beschrieben.

Option	Beschreibung	Standard
AwsCredentialsProviderClass	Eine von `PropertiesFileCredentialsProvider` oder `InstanceProfileCredentialsProvider`, die für die Authentifizierung verwendet werden soll.	NONE
CustomCredentialsFilePath	Der Pfad zu einer Eigenschaftendatei, die AWS Sicherheitsanmeldedaten `accessKey` und enthält `secretKey`. Dies ist nur erforderlich, wenn `AwsCredentialsProviderClass` als angegeben ist `PropertiesFileCredentialsProvider`.	NONE

SAMLbasierte Authentifizierungsoptionen für Okta

In der folgenden Tabelle werden die verfügbaren SAML basierten Authentifizierungsoptionen für Okta beschrieben.

Option	Beschreibung	Standard
IdpName	Der Name des Identity Providers (Idp), der für die SAML basierte Authentif izierung verwendet werden soll. Einer von `Okta` oder`AzureAD`.	NONE
IdpHost	Der Hostname des angegeben en Idp.	NONE
IdpUserName	Der Benutzername für das angegebene Idp-Konto.	NONE
IdpPassword	Das Passwort für das angegebene Idp-Konto.	NONE
OktaApplicationID	Die eindeutige, von Okta bereitgestellte ID, die dem Timestream für die Anwendung zugeordnet ist. LiveAnalytics `AppId`finden Sie in dem `entityID` Feld, das in den Metadaten der Anwendung angegeben ist. Betrachten Sie das folgende Beispiel: `entityID = http://www.okta.co m//IdpAppID`	NONE
Rolle ARN	Der Amazon-Ressourcenn ame (ARN) der Rolle, die der Aufrufer annimmt.	NONE
Idp ARN	Der Amazon-Ressourcenn ame (ARN) des SAML	NONE

Option	Beschreibung	Standard
	AnbietersIAM, der den Idp beschreibt.	

SAMLbasierte Authentifizierungsoptionen für Azure AD

In der folgenden Tabelle werden die verfügbaren SAML basierten Authentifizierungsoptionen für Azure AD beschrieben.

Option	Beschreibung	Standard
IdpName	Der Name des Identität sanbieters (Idp), der für die SAML basierte Authentif izierung verwendet werden soll. Einer von `Okta` oder`AzureAD`.	NONE
IdpHost	Der Hostname des angegeben en Idp.	NONE
IdpUserName	Der Benutzername für das angegebene Idp-Konto.	NONE
IdpPassword	Das Passwort für das angegebene Idp-Konto.	NONE
AADApplicationID	Die eindeutige ID der registrie rten Anwendung in Azure AD.	NONE
AADClientSecret	Der geheime Client-Sc hlüssel, der der registrie rten Anwendung in Azure AD zugeordnet ist und zum Autorisieren des Abrufs von Token verwendet wird.	NONE

Option	Beschreibung	Standard
AADTenant	Die Azure AD-Mandanten-ID.	NONE
Idp ARN	Der Amazon-Ressourcenn ame (ARN) des SAML AnbietersIAM, der den Idp beschreibt.	NONE

JDBCURLBeispiele

In diesem Abschnitt wird beschrieben, wie eine JDBC Verbindung hergestellt wirdURL, und es werden Beispiele bereitgestellt. Verwenden Sie das folgende URL Format, um die optionalen Verbindungseigenschaften anzugeben:

```
jdbc:timestream://PropertyName1=value1;PropertyName2=value2...
```

> **Note**
>
> Alle Verbindungseigenschaften sind optional. Bei allen Eigenschaftsschlüsseln wird zwischen Groß- und Kleinschreibung unterschieden.

Im Folgenden finden Sie einige JDBC URLs Verbindungsbeispiele.

Beispiel mit grundlegenden Authentifizierungsoptionen und Region:

```
jdbc:timestream://
AccessKeyId=<myAccessKeyId>;SecretAccessKey=<mySecretAccessKey>;SessionToken=<mySessionToken
east-1
```

Beispiel mit Kundeninformationen, Region und SDK Optionen:

```
jdbc:timestream://ApplicationName=MyApp;Region=us-
east-1;MaxRetryCountClient=10;MaxConnections=5000;RequestTimeout=20000
```

Stellen Sie eine Verbindung her, indem Sie die standardmäßige Anbieterkette für Anmeldeinformationen verwenden, wobei die AWS Anmeldeinformationen in Umgebungsvariablen festgelegt sind:

```
jdbc:timestream
```

Connect mithilfe der standardmäßigen Anbieterkette für Anmeldeinformationen eine Verbindung her, wobei die AWS Anmeldeinformationen in der Verbindung festgelegt sind: URL

```
jdbc:timestream://
AccessKeyId=<myAccessKeyId>;SecretAccessKey=<mySecretAccessKey>;SessionToken=<mySessionToken
```

Connect mit der PropertiesFileCredentialsProvider als Authentifizierungsmethode her:

```
jdbc:timestream://
AwsCredentialsProviderClass=PropertiesFileCredentialsProvider;CustomCredentialsFilePath=<pat
 to properties file>
```

Connect mit der InstanceProfileCredentialsProvider als Authentifizierungsmethode her:

```
jdbc:timestream://AwsCredentialsProviderClass=InstanceProfileCredentialsProvider
```

Connect mit den Okta-Anmeldeinformationen als Authentifizierungsmethode her:

```
jdbc:timestream://
IdpName=Okta;IdpHost=<host>;IdpUserName=<name>;IdpPassword=<password>;OktaApplicationID=<id>
```

Connect mit den Azure AD-Anmeldeinformationen als Authentifizierungsmethode her:

```
jdbc:timestream://
IdpName=AzureAD;IdpUserName=<name>;IdpPassword=<password>;AADApplicationID=<id>;AADClientSec
```

Stellen Sie eine Connect mit einem bestimmten Endpunkt her:

```
jdbc:timestream://Endpoint=abc.us-east-1.amazonaws.com;Region=us-east-1
```

Timestream für die LiveAnalytics JDBC Single Sign-On-Authentifizierung mit Okta einrichten

Timestream for LiveAnalytics unterstützt Timestream für die LiveAnalytics JDBC Single Sign-On-Authentifizierung mit Okta. Um Timestream für die LiveAnalytics JDBC Single Sign-On-Authentifizierung mit Okta zu verwenden, füllen Sie jeden der unten aufgeführten Abschnitte aus.

Themen

Voraussetzungen

Stellen Sie sicher, dass Sie die folgenden Voraussetzungen erfüllt haben, bevor Sie Timestream für die LiveAnalytics JDBC Single Sign-On-Authentifizierung mit Okta verwenden:

- Administratorberechtigungen AWS zum Erstellen des Identitätsanbieters und der Rollen.
- Ein Okta-Konto (Gehen Sie zu, https://www.okta.com/login/um ein Konto zu erstellen).
- Zugriff auf Amazon Timestream für LiveAnalytics.

Nachdem Sie die Voraussetzungen erfüllt haben, können Sie fortfahrenAWS Kontenverbund in Okta.

AWS Kontenverbund in Okta

Der Timestream for LiveAnalytics JDBC Driver unterstützt den AWS Kontoverbund in Okta. Gehen Sie wie folgt vor, um den AWS Kontoverbund in Okta einzurichten:

1. Melden Sie sich wie folgt im Okta Admin-Dashboard an: URL

```
https://<company-domain-name>-admin.okta.com/admin/apps/active
```

> **Note**
>
> Ersetzen Sie < company-domain-name > durch Ihren Domainnamen.

2. Wählen Sie nach erfolgreicher Anmeldung „Anwendung hinzufügen“ und suchen Sie nach AWS Account Federation.
3. Wählen Sie Hinzufügen
4. Ändern Sie das Login URL auf das entsprechendeURL.
5. Wählen Sie Weiter
6. Wählen Sie SAML2.0 als Anmeldemethode
7. Wählen Sie Identity Provider-Metadaten aus, um die XML Metadatendatei zu öffnen. Speichern Sie die Datei lokal.
8. Lassen Sie alle anderen Konfigurationsoptionen leer.
9. Wählen Sie Erledigt aus.

Nachdem Sie den AWS Kontoverbund in Okta abgeschlossen haben, können Sie mit dem Vorgang fortfahrenOkta einrichten für SAML.

Okta einrichten für SAML

1. Wählen Sie die Registerkarte Sign On (Anmelden) aus. Wählen Sie die Ansicht.
2. Wählen Sie im Bereich „Einstellungen“ die Schaltfläche „Anweisungen zur Einrichtung“.

Das Okta-Metadatendokument finden

1. Um das Dokument zu finden, gehen Sie zu:

```
https://<domain>-admin.okta.com/admin/apps/active
```

> **Note**
>
> <domain>ist Ihr eindeutiger Domainname für Ihr Okta-Konto.

2. Wählen Sie die AWS Account Federation-Anwendung
3. Wählen Sie den Tab „Anmelden“

Timestream für die LiveAnalytics JDBC Single Sign-On-Authentifizierung mit Microsoft Azure AD einrichten

Timestream for LiveAnalytics unterstützt Timestream für die LiveAnalytics JDBC Single Sign-On-Authentifizierung mit Microsoft Azure AD. Um Timestream für die LiveAnalytics JDBC Single Sign-On-Authentifizierung mit Microsoft Azure AD zu verwenden, füllen Sie jeden der unten aufgeführten Abschnitte aus.

Themen

Voraussetzungen

Stellen Sie sicher, dass Sie die folgenden Voraussetzungen erfüllt haben, bevor Sie den Timestream für die LiveAnalytics JDBC Single Sign-On-Authentifizierung mit Microsoft Azure AD verwenden:

- Administratorberechtigungen AWS zum Erstellen des Identitätsanbieters und der Rollen.
- Ein Azure Active Directory-Konto (Gehen Sie zu https://azure.microsoft.com/en-ca/services/active-directory/, um ein Konto zu erstellen)
- Zugriff auf Amazon Timestream für LiveAnalytics.

Azure AD einrichten

1. Melden Sie sich beim Azure-Portal an
2. Wählen Sie Azure Active Directory in der Liste der Azure-Dienste aus. Dadurch wird zur Seite mit dem Standardverzeichnis weitergeleitet.
3. Wählen Sie in der Seitenleiste im Bereich Verwalten die Option Unternehmensanwendungen
4. Wählen Sie + Neue Anwendung.
5. Suchen und wählen Sie Amazon Web Services aus.
6. Wählen Sie Single Sign-On im Bereich Verwalten in der Seitenleiste
7. Wählen Sie SAMLals Single-Sign-On-Methode
8. Geben Sie im Abschnitt SAML Grundkonfiguration Folgendes sowohl URL für den Identifier als auch für die Antwort URL ein:

```
https://signin.aws.amazon.com/saml
```

9. Wählen Sie Speichern.
10. Laden Sie die Verbund-Metadaten XML im Abschnitt „SAMLSignaturzertifikat“ herunter. Dies wird später bei der Erstellung des IAM Identity Providers verwendet
11. Kehren Sie zur Standardverzeichnisseite zurück und wählen Sie unter Verwalten die Option App-Registrierungen aus.
12. Wählen Sie im Abschnitt Alle Anwendungen die Option Timestream für LiveAnalytics aus. Die Seite wird zur Übersichtsseite der Anwendung weitergeleitet

> **Note**
>
> Notieren Sie sich die Anwendungs-ID (Client) und die Verzeichnis-ID (Mandanten-ID). Diese Werte sind erforderlich, wenn eine Verbindung hergestellt wird.

13. Wählen Sie Certificates and Secrets
14. Erstellen Sie unter Kundengeheimnisse einen neuen geheimen Kundenschlüssel mit + Neuer geheimer Clientschlüssel.

> **Note**
>
> Notieren Sie sich das generierte Client-Geheimnis, da dies erforderlich ist, wenn Sie eine Verbindung zu Timestream für LiveAnalytics herstellen.

15. Wählen Sie in der Seitenleiste unter Verwalten die Option Berechtigungen aus API
16. Verwenden Sie unter Konfigurierte Berechtigungen die Option Berechtigung hinzufügen, um Azure AD die Berechtigung zu erteilen, sich bei Timestream anzumelden für. LiveAnalytics Wählen Sie auf der Seite API Berechtigungen anfordern die Option Microsoft Graph aus.
17. Wählen Sie Delegierte Berechtigungen und anschließend die Berechtigung User.Read
18. Wählen Sie „Berechtigungen hinzufügen“
19. Wählen Sie Administratorzustimmung erteilen für Standardverzeichnis

Einrichtung des IAM Identitätsanbieters und der Rollen in AWS

Füllen Sie jeden der folgenden Abschnitte aus, um Timestream IAM für die LiveAnalytics JDBC Single Sign-On-Authentifizierung mit Microsoft Azure AD einzurichten:

Themen

- Erstellen Sie einen Identitätsanbieter SAML
- Erstellen Sie eine IAM-Rolle
- Erstellen Sie eine IAM Richtlinie
- Bereitstellung

Erstellen Sie einen Identitätsanbieter SAML

Gehen Sie wie folgt vor, um einen SAML Identitätsanbieter für den Timestream für die LiveAnalytics JDBC Single Sign-On-Authentifizierung mit Microsoft Azure AD zu erstellen:

1. Melden Sie sich bei der Management Console an AWS
2. Wählen Sie Dienste und dann IAMunter Sicherheit, Identität und Compliance
3. Wählen Sie unter Zugriffsverwaltung die Option Identitätsanbieter aus
4. Wählen Sie Create Provider und wählen Sie SAMLals Anbietertyp. Geben Sie den Namen des Anbieters ein. In diesem Beispiel wird A verwendetzureADProvider.
5. Laden Sie die zuvor heruntergeladene XML Federation-Metadatendatei hoch
6. Wählen Sie „Weiter“ und anschließend „Erstellen“.
7. Nach Abschluss wird die Seite zurück zur Seite mit den Identitätsanbietern umgeleitet

Erstellen Sie eine IAM-Rolle

Gehen Sie wie folgt vor, um eine IAM Rolle für den Timestream für die LiveAnalytics JDBC Single Sign-On-Authentifizierung mit Microsoft Azure AD zu erstellen:

1. Wählen Sie in der Seitenleiste unter Zugriffsverwaltung die Option Rollen aus
2. Wählen Sie Create role (Rolle erstellen) aus
3. Wählen Sie SAML2.0 Federation als vertrauenswürdige Entität
4. Wählen Sie den Azure AD-Anbieter

5. Wählen Sie Programmatischen Zugriff und Zugriff auf die AWS Managementkonsole zulassen
6. Wählen Sie Next: Permissions (Weiter: Berechtigungen) aus
7. Hängen Sie Berechtigungsrichtlinien an oder fahren Sie mit Weiter:Tags fort
8. Fügen Sie optionale Tags hinzu oder fahren Sie mit Next:Review fort
9. Geben Sie einen Role name ein. In diesem Beispiel wird A verwendet zureSAMLRole
10. Geben Sie eine Rollenbeschreibung an
11. Wählen Sie „Rolle erstellen“, um den Vorgang abzuschließen

Erstellen Sie eine IAM Richtlinie

Gehen Sie wie folgt vor, um eine IAM Richtlinie für den Timestream für die LiveAnalytics JDBC Single Sign-On-Authentifizierung mit Microsoft Azure AD zu erstellen:

1. Wählen Sie in der Seitenleiste unter Zugriffsverwaltung die Option Richtlinien aus
2. Wählen Sie Richtlinie erstellen und wählen Sie den Tab JSON
3. Fügen Sie die folgende Richtlinie hinzu

```
{
"Version": "2012-10-17",
"Statement": [
    {
          "Effect": "Allow",
          "Action": [
                "iam:ListRoles",
                "iam:ListAccountAliases"
           ],
           "Resource": "*"
      }
]
}
```

4. Wählen Sie Richtlinie erstellen aus
5. Geben Sie den Namen einer Richtlinie ein. In diesem Beispiel wird verwendet TimestreamAccessPolicy.
6. Wählen Sie Create Policy
7. Wählen Sie in der Seitenleiste unter Zugriffsverwaltung die Option Rollen aus.

8. Wählen Sie die zuvor erstellte Azure AD-Rolle aus und klicken Sie unter Berechtigungen auf Richtlinien anhängen.
9. Wählen Sie die zuvor erstellte Zugriffsrichtlinie aus.

Bereitstellung

Gehen Sie wie folgt vor, um den Identitätsanbieter für Timestream für die LiveAnalytics JDBC Single Sign-On-Authentifizierung mit Microsoft Azure AD bereitzustellen:

1. Gehen Sie zurück zum Azure-Portal
2. Wählen Sie Azure Active Directory in der Liste der Azure-Dienste aus. Dadurch wird zur Seite mit dem Standardverzeichnis weitergeleitet
3. Wählen Sie in der Seitenleiste im Bereich Verwalten die Option Unternehmensanwendungen
4. Wählen Sie Provisioning
5. Wählen Sie den automatischen Modus für die Bereitstellungsmethode
6. Geben Sie unter Admin-Anmeldeinformationen Ihre AwsAccessKeyID für clientsecret und SecretAccessKeyfür Secret Token ein
7. Setzen Sie den Bereitstellungsstatus auf Ein
8. Wählen Sie „Speichern“. Dadurch kann Azure AD die erforderlichen IAM Rollen laden
9. Sobald der Status des aktuellen Zyklus abgeschlossen ist, wählen Sie in der Seitenleiste Benutzer und Gruppen aus
10. Wählen Sie + Benutzer hinzufügen
11. Wählen Sie den Azure AD-Benutzer aus, für den Sie Zugriff auf Timestream gewähren möchten LiveAnalytics
12. Wählen Sie die IAM Azure AD-Rolle und den entsprechenden Azure Identity Provider aus, der in erstellt wurde AWS
13. Wählen Sie Zuweisen

ODBC

Der ODBCOpen-Source-Treiber für Amazon Timestream for LiveAnalytics bietet Entwicklern eine SQL relationale Schnittstelle zu Timestream und ermöglicht Konnektivität über Business Intelligence (BI) -Tools wie Power BI Desktop und Microsoft Excel. LiveAnalytics Der Timestream for

LiveAnalytics ODBC Driver ist derzeit unter Windows, macOS und Linux verfügbar und unterstützt SSO auch Okta und Microsoft Azure Active Directory (AD).

Weitere Informationen finden Sie in der LiveAnalyticsODBCTreiberdokumentation zu Amazon Timestream. GitHub

Themen

- Timestream für den Treiber einrichten LiveAnalytics ODBC
- Syntax und Optionen für die Verbindungszeichenfolge für den ODBC Treiber
- Beispiele für Verbindungszeichenfolgen für den Timestream for Driver LiveAnalytics ODBC
- Problembehebung bei der Verbindung mit dem ODBC Treiber

Timestream für den Treiber einrichten LiveAnalytics ODBC

Richten Sie den Zugriff auf Timestream für LiveAnalytics in Ihrem Konto ein AWS

Wenn Sie Ihr AWS Konto noch nicht für die Arbeit mit Timestream for eingerichtet haben LiveAnalytics, folgen Sie den Anweisungen unter. Zugreifen auf Timestream für LiveAnalytics

Installieren Sie den ODBC Treiber auf Ihrem System

Laden Sie das entsprechende ODBC Timestream-Treiber-Installationsprogramm für Ihr System aus dem ODBC GitHubRepository herunter, und folgen Sie den Installationsanweisungen, die für Ihr System gelten:.

- Windows-Installationsanleitung
- macOS-Installationsanleitung
- Installationsanleitung für Linux

Richten Sie einen Datenquellennamen (DSN) für den ODBC Treiber ein

Folgen Sie den Anweisungen in der DSN Konfigurationsanleitung für Ihr System:

- DSNWindows-Konfiguration
- DSNmacOS-Konfiguration
- DSNLinux-Konfiguration

Richten Sie Ihre Business Intelligence (BI) -Anwendung so ein, dass sie mit dem ODBC Treiber funktioniert

Im Folgenden finden Sie Anweisungen zum Einrichten mehrerer gängiger BI-Anwendungen für die Verwendung mit dem ODBC Treiber:

- Microsoft Power BI einrichten.
- Microsoft Excel einrichten
- Tableau einrichten

Für andere Anwendungen

Syntax und Optionen für die Verbindungszeichenfolge für den ODBC Treiber

Die Syntax für die Angabe von Verbindungszeichenfolgenoptionen für den ODBC Treiber lautet wie folgt:

```
DRIVER={Amazon Timestream ODBC Driver};(option)=(value);
```

Die verfügbaren Optionen lauten wie folgt:

Verbindungsoptionen für den Treiber

- **`Driver`**(erforderlich) — Der Treiber, mit dem verwendet wirdODBC.

  Die Standardeinstellung ist Amazon Timestream.
- **`DSN`**— Der Datenquellenname (DSN), der für die Konfiguration der Verbindung verwendet werden soll.

  Der Standardwert ist `NONE`.
- **`Auth`**— Der Authentifizierungsmodus. Dies muss einer der folgenden sein:
  - `AWS_PROFILE`— Verwenden Sie die standardmäßige Anmeldeinformationskette.
  - `IAM`— Verwenden Sie AWS IAM Anmeldeinformationen.
  - `AAD`— Verwenden Sie den Azure Active Directory (AD) -Identitätsanbieter.
  - `OKTA`— Verwenden Sie den Okta-Identitätsanbieter.

  Der Standardwert ist `AWS_PROFILE`.

Optionen für die Konfiguration von Endpunkten

- **`EndpointOverride`**— Die Endpunkt-Überschreibung für den Timestream for LiveAnalytics Service. Dies ist eine erweiterte Option, die die Region überschreibt. Beispielsweise:

  ```
  query-cell2.timestream.us-east-1.amazonaws.com
  ```

- **`Region`**— Die Signaturregion für den Timestream for LiveAnalytics Service-Endpunkt.

  Der Standardwert ist `us-east-1`.

Option für den Anbieter von Anmeldeinformationen

- **`ProfileName`**— Der Profilname in der AWS Konfigurationsdatei.

  Der Standardwert ist `NONE`.

AWS IAMAuthentifizierungsoptionen

- **`UID`**oder **`AccessKeyId`**— Die ID des AWS Benutzerzugriffsschlüssels. Wenn `UID` sowohl als auch in der Verbindungszeichenfolge angegeben `AccessKeyId` werden, wird der `UID` Wert verwendet, sofern er nicht leer ist.

  Der Standardwert ist `NONE`.

- **`PWD`**oder **`SecretKey`**— Der geheime Zugriffsschlüssel des AWS Benutzers. Wenn `PWD` sowohl als auch in der Verbindungszeichenfolge angegeben `SecretKey` werden, wird der `PWD` Wert mit verwendet, sofern er nicht leer ist.

  Der Standardwert ist `NONE`.

- **`SessionToken`**— Das temporäre Sitzungstoken, das für den Zugriff auf eine Datenbank mit aktivierter Multi-Faktor-Authentifizierung (MFA) erforderlich ist. Fügen Sie kein abschließendes Zeichen `=` in die Eingabe ein.

  Der Standardwert ist `NONE`.

SAMLbasierte Authentifizierungsoptionen für Okta

- **`IdPHost`**— Der Hostname des angegebenen IdP.

Der Standardwert ist NONE.

- **UID**oder **IdPUserName**— Der Benutzername für das angegebene IdP-Konto. Wenn beide UID und in der Verbindungszeichenfolge angegeben IdPUserName werden, wird der UID Wert verwendet, sofern er nicht leer ist.

  Der Standardwert ist NONE.

- **PWD**oder **IdPPassword**— Das Passwort für das angegebene IdP-Konto. Wenn PWD sowohl als auch in der Verbindungszeichenfolge angegeben IdPPassword werden, wird der PWD Wert verwendet, sofern er nicht leer ist.

  Der Standardwert ist NONE.

- **OktaApplicationID**— Die eindeutige von Okta bereitgestellte ID, die dem Timestream für die Anwendung zugeordnet ist. LiveAnalytics Die Anwendungs-ID (AppId) finden Sie in dem entityID Feld, das in den Anwendungsmetadaten angegeben ist. Ein Beispiel ist:

  ```
  entityID="http://www.okta.com//(IdPAppID)
  ```

  Der Standardwert ist NONE.

- **RoleARN**— Der Amazon-Ressourcenname (ARN) der Rolle, die der Aufrufer annimmt.

  Der Standardwert ist NONE.

- **IdPARN**— Der Amazon-Ressourcenname (ARN) des SAML AnbietersIAM, der den IdP beschreibt.

  Der Standardwert ist NONE.

SAMLbasierte Authentifizierungsoptionen für Azure Active Directory

- **UID**oder **IdPUserName**— Der Benutzername für das angegebene IdP-Konto..

  Der Standardwert ist NONE.

- **PWD**oder **IdPPassword**— Das Passwort für das angegebene IdP-Konto.

  Der Standardwert ist NONE.

- **AADApplicationID**— Die eindeutige ID der registrierten Anwendung in Azure AD.

  Der Standardwert ist NONE.

- **`AADClientSecret`**— Das Client-Geheimnis, das der registrierten Anwendung in Azure AD zugeordnet ist und zum Autorisieren des Abrufs von Token verwendet wird.

  Der Standardwert ist `NONE`.
- **`AADTenant`**— Die Azure AD-Mandanten-ID.

  Der Standardwert ist `NONE`.
- **`RoleARN`**— Der Amazon-Ressourcenname (ARN) der Rolle, die der Aufrufer annimmt.

  Der Standardwert ist `NONE`.
- **`IdPARN`**— Der Amazon-Ressourcenname (ARN) des SAML AnbietersIAM, der den IdP beschreibt.

  Der Standardwert ist `NONE`.

AWS SDK(erweiterte) Optionen

- **`RequestTimeout`**— Die Zeit in Millisekunden, die auf eine Abfrageanforderung AWS SDK wartet, bevor das Timeout eintritt. Jeder nicht positive Wert deaktiviert das Anforderungs-Timeout.

  Der Standardwert ist `3000`.
- **`ConnectionTimeout`**— Die Zeit in Millisekunden, die auf die Übertragung von Daten über eine AWS SDK offene Verbindung wartet, bevor das Timeout eintritt. Ein Wert von 0 deaktiviert das Verbindungs-Timeout. Dieser Wert darf nicht negativ sein.

  Der Standardwert ist `1000`.
- **`MaxRetryCountClient`**— Die maximale Anzahl von Wiederholungsversuchen bei wiederholbaren Fehlern mit 5xx Fehlercodes im. SDK Der Wert darf nicht negativ sein.

  Der Standardwert ist `0`.
- **`MaxConnections`**— Die maximale Anzahl von erlaubten gleichzeitig geöffneten HTTP Verbindungen zum Timestream-Dienst. Der Wert muss positiv sein.

  Der Standardwert ist `25`.

ODBCOptionen für die Treiberprotokollierung

- **`LogLevel`**— Die Protokollebene für die Treiberprotokollierung. Zulässige Werte sind:
  - (0OFF).

- (1ERROR).
- (2WARNING).
- (3INFO).
- (4DEBUG).

  Die Standardeinstellung ist 1 (ERROR).

  Warnung: Wenn der Fahrer den DEBUG Protokollierungsmodus verwendet, können persönliche Daten protokolliert werden.

- **`LogOutput`**— Ordner, in dem die Protokolldatei gespeichert werden soll.

  Die Standardeinstellung ist:

  - Windows:`%USERPROFILE%`, oder falls nicht verfügbar,`%HOMEDRIVE%%HOMEPATH%`.
  - macOS und Linux: `$HOME` oder falls nicht verfügbar, das Feld `pw_dir` aus dem `getpwuid(getuid())` Rückgabewert der Funktion.

SDKOptionen für die Protokollierung

Die AWS SDK Protokollebene ist getrennt von der LiveAnalytics ODBC Timestream-Protokollebene für Treiber. Die Einstellung des einen hat keinen Einfluss auf das andere.

Die SDK Protokollebene wird mithilfe der Umgebungsvariablen festgelegt`TS_AWS_LOG_LEVEL`. Gültige Werte für sind:

- `OFF`
- `ERROR`
- `WARN`
- `INFO`
- `DEBUG`
- `TRACE`
- `FATAL`

Wenn nicht festgelegt, `TS_AWS_LOG_LEVEL` wird die SDK Protokollebene auf den Standardwert gesetzt, der ist`WARN`.

Verbindung über einen Proxy herstellen

Der ODBC Treiber unterstützt die Verbindung zu Amazon Timestream LiveAnalytics über einen Proxy. Um diese Funktion zu verwenden, konfigurieren Sie die folgenden Umgebungsvariablen basierend auf Ihrer Proxyeinstellung:

- **TS_PROXY_HOST**— der Proxy-Host.
- **TS_PROXY_PORT**— Die Proxy-Portnummer.
- **TS_PROXY_SCHEME**— Das Proxyschema, entweder `http` oder`https`.
- **TS_PROXY_USER**— Der Benutzername für die Proxyauthentifizierung.
- **TS_PROXY_PASSWORD**— Das Benutzerkennwort für die Proxyauthentifizierung.
- **TS_PROXY_SSL_CERT_PATH**— Die SSL Zertifikatsdatei, die für die Verbindung zu einem HTTPS Proxy verwendet werden soll.
- **TS_PROXY_SSL_CERT_TYPE**— Der Typ des SSL Proxy-Client-Zertifikats.
- **TS_PROXY_SSL_KEY_PATH**— Die private Schlüsseldatei, die für die Verbindung zu einem HTTPS Proxy verwendet werden soll.
- **TS_PROXY_SSL_KEY_TYPE**— Der Typ der privaten Schlüsseldatei, die für die Verbindung zu einem HTTPS Proxy verwendet wird.
- **TS_PROXY_SSL_KEY_PASSWORD**— Die Passphrase für die private Schlüsseldatei, die für die Verbindung zu einem Proxy verwendet wird. HTTPS

Beispiele für Verbindungszeichenfolgen für den Timestream for Driver LiveAnalytics ODBC

Beispiel für eine Verbindung zum ODBC Treiber mit Anmeldeinformationen IAM

```
Driver={Amazon Timestream ODBC Driver};Auth=IAM;AccessKeyId=(your access key
 ID);secretKey=(your secret key);SessionToken=(your session token);Region=us-east-2;
```

Beispiel für eine Verbindung mit dem ODBC Treiber über ein Profil

```
Driver={Amazon Timestream ODBC Driver};ProfileName=(the profile name);region=us-west-2;
```

Der Treiber versucht, mit den in der Umgebungsvariablen angegebenen Anmeldeinformationen oder`~/.aws/credentials`, falls eine Datei in der Umgebungsvariablen angegeben

istAWS_SHARED_CREDENTIALS_FILE, mithilfe der Anmeldeinformationen in dieser Datei eine Verbindung herzustellen.

Beispiel für eine Verbindung zum ODBC Treiber mit 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);
```

Beispiel für eine Verbindung zum ODBC Treiber mit 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);
```

Beispiel für eine Verbindung zum ODBC Treiber mit einem angegebenen Endpunkt und einer Protokollebene von 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;
```

Problembehebung bei der Verbindung mit dem ODBC Treiber

> **Note**
>
> Wenn der Benutzername und das Passwort bereits in der angegeben sindDSN, müssen sie nicht erneut angegeben werden, wenn der ODBC Treibermanager danach fragt.

Der Fehlercode `01S02` mit einer Meldung `Re-writing` *`(connection string option)`* `(have you specified it several times?` tritt auf, wenn eine Verbindungszeichenfolgenoption in der Verbindungszeichenfolge mehr als einmal übergeben wird. Wenn Sie eine Option mehr als einmal angeben, wird ein Fehler ausgelöst. Wenn Sie eine Verbindung mit einer DSN und einer Verbindungszeichenfolge herstellen und eine Verbindungsoption bereits in der angegeben istDSN, geben Sie sie nicht erneut in der Verbindungszeichenfolge an.

VPCEndpunkte ()AWS PrivateLink

Sie können eine private Verbindung zwischen Ihrem VPC und Amazon Timestream für einrichten, LiveAnalytics indem Sie einen VPCSchnittstellenendpunkt erstellen. Weitere Informationen finden Sie unter VPCEndpunkte ()AWS PrivateLink.

Bewährte Methoden

Um die Vorteile von Amazon Timestream for voll auszuschöpfen LiveAnalytics, befolgen Sie die unten beschriebenen Best Practices.

> **Note**
>
> Wenn Sie proof-of-concept Anwendungen ausführen, sollten Sie bei der Bewertung der Leistung und des Umfangs von Timestream die Datenmenge berücksichtigen, die sich in einigen Monaten oder Jahren ansammeln wird. LiveAnalytics Wenn Ihre Daten im Laufe der Zeit wachsen, werden Sie feststellen, dass die Leistung von Timestream for weitgehend unverändert LiveAnalytics bleibt, da die serverlose Architektur enorme Mengen an Parallelität für die Verarbeitung größerer Datenmengen nutzen und automatisch skalieren kann, um den Anforderungen Ihrer Anwendung gerecht zu werden.

Themen

Datenmodellierung

Amazon Timestream for LiveAnalytics wurde entwickelt, um Zeitreihendaten von Anwendungen und Geräten zu sammeln, zu speichern und zu analysieren, die eine Datensequenz mit einem Zeitstempel

ausgeben. Für eine optimale Leistung LiveAnalytics müssen die an Timestream gesendeten Daten zeitliche Merkmale aufweisen, und Zeit muss ein wesentlicher Bestandteil der Daten sein.

Timestream for LiveAnalytics bietet Ihnen die Flexibilität, Ihre Daten auf unterschiedliche Weise zu modellieren, um sie an die Anforderungen Ihrer Anwendung anzupassen. In diesem Abschnitt behandeln wir mehrere dieser Muster und stellen Ihnen Richtlinien zur Verfügung, mit denen Sie Ihre Kosten und Leistung optimieren können. Machen Sie sich mit den wichtigsten LiveAnalytics Timestream-Konzepten wie Dimensionen und Kennzahlen vertraut. In diesem Abschnitt erfahren Sie mehr über:

Beachten Sie bei der Entscheidung, ob Sie eine einzelne Tabelle oder mehrere Tabellen zum Speichern von Daten erstellen möchten, Folgendes:

- Welche Daten in dieselbe Tabelle aufgenommen werden sollen und wann Sie Daten auf mehrere Tabellen und Datenbanken verteilen möchten.
- Wie Sie zwischen Timestream für Datensätze mit LiveAnalytics mehreren Messwerten und Datensätzen mit nur einer Messung wählen können und welche Vorteile die Modellierung mit Datensätzen mit mehreren Messwerten bietet, insbesondere wenn Ihre Anwendung mehrere Messungen gleichzeitig und sofort verfolgt.
- Welche Attribute sollen als Dimensionen oder als Kennzahlen modelliert werden?
- So verwenden Sie die Kennzahlnamenattribute effektiv, um Ihre Abfragelatenz zu optimieren.

Themen

Einzelne Tabelle im Vergleich zu mehreren Tabellen

Bei der Modellierung Ihrer Daten in der Anwendung ist ein weiterer wichtiger Aspekt die Modellierung der Daten in Tabellen und Datenbanken. Datenbanken und Tabellen in Timestream for LiveAnalytics sind Abstraktionen für die Zugriffskontrolle, die Angabe von KMS Schlüsseln, Aufbewahrungsfristen usw. Timestream dient zur LiveAnalytics automatischen Partitionierung Ihrer Daten und ist darauf

ausgelegt, Ressourcen so zu skalieren, dass sie der Aufnahme-, Speicher- und Abfragelast sowie den Anforderungen Ihrer Anwendungen entsprechen.

Eine Tabelle in Timestream for LiveAnalytics kann auf Petabyte an gespeicherten Daten, mehrere zehn Gigabyte/Sekunde an Datenschreibvorgängen skaliert werden, und Abfragen können Hunderte von Daten pro Stunde verarbeiten. TBs Abfragen in Timestream for LiveAnalytics können sich über mehrere Tabellen und Datenbanken erstrecken und bieten Verknüpfungen und Unions, um einen nahtlosen Zugriff auf Ihre Daten über mehrere Tabellen und Datenbanken hinweg zu ermöglichen. Daher sind der Umfang der Daten oder das Anforderungsvolumen in der Regel nicht das Hauptaugenmerk bei der Entscheidung, wie Ihre Daten in Timestream für organisiert werden sollen. LiveAnalytics Im Folgenden finden Sie einige wichtige Überlegungen bei der Entscheidung, welche Daten in derselben Tabelle, in verschiedenen Tabellen oder Tabellen in verschiedenen Datenbanken zusammengefasst werden sollen.

- Datenaufbewahrungsrichtlinien (Aufbewahrung von Speicherspeichern, Aufbewahrung von Magnetspeichern usw.) werden bei der Granularität einer Tabelle unterstützt. Daher müssen sich Daten, für die unterschiedliche Aufbewahrungsrichtlinien gelten, in unterschiedlichen Tabellen befinden.
- AWS KMS Schlüssel, die zur Verschlüsselung Ihrer Daten verwendet werden, werden auf Datenbankebene konfiguriert. Daher bedeuten unterschiedliche Anforderungen an die Verschlüsselungsschlüssel, dass sich die Daten in unterschiedlichen Datenbanken befinden müssen.
- Timestream for LiveAnalytics unterstützt ressourcenbasierte Zugriffskontrolle mit der Granularität von Tabellen und Datenbanken. Berücksichtigen Sie Ihre Anforderungen an die Zugriffskontrolle, wenn Sie entscheiden, welche Daten Sie in dieselbe Tabelle schreiben und welche in verschiedene Tabellen.
- Beachten Sie bei der Entscheidung, welche Daten in welcher Tabelle gespeichert werden, die Beschränkungen in Bezug auf die Anzahl der Dimensionen, Kennzahlnamen und Namen von Attributen mit mehreren Kennzahlen.
- Berücksichtigen Sie bei der Entscheidung, wie Sie Ihre Daten organisieren, Ihre Abfrageauslastung und Ihre Zugriffsmuster, da die Abfragelatenz und das einfache Schreiben Ihrer Abfragen davon abhängen.
  - Wenn Sie Daten, die Sie häufig abfragen, in derselben Tabelle speichern, erleichtert dies im Allgemeinen das Schreiben Ihrer Abfragen, sodass Sie häufig vermeiden können, Verknüpfungen, Vereinigungen oder allgemeine Tabellenausdrücke schreiben zu müssen. Dies führt in der Regel auch zu einer geringeren Abfragelatenz. Sie können Prädikate für

Dimensionen und Kennzahlnamen verwenden, um die für die Abfragen relevanten Daten zu filtern.

Stellen Sie sich zum Beispiel einen Fall vor, in dem Sie Daten von Geräten auf sechs Kontinenten speichern. Wenn Ihre Abfragen häufig auf Daten von verschiedenen Kontinenten zugreifen, um eine globale aggregierte Ansicht zu erhalten, erleichtert das Speichern von Daten aus diesen Kontinenten in derselben Tabelle das Schreiben von Abfragen. Wenn Sie dagegen Daten in verschiedenen Tabellen speichern, können Sie die Daten immer noch in derselben Abfrage kombinieren. Sie müssen jedoch eine Abfrage schreiben, um die Daten aus verschiedenen Tabellen zu vereinigen.

- Timestream for LiveAnalytics verwendet adaptive Partitionierung und Indizierung Ihrer Daten. Abfragen werden also nur für Daten in Rechnung gestellt, die für Ihre Abfragen relevant sind. Wenn Sie beispielsweise über eine Tabelle verfügen, in der Daten von einer Million Geräten auf sechs Kontinenten gespeichert sind, wenn Ihre Abfrage Prädikate in der Form `WHERE device_id = 'abcdef'` oder hat`WHERE continent = 'North America'`, dann werden Abfragen nur für Daten für das Gerät oder für den Kontinent berechnet.
- Wenn Sie den Kennzahlnamen verwenden, um Daten in derselben Tabelle zu trennen, die nicht gleichzeitig ausgegeben oder nicht häufig abgefragt werden, und wenn Sie Prädikate wie `WHERE measure_name = 'cpu'` in Ihrer Abfrage verwenden, profitieren Sie nicht nur von den Messvorteilen, sondern Timestream for LiveAnalytics kann auch Partitionen effektiv eliminieren, die nicht den in Ihrem Abfrageprädikat verwendeten Kennzahlnamen haben. Auf diese Weise können Sie verwandte Daten mit unterschiedlichen Kennzahlnamen in derselben Tabelle speichern, ohne die Abfragelatenz oder die Kosten zu beeinträchtigen, und es wird vermieden, dass die Daten auf mehrere Tabellen verteilt werden. Der Kennzahlname wird im Wesentlichen verwendet, um die Daten zu partitionieren und Partitionen zu bereinigen, die für die Abfrage irrelevant sind.

Datensätze mit mehreren Kennzahlen im Vergleich zu Datensätzen mit nur einer Kennzahl

Mit Timestream for LiveAnalytics können Sie Daten mit mehreren Messwerten pro Datensatz (Multi-Measure) oder mit einer einzigen Messgröße pro Datensatz (Single-Massage) schreiben.

Datensätze mit mehreren Messungen

In vielen Anwendungsfällen kann ein Gerät oder eine Anwendung, die Sie verfolgen, mehrere Messwerte oder Ereignisse gleichzeitig ausgeben. In solchen Fällen können Sie alle zum gleichen

Zeitstempel ausgegebenen Messwerte in demselben Datensatz mit mehreren Messwerten speichern. Das heißt, alle in demselben Datensatz mit mehreren Kennzahlen gespeicherten Kennzahlen werden als unterschiedliche Spalten in derselben Datenzeile angezeigt.

Stellen Sie sich zum Beispiel vor, dass Ihre Anwendung Messwerte wie CPU, Arbeitsspeicher und disk_iops von einem Gerät ausgibt, das zur gleichen Zeit gemessen wurde. Im Folgenden finden Sie ein Beispiel für eine solche Tabelle, in der mehrere gleichzeitig ausgegebene Messwerte in derselben Zeile gespeichert werden. Sie werden sehen, dass zwei Hosts die Metriken einmal pro Sekunde ausgeben.

Hostname	measure_name	Zeit	cpu	Arbeitsspeicher	disk_iops
Host-24GJU	Kennzahlen	2021-12-01 19:00:00	35	54,9	38,2
Host-24GJU	Kennzahlen	2021-12-01 19:00:01	36	58	39
Gastgeber - 28 GJU	Kennzahlen	2021-12-01 19:00:00	15	55	92
Gastgeber: 28 GJU	Kennzahlen	2021-12-01 19:00:01	16	50	40

Aufzeichnungen über einzelne Messwerte

Die Einzelmessdatensätze eignen sich, wenn Ihre Geräte zu unterschiedlichen Zeitpunkten unterschiedliche Messwerte ausgeben oder wenn Sie eine benutzerdefinierte Verarbeitungslogik verwenden, die metrics/events at different time periods (for instance, when a device's reading/state Änderungen ausgibt). Da jede Kennzahl einen eindeutigen Zeitstempel hat, können die Messwerte in ihren eigenen Datensätzen in Timestream for gespeichert werden. LiveAnalytics Stellen Sie sich zum Beispiel einen IoT-Sensor vor, der Bodentemperatur und Feuchtigkeit erfasst und nur dann eine Aufzeichnung ausgibt, wenn er eine Änderung gegenüber dem zuvor gemeldeten Eintrag feststellt. Das folgende Beispiel bietet ein Beispiel dafür, wie solche Daten mithilfe von Einzelmessdatensätzen ausgegeben werden.

Gerät_ID	measure_name	Zeit	Messwert: :doppelt	Messwert::bigint
sensor-sea478	temperature	2021-12-01 19:22:32	35	NULL
sensor-sea478	temperature	2021-12-01 18:07:51	36	NULL
sensor-sea478	Feuchtigkeit	2021-12-01 19:05:30	NULL	21
sensor-sea478	Feuchtigkeit	2021-12-01 19:00:01	NULL	23

Vergleich von Einzelmesswerten und Mehrmessdatensätzen

Timestream for LiveAnalytics bietet Ihnen die Flexibilität, Ihre Daten je nach den Anforderungen und Merkmalen Ihrer Anwendung als Datensätze mit einer oder mehreren Messungen zu modellieren. In einer einzigen Tabelle können sowohl Datensätze für einzelne Messungen als auch Datensätze mit mehreren Messungen gespeichert werden, wenn Ihre Anwendung dies wünscht. Wenn Ihre Anwendung mehrere Messwerte/Ereignisse gleichzeitig ausgibt, empfiehlt es sich im Allgemeinen, die Daten als Datensätze mit mehreren Messwerten zu modellieren, um einen performanten Datenzugriff und eine kostengünstige Datenspeicherung zu gewährleisten.

Wenn Sie beispielsweise einen DevOps Anwendungsfall zur Erfassung von Metriken und Ereignissen von Hunderttausenden von Servern in Betracht ziehen, sendet jeder Server in regelmäßigen Abständen 20 Messwerte und 5 Ereignisse aus, wobei die Ereignisse und Messwerte gleichzeitig und sofort ausgegeben werden. Diese Daten können entweder mithilfe von Datensätzen mit einzelnen Kennzahlen oder mithilfe von Datensätzen mit mehreren Kennzahlen modelliert werden (das resultierende Schema finden Sie im Open-Source-Datengenerator). In diesem Anwendungsfall führt die Modellierung der Daten mithilfe von Datensätzen mit mehreren Kennzahlen im Vergleich zu Datensätzen mit einzelnen Kennzahlen zu folgenden Ergebnissen:

- Messung der Datenaufnahme — Aufzeichnungen mit mehreren Messungen führen dazu, dass etwa 40% weniger aufgenommene Byte geschrieben werden.

- Batching bei der Datenaufnahme — Datensätze mit mehreren Messungen führen dazu, dass größere Datenpakete gesendet werden, was bedeutet, dass die Clients weniger Threads und weniger Zeit für die Verarbeitung der Datenaufnahme benötigen. CPU
- Speichermessung — Aufzeichnungen mit mehreren Messwerten führen zu etwa achtmal weniger Speicherplatz, was zu erheblichen Speichereinsparungen sowohl für Speicher als auch für Magnetspeicher führt.
- Abfragelatenz — Datensätze mit mehreren Messwerten führen bei den meisten Abfragetypen zu einer geringeren Abfragelatenz im Vergleich zu Datensätzen mit nur einer Messung.
- Gemessene Byte s abfragen — Bei Abfragen, bei denen weniger als 10 MB Daten gescannt werden, sind sowohl Datensätze mit Einzelmesswerten als auch Datensätze mit mehreren Messwerten vergleichbar. Bei Abfragen, die auf eine einzelne Messgröße zugreifen und Daten mit mehr als 10 MB scannen, führen Einzelmessdatensätze in der Regel zu einer geringeren Bytezahl. Bei Abfragen, die sich auf 3 oder mehr Messwerte beziehen, führen Datensätze mit mehreren Messwerten zu einer geringeren Bytezahl.
- Einfaches Ausdrücken von Abfragen mit mehreren Kennzahlen — Wenn Ihre Abfragen auf mehrere Kennzahlen verweisen, führt das Modellieren Ihrer Daten mit Datensätzen mit mehreren Kennzahlen dazu, dass Abfragen einfacher und kompakter geschrieben werden können.

Die oben genannten Faktoren hängen davon ab, wie viele Metriken Sie verfolgen, wie viele Dimensionen Ihre Daten haben usw. Während das vorherige Beispiel einige konkrete Daten für ein Beispiel liefert, sehen wir in vielen Anwendungsszenarien und Anwendungsfällen, in denen das Speichern von Daten als Datensätze mit mehreren Kennzahlen effektiver ist, wenn Ihre Anwendung mehrere Messwerte gleichzeitig ausgibt. Darüber hinaus bieten Ihnen Datensätze mit mehreren Kennzahlen die Flexibilität, Datentypen zu verwenden und mehrere andere Werte als Kontext zu speichern (z. B. das Speichern von Anfragen und zusätzlichen ZeitstempelnIDs, worauf später noch eingegangen wird).

Beachten Sie, dass ein Datensatz mit mehreren Kennzahlen auch dünn besetzte Kennzahlen modellieren kann, wie im vorherigen Beispiel für Einzelmessdatensätze: Sie können den Measure_Name verwenden, um den Namen der Kennzahl zu speichern`DOUBLE`, und einen generischen Multi-Measure-Attributnamen verwenden, wie value_double zum Speichern von `BIGINT` Kennzahlen, value_timestamp zum Speichern zusätzlicher `TIMESTAMP` Werte usw.

Dimensionen und Maße

Eine Tabelle in Timestream für LiveAnalytics ermöglicht Ihnen das Speichern von Dimensionen (zur Identifizierung von Attributen des Geräts/der Daten, die Sie speichern) und Kennzahlen (die Metriken/

Werte, die Sie verfolgen). Weitere Informationen finden Sie unter Timestream für LiveAnalytics Konzepte. Wenn Sie Ihre Anwendung auf Timestream für modellieren, wirkt sich die Art und Weise LiveAnalytics, wie Sie Ihre Daten in Dimensionen und Kennzahlen zuordnen, auf Ihre Datenaufnahme und Abfragelatenz aus. Im Folgenden finden Sie Richtlinien zur Modellierung Ihrer Daten als Dimensionen und Kennzahlen, die Sie auf Ihren Anwendungsfall anwenden können.

Dimensionen auswählen

Daten, die die Quelle identifizieren, von der die Zeitreihendaten gesendet werden, eignen sich hervorragend für Dimensionen, bei denen es sich um Attribute handelt, die sich im Laufe der Zeit nicht ändern. Wenn Sie beispielsweise einen Server haben, der Messwerte ausgibt, dann sind die Attribute, die den Server identifizieren, wie Hostname, Region, Rack, Availability Zone, Kandidaten für Dimensionen. In ähnlicher Weise eignen sich für ein IoT-Gerät mit mehreren Sensoren, die Zeitreihendaten melden, Geräte-ID, Sensor-ID usw. für Abmessungen.

Wenn Sie Daten als Datensätze mit mehreren Kennzahlen schreiben, werden Dimensionen und Attribute mit mehreren Kennzahlen als Spalten in der Tabelle angezeigt, wenn Sie eine SELECT Anweisung in der Tabelle ausführen DESCRIBE oder ausführen. Daher können Sie beim Schreiben Ihrer Abfragen die Dimensionen und Kennzahlen in derselben Abfrage beliebig verwenden. Beachten Sie bei der Erstellung Ihres Schreibdatensatzes für die Datenaufnahme jedoch Folgendes, wenn Sie auswählen, welche Attribute als Dimensionen und welche Kennzahlwerte angegeben werden:

- Die Dimensionsnamen, Werte, der Kennzahlname und der Zeitstempel identifizieren die Zeitreihendaten eindeutig. Timestream for LiveAnalytics verwendet diesen eindeutigen Bezeichner, um Daten automatisch zu deduplizieren. Das heißt, wenn Timestream for zwei Datenpunkte mit denselben Werten für Dimensionsnamen, Dimensionswerte, Kennzahlnamen und Zeitstempel LiveAnalytics empfängt und wenn die Werte dieselbe Versionsnummer haben, dann Timestream für Deduplikate. LiveAnalytics Wenn die neue Schreibanforderung eine niedrigere Version als die bereits in Timestream vorhandenen Daten hat LiveAnalytics, wird die Schreibanforderung zurückgewiesen. Wenn die neue Schreibanforderung eine höhere Version hat, überschreibt der neue Wert den alten Wert. Daher wirkt sich die Art und Weise, wie Sie Ihre Dimensionswerte auswählen, auf dieses Deduplizierungsverhalten aus.
- Dimensionsnamen und -werte können nicht aktualisiert werden, Kennzahlwerte dagegen schon. Daher lassen sich alle Daten, die möglicherweise aktualisiert werden müssen, besser als Messwerte modellieren. Wenn Sie beispielsweise eine Maschine in der Werkshalle haben, deren Farbe sich ändern kann, können Sie die Farbe als Messwert modellieren, es sei denn, Sie möchten die Farbe auch als identifizierendes Attribut verwenden, das für die Deduplizierung benötigt wird.

Das heißt, Messwerte können zum Speichern von Attributen verwendet werden, die sich im Laufe der Zeit nur langsam ändern.

Beachten Sie, dass eine Tabelle in Timestream for die Anzahl der eindeutigen Kombinationen von Dimensionsnamen und Werten LiveAnalytics nicht begrenzt. Sie können beispielsweise Milliarden solcher eindeutigen Wertekombinationen in einer Tabelle speichern. Wie Sie jedoch anhand der folgenden Beispiele sehen werden, kann eine sorgfältige Auswahl von Dimensionen und Kennzahlen Ihre Anforderungslatenz erheblich optimieren, insbesondere bei Abfragen.

Einzigartig IDs in den Dimensionen

Wenn Ihr Anwendungsszenario erfordert, dass Sie für jeden Datenpunkt eine eindeutige Kennung speichern (z. B. eine Anforderungs-ID, eine Transaktions-ID oder eine Korrelations-ID), führt die Modellierung des ID-Attributs als Kennzahlwert zu einer deutlich besseren Abfragelatenz. Wenn Sie Ihre Daten mit Datensätzen mit mehreren Kennzahlen modellieren, wird die ID in derselben Zeile im Kontext mit Ihren anderen Dimensionen und Zeitreihendaten angezeigt, sodass Ihre Abfragen sie weiterhin effektiv verwenden können. Betrachtet man beispielsweise einen DevOps Anwendungsfall, bei dem jeder von einem Server ausgegebene Datenpunkt über ein eindeutiges Anforderungs-ID-Attribut verfügt, führt die Modellierung der Anforderungs-ID als Kennzahlwert zu einer bis zu viermal geringeren Abfragelatenz bei verschiedenen Abfragetypen, im Gegensatz zur Modellierung der eindeutigen Anforderungs-ID als Dimension.

Sie können eine ähnliche Analogie für Attribute verwenden, die nicht für jeden Datenpunkt eindeutig sind, sondern Hunderttausende oder Millionen von Einzelwerten haben. Sie können diese Attribute sowohl als Dimensionen als auch als Messwerte modellieren. Sie sollten es als Dimension modellieren, wenn die Werte für die Deduplizierung auf dem Schreibpfad erforderlich sind, wie bereits erwähnt, oder wenn Sie sie häufig als Prädikat verwenden (z. B. in der `WHERE` Klausel mit einem Gleichheitsprädikat für einen Wert dieses Attributs, z. B. `device_id = 'abcde'` wenn Ihre Anwendung Millionen von Geräten verfolgt) in Ihren Abfragen.

Vielfältige Datentypen mit Datensätzen mit mehreren Kennzahlen

Datensätze mit mehreren Messwerten bieten Ihnen die Flexibilität, Ihre Daten effektiv zu modellieren. Daten, die Sie in einem Datensatz mit mehreren Kennzahlen speichern, werden in der Tabelle ähnlich wie Dimensionen als Spalten angezeigt, sodass Dimensionen und Kennzahlwerte genauso einfach abgefragt werden können. Sie haben einige dieser Muster in den zuvor besprochenen Beispielen gesehen. Im Folgenden finden Sie weitere Muster, mit denen Sie Datensätze mit

mehreren Messwerten effektiv nutzen können, um den Anwendungsfällen Ihrer Anwendung gerecht zu werden.

Datensätze mit mehreren Messwerten unterstützen Attribute der DatentypenDOUBLE, BIGINT VARCHARBOOLEAN, und. TIMESTAMP Daher passen sie naturgemäß zu unterschiedlichen Attributtypen:

- Standortinformationen: Wenn Sie beispielsweise den Standort (ausgedrückt als Breitengrad und Längengrad) verfolgen möchten, führt die Modellierung als Attribut mit mehreren Messgrößen zu einer geringeren Abfragelatenz im Vergleich zum Speichern der Daten als VARCHAR Dimensionen, insbesondere wenn Sie Prädikate für Breitengrad und Längengrad haben.
- Mehrere Zeitstempel in einem Datensatz: Wenn Ihr Anwendungsszenario erfordert, dass Sie mehrere Zeitstempel für einen Zeitreihendatensatz verfolgen, können Sie sie als zusätzliche Attribute im Datensatz mit mehreren Messwerten modellieren. Dieses Muster kann verwendet werden, um Daten mit future oder vergangenen Zeitstempeln zu speichern. Beachten Sie, dass jeder Datensatz weiterhin den Zeitstempel in der Zeitspalte verwendet, um einen Datensatz zu partitionieren, zu indizieren und eindeutig zu identifizieren.

Insbesondere wenn Sie über numerische Daten oder Zeitstempel verfügen, für die Sie Prädikate in der Abfrage verwenden, führt die Modellierung dieser Attribute als Attribute mit mehreren Kennzahlen und nicht als Dimensionen zu einer geringeren Abfragelatenz. Dies liegt daran, dass Sie beim Modellieren solcher Daten mithilfe der umfangreichen Datentypen, die in Datensätzen mit mehreren Kennzahlen unterstützt werden, die Prädikate mithilfe systemeigener Datentypen ausdrücken können, anstatt Werte in einen anderen Datentyp umzuwandeln, wenn Sie solche Daten als Dimensionen modelliert haben. VARCHAR

Verwenden des Kennzahlnamens bei Datensätzen mit mehreren Kennzahlen

Tabellen in Timestream LiveAnalytics unterstützen ein spezielles Attribut (oder eine Spalte), das als Kennzahlname bezeichnet wird. Sie geben für jeden Datensatz, für den Sie in Timestream schreiben, einen Wert für dieses Attribut an. LiveAnalytics Für Datensätze mit einzelnen Messwerten ist es selbstverständlich, den Namen Ihrer Metrik zu verwenden (z. B. CPU, Speicher für Servermetriken oder Temperatur, Druck für Sensormetriken). Wenn Sie Datensätze mit mehreren Messwerten verwenden, kann Druck zu Attributnamen mit mehreren Messwerten werden, da die Attribute in einem Datensatz mit mehreren Messwerten die Namen CPU, Speicher oder Temperatur haben (und diese Namen werden zu Spaltennamen in der Tabelle). Eine natürliche Frage ist also, wie der Messname effektiv verwendet werden kann.

Timestream for LiveAnalytics verwendet die Werte im Kennzahlnamenattribut, um die Daten zu partitionieren und zu indizieren. Wenn also eine Tabelle mehrere verschiedene Kennzahlnamen hat und die Abfragen diese Werte als Abfrageprädikate verwenden, LiveAnalytics kann Timestream for seine benutzerdefinierte Partitionierung und Indizierung verwenden, um Daten herauszuschneiden, die für Abfragen nicht relevant sind. Wenn Ihre Tabelle beispielsweise CPU- und Speicherkennzahlnamen hat und Ihre Abfrage ein Prädikat hat`WHERE measure_name = 'cpu'`, LiveAnalytics kann Timestream for Daten für Kennzahlnamen, die für die Abfrage nicht relevant sind, effektiv bereinigen, z. B. Zeilen mit Kennzahlnamenspeicher in diesem Beispiel. Diese Bereinigung gilt auch dann, wenn Kennzahlnamen mit Datensätzen mit mehreren Kennzahlen verwendet werden. Sie können das Kennzahlnamenattribut effektiv als Partitionierungsattribut für eine Tabelle verwenden. Der Kennzahlname zusammen mit den Namen und Werten der Dimensionen sowie die Zeit werden verwendet, um die Daten in einem Timestream für LiveAnalytics eine Tabelle zu partitionieren. Beachten Sie die Beschränkungen für die Anzahl der eindeutigen Kennzahlnamen, die in einem Timestream für LiveAnalytics eine Tabelle zulässig sind. Beachten Sie auch, dass ein Kennzahlname auch einem Messwert-Datentyp zugeordnet ist. Beispielsweise kann ein einzelner Kennzahlname nur einem Messwerttyp zugeordnet werden. Dieser Typ kann einer der folgenden sein: `DOUBLE`,`BIGINT`,`BOOLEAN`,`VARCHAR`, und`MULTI`. Datensätze mit mehreren Kennzahlen, die mit einem Kennzahlnamen gespeichert werden, haben den Datentyp als`MULTI`. Da in einem einzigen Datensatz mit mehreren Kennzahlen mehrere Metriken mit unterschiedlichen Datentypen (`DOUBLE`,, `BIGINT``VARCHAR`, `BOOLEAN` und`TIMESTAMP`) gespeichert werden können, können Sie Daten verschiedener Typen in einem Datensatz mit mehreren Kennzahlen verknüpfen.

In den folgenden Abschnitten werden einige verschiedene Beispiele dafür beschrieben, wie das Attribut „Kennzahlname“ effektiv verwendet werden kann, um verschiedene Datentypen in derselben Tabelle zu gruppieren.

IoT-Sensoren melden Qualität und Wert

Stellen Sie sich vor, Sie haben eine Anwendung, die Daten von IoT-Sensoren überwacht. Jeder Sensor verfolgt unterschiedliche Messwerte wie Temperatur und Druck. Zusätzlich zu den tatsächlichen Werten melden die Sensoren auch die Qualität der Messungen, die ein Maß dafür ist, wie genau der Messwert ist, und eine Einheit für die Messung. Da Qualität, Einheit und Wert zusammen ausgegeben werden, können sie als Datensätze mit mehreren Messwerten modelliert werden, wie in den folgenden Beispieldaten gezeigt, wobei device_id eine Dimension ist, Qualität, Wert und Einheit Attribute mit mehreren Messwerten sind:

Gerät_ID	measure_name	Zeit	Qualität	Wert	Einheit
sensor-sea478	temperature	2021-12-01 19:22:32	92	35	c
sensor-sea478	temperature	2021-12-01 18:07:51	93	34	c
sensor-sea478	pressure	2021-12-01 19:05:30	98	31	psi
sensor-sea478	pressure	2021-12-01 19:00:01	24	132	psi

Dieser Ansatz ermöglicht es Ihnen, die Vorteile von Datensätzen mit mehreren Kennzahlen mit der Partitionierung und Bereinigung von Daten unter Verwendung der Werte des Kennzahlnamens zu kombinieren. Wenn Abfragen auf eine einzelne Messgröße verweisen, z. B. Temperatur, können Sie ein Prädikat für den Namen einer Kennzahl in die Abfrage aufnehmen. Im Folgenden finden Sie ein Beispiel für eine solche Abfrage, bei der die Einheit auch für Messungen mit einer Qualität über 90 projiziert wird.

```
SELECT device_id, time, value AS temperature, unit
FROM db.table
WHERE time > ago(1h)
    AND measure_name = 'temperature'
    AND quality > 90
```

Durch die Verwendung des Prädikats measure_name für die Abfrage kann Timestream for Partitionen und Daten LiveAnalytics , die für die Abfrage nicht relevant sind, effektiv bereinigen und so die Latenz Ihrer Abfrage verbessern.

Es ist auch möglich, alle Metriken in demselben Datensatz mit mehreren Kennzahlen zu speichern, wenn alle Metriken mit demselben Zeitstempel ausgegeben werden und/oder mehrere Metriken zusammen in derselben Abfrage abgefragt werden. Sie können beispielsweise einen Datensatz mit mehreren Messungen mit den Attributen temperature_quality, temperature_value, temperature_unit, pressure_quality, pressure_value, pressure_unit usw. erstellen. Viele der zuvor erörterten Punkte zur Modellierung von Daten mit Einzelmesswerten und Datensätzen mit mehreren Messwerten

beziehen sich auf Ihre Entscheidung, wie die Daten modelliert werden sollen. Berücksichtigen Sie Ihre Zugriffsmuster für Abfragen und die Art und Weise, wie Ihre Daten generiert werden, um ein Modell auszuwählen, das Ihre Kosten, die Aufnahme und die Abfragelatenz optimiert und das Schreiben Ihrer Abfragen vereinfacht.

Verschiedene Arten von Metriken in derselben Tabelle

Ein weiterer Anwendungsfall, bei dem Sie Datensätze mit mehreren Kennzahlen mit Kennzahlnamen kombinieren können, besteht darin, verschiedene Datentypen zu modellieren, die unabhängig voneinander von demselben Gerät ausgegeben werden. Stellen Sie sich den Anwendungsfall DevOps Überwachung vor, bei dem Server zwei Arten von Daten ausgeben: regelmäßig ausgestrahlte Messwerte und unregelmäßige Ereignisse. Ein Beispiel für diesen Ansatz ist das Schema, das im Datengenerator beschrieben wird, der einen DevOps Anwendungsfall modelliert. In diesem Fall können Sie die verschiedenen Datentypen, die von demselben Server ausgegeben werden, in derselben Tabelle speichern, indem Sie unterschiedliche Kennzahlnamen verwenden. Beispielsweise werden alle Metriken, die zur gleichen Zeit ausgegeben werden, zusammen mit den Kennzahlnamen „Metriken“ gespeichert. Alle Ereignisse, die zu einem anderen Zeitpunkt als die Metriken ausgegeben werden, werden mit dem Namen der Kennzahl „Ereignisse“ gespeichert. Das Kennzahlschema für die Tabelle (z. B. Ausgabe einer `SHOW MEASURES` Abfrage) lautet:

measure_name	data_type	Dimensionen
Veranstaltungen	mehrfach	[{"data_type“ :"varchar“, "dimension_name“ :"availability_zone "}, {" data_type“ :"varchar“, "dimension_name“ :"microservice_name "}, {" data_type“ :"varchar“, "dimension_name“ :"instance_name "}, {" data_type“ :"varchar“, "dimension_name“ :"Prozessname "}, {" data_type“ :"varchar“, "dimension_name“ :"jdk_version "}, {" data_type“ :"varchar“, "dimension_name“ :"cell "}, {" data_type“ :"varchar“, "dimension_name“ :"region

measure_name	data_type	Dimensionen
		"}, {" data_type“ :"dimensi on_type“ :"dimensi on_type“ :"region "} „varchar“, "dimension_name“ :"silo "}]
Kennzahlen	mehrfach	[{"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 “ :"varchar“, "dimensio n_name“ :"os_version "}, {" data_type“ :"varchar“, "dimension_name“ :"cell "}, {" data_type“ :"varchar“, "dimension_name“ :"region "}, {" data_type“ :"varchar “, "dimension_name“ :"silo "}, {" data_type“ :"varchar“, "dimension_name“ :"silo "}, {" data_type“ :"varchar“ archar“, "dimension_name“ :"Instanztyp "}]

In diesem Fall können Sie sehen, dass die Ereignisse und Metriken auch unterschiedliche Dimensionssätze haben, wobei Ereignisse unterschiedliche Dimensionen haben jdk_version und process_name, während Metriken die Dimensionen instance_type und os_version haben.

Die Verwendung verschiedener Kennzahlnamen ermöglicht es Ihnen, Abfragen mit Prädikaten zu schreiben, z. B. um nur die Metriken abzurufen. `WHERE measure_name = 'metrics'` Wenn alle von derselben Instanz ausgegebenen Daten in derselben Tabelle enthalten sind, bedeutet das, dass Sie auch eine einfachere Abfrage mit dem Prädikat instance_name schreiben können, um alle Daten

für diese Instanz abzurufen. Beispielsweise gibt ein Prädikat der Form `WHERE instance_name = 'instance-1234'` ohne das Prädikat measure_name alle Daten für eine bestimmte Serverinstanz zurück.

Empfehlungen für die Partitionierung von Datensätzen mit mehreren Kennzahlen

> **Important**
>
> Dieser Abschnitt ist veraltet!
> Diese Empfehlungen sind veraltet. Die Partitionierung kann jetzt besser mit benutzerdefinierten Partitionsschlüsseln gesteuert werden.

Wir haben festgestellt, dass es im Zeitreihen-Ökosystem immer mehr Workloads gibt, die riesige Datenmengen aufnehmen und speichern müssen und gleichzeitig Abfrageantworten mit geringer Latenz benötigen, wenn auf Daten mit einem Satz von Dimensionswerten mit hoher Kardinalität zugegriffen wird.

Aufgrund dieser Merkmale sind die Empfehlungen in diesem Abschnitt für Kunden-Workloads nützlich, die Folgendes aufweisen.

- Datensätze mit mehreren Messwerten übernommen oder möchten diese übernehmen.
- Rechnen Sie damit, dass eine große Menge an Daten in das System eingeht, die über lange Zeiträume gespeichert werden.
- Erfordern Reaktionszeiten mit niedriger Latenz für ihre Hauptzugriffsmuster (Abfrage).
- Beachten Sie, dass die wichtigsten Abfragemuster eine Art von Filterbedingung im Prädikat beinhalten. Diese Filterbedingung basiert auf einer Dimension mit hoher Kardinalität. Betrachten Sie beispielsweise Ereignisse oder Aggregationen nach UserId, DeviceId, ServerID, Hostname usw.

In diesen Fällen hilft ein einziger Name für alle Mehrkennzahlen nicht weiter, da unsere Engine Namen mit mehreren Kennzahlen verwendet, um die Daten zu partitionieren, und ein einziger Wert den Partitionsvorteil, den Sie erhalten, einschränkt. Die Partitionierung dieser Datensätze basiert hauptsächlich auf zwei Dimensionen. Nehmen wir an, die Zeit befindet sich auf der X-Achse und ein Hash von Dimensionsnamen und der `measure_name` auf der Y-Achse. Das funktioniert `measure_name` in diesen Fällen fast wie ein Partitionierungsschlüssel.

Unsere Empfehlung lautet wie folgt.

- Wenn Sie Ihre Daten für Anwendungsfälle wie den genannten modellieren, verwenden Sie a, `measure_name` das eine direkte Ableitung Ihres Hauptzugriffsmusters für Abfragen ist. Beispielsweise:
  - In Ihrem Anwendungsfall müssen Sie die Anwendungsleistung und QoE aus der Sicht des Endbenutzers verfolgen. Dies könnte auch die Nachverfolgung von Messungen für einen einzelnen Server oder ein IoT-Gerät sein.
  - Wenn Sie abfragen und nach filtern UserId, müssen Sie zum Zeitpunkt der Aufnahme herausfinden, wie Sie am besten eine Verbindung herstellen können. `measure_name` UserId
  - Da eine Tabelle mit mehreren Kennzahlen nur 8192 verschiedene Kennzahlnamen enthalten kann, sollten unabhängig von der verwendeten Formel nicht mehr als 8192 verschiedene Werte generiert werden.
- Ein Ansatz, den wir erfolgreich für Zeichenkettenwerte angewendet haben, besteht darin, einen Hashing-Algorithmus auf den Zeichenkettenwert anzuwenden. Führen Sie dann die Modulo-Operation mit dem absoluten Wert des Hash-Ergebnisses und 8192 durch.

```
measure_name = getMeasureName(UserId)
int getMeasureName(value) {
    hash_value =  abs(hash(value))
    return hash_value % 8192
}
```

- Wir haben auch hinzugefügt`abs()`, das Zeichen zu entfernen, sodass die Möglichkeit entfällt, dass Werte zwischen -8192 und 8192 liegen. Dies sollte vor der Modulo-Operation durchgeführt werden.
- Mit dieser Methode können Ihre Abfragen in einem Bruchteil der Zeit ausgeführt werden, die für die Ausführung in einem unpartitionierten Datenmodell erforderlich wäre.
- Achten Sie beim Abfragen der Daten darauf, dass Sie eine Filterbedingung in das Prädikat aufnehmen, die den neu abgeleiteten Wert von measure_name verwendet. Beispielsweise:
  -
```
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))
```
  - Dadurch wird die Gesamtzahl der gescannten Partitionen minimiert, sodass Sie Daten erhalten, die im Laufe der Zeit zu schnelleren Abfragen führen.

Denken Sie daran, dass, wenn Sie die Vorteile dieses Partitionsschemas nutzen möchten, der Hash auf der Clientseite berechnet und an Timestream LiveAnalytics als statischer Wert an die Abfrage-Engine übergeben werden muss. Das vorherige Beispiel bietet eine Möglichkeit zu überprüfen, ob der generierte Hash bei Bedarf von der Engine aufgelöst werden kann.

time	host_name	location	Servertyp	cpu_usage	verfügbarer_Speicher	CPU_Temp
07.09.2022 21:48:44.000000000	Host-1235	us-ost-1	5,8 xl	55	16,2	78
R2022-09-07 21:48:44.000000000	Host-3587	us-west1	5,8 xl	62	18,1	81
07.09.2022 21:48:45.000 000000	Host-258743	EU-Zentral	5,8 XL	88	9,4	91
07.09.2022 21:48:45,000000000	Host-35654	us-ost2	5,8 xl	29	24	54
R2022-09-07 21:48:45,000000000	Host-254	us-west1	5,8 xl	44	32	48

Zur Generierung der zugehörigen `measure_name` Informationen gibt es zwei Wege, die von Ihrem Aufnahmemuster abhängen.

1. Für die Batch-Aufnahme historischer Daten — Sie können die Transformation zu Ihrem Schreibcode hinzufügen, wenn Sie Ihren eigenen Code für den Batch-Prozess verwenden.

Aufbauend auf dem vorherigen Beispiel.

```
        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
```

Resultierender Datensatz

time	host_name	location	measure_name	Servertyp	cpu_usage	verfügbarer_Speicher	CPU_Temp
07.09.2022 21:48:44.000000000	Host-1235	us-ost-1	6445	5,8 xl	55	16,2	78

time	host_name	location	measure_name	Servertyp	cpu_usage	verfügbarer_Speicher	CPU_Temp
R2022-09-07 21:48:44.000000000	Host-3587	us-west1	6399	5,8 xl	62	18,1	81
07.09.2022 21:48:45.000 000000	Host-258743	EU-Zentral	640	5,8 XL	88	9,4	91
07.09.2022 21:48:45,000000000	Host-35654	us-ost2	2093	5,8 xl	29	24	54
R2022-09-07 21:48:45,000000000	Host-254	us-west1	7051	5,8 xl	44	32	48

2. Für die Erfassung in Echtzeit — Sie müssen die Daten während der Übertragung generieren, sobald die Daten `measure_name` eintreffen.

In beiden Fällen empfehlen wir Ihnen, Ihren Algorithmus zur Hash-Generierung an beiden Enden (Aufnahme und Abfrage) zu testen, um sicherzustellen, dass Sie dieselben Ergebnisse erhalten.

Im Folgenden finden Sie einige Codebeispiele, anhand derer Sie den Hashwert generieren können. `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 Abhängigkeit in Maven

```
        <dependency>
            <groupId>net.jpountz.lz4</groupId>
            <artifactId>lz4</artifactId>
            <version>1.3.0</version>
        </dependency>
```

Sicherheit

- Stellen Sie für den kontinuierlichen Zugriff auf Timestream for sicher LiveAnalytics, dass die Verschlüsselungsschlüssel gesichert sind und nicht gesperrt oder unzugänglich gemacht werden.
- Überwachen Sie die API Zugriffsprotokolle von. AWS CloudTrail Prüfen Sie alle anomalen Zugriffsmuster unberechtigter Benutzer und sperren Sie sie.
- Beachten Sie die zusätzlichen Richtlinien, die unter beschrieben sind. Bewährte Sicherheitsmethoden für Amazon Timestream für LiveAnalytics

Konfiguration von Amazon Timestream für LiveAnalytics

Konfigurieren Sie den Datenaufbewahrungszeitraum für den Speicher und den Magnetspeicher entsprechend den Anforderungen an Datenverarbeitung, Speicherung, Abfrageleistung und Kosten.

- Stellen Sie die Datenspeicherung des Speicherspeichers so ein, dass sie den Anforderungen Ihrer Anwendung für die Verarbeitung spät eingehender Daten entspricht. Bei spät eintreffenden Daten handelt es sich um eingehende Daten, deren Zeitstempel vor der aktuellen Uhrzeit liegt. Es wird von Ressourcen ausgegeben, die Ereignisse für einen bestimmten Zeitraum stapeln, bevor sie an Timestream gesendet werden LiveAnalytics, und auch von Ressourcen mit intermittierender Konnektivität, z. B. einem IoT-Sensor, der zeitweise online ist.
- Wenn Sie davon ausgehen, dass verspätet eingehende Daten gelegentlich mit Zeitstempeln ankommen, die vor der Aufbewahrung des Speichers liegen, sollten Sie Magnetspeicher-Schreibvorgänge für Ihre Tabelle aktivieren. Sobald Sie die EnableMagneticStoreWrites

Einstellungen MagneticStoreWritesProperties für eine Tabelle eingegeben haben, akzeptiert die Tabelle Daten, deren Zeitstempel vor der Aufbewahrung im Speicher liegt, aber innerhalb der Aufbewahrungsfrist für den Magnetspeicher liegt.

- Berücksichtigen Sie die Merkmale der Abfragen, die Sie auf Timestream ausführen möchten, LiveAnalytics z. B. die Art der Abfragen, die Häufigkeit, den Zeitraum und die Leistungsanforderungen. Dies liegt daran, dass der Speicherspeicher und der Magnetspeicher für unterschiedliche Szenarien optimiert sind. Der Speicherspeicher ist für schnelle point-in-time Abfragen optimiert, bei denen kleine Mengen aktueller Daten verarbeitet werden, die an Timestream für LiveAnalytics gesendet wurden. Der Magnetspeicher ist für schnelle analytische Abfragen optimiert, bei denen mittlere bis große Datenmengen verarbeitet werden, die an Timestream for gesendet werden. LiveAnalytics
- Ihre Aufbewahrungsfrist für Daten sollte auch von den Kostenanforderungen Ihres Systems beeinflusst werden.

  Stellen Sie sich beispielsweise ein Szenario vor, in dem der Schwellenwert für verspätete Daten für Ihre Anwendung 2 Stunden beträgt und Ihre Anwendungen viele Anfragen senden, die Daten im Wert von einem Tag, einer Woche oder einem Monat verarbeiten. In diesem Fall empfiehlt es sich, eine kürzere Aufbewahrungszeit für den Speicherspeicher (2-3 Stunden) zu konfigurieren und mehr Daten in den Magnetspeicher fließen zu lassen, sofern der Magnetspeicher für schnelle analytische Abfragen optimiert ist.

Machen Sie sich mit den Auswirkungen einer Erhöhung oder Verkürzung der Datenaufbewahrungsdauer des Speicherspeichers und des Magnetspeichers einer vorhandenen Tabelle vertraut.

- Wenn Sie die Aufbewahrungszeit des Speicherspeichers verkürzen, werden die Daten vom Speicherspeicher in den Magnetspeicher verschoben, und diese Datenübertragung ist dauerhaft. Timestream for ruft LiveAnalytics keine Daten aus dem Magnetspeicher ab, um den Speicherspeicher zu füllen. Wenn Sie die Aufbewahrungsdauer des Magnetspeichers verringern, werden die Daten aus dem System gelöscht, und die Daten werden dauerhaft gelöscht.
- Wenn Sie die Aufbewahrungsdauer des Speicherspeichers oder des Magnetspeichers verlängern, wird die Änderung ab diesem Zeitpunkt für Daten wirksam, die an Timestream gesendet werden. LiveAnalytics Timestream for ruft LiveAnalytics keine Daten aus dem Magnetspeicher ab, um den Speicherspeicher zu füllen. Wenn beispielsweise die Aufbewahrungsdauer des Speicherspeichers ursprünglich auf 2 Stunden festgelegt und dann auf 24 Stunden erhöht wurde, dauert es 22 Stunden, bis der Speicherspeicher Daten im Wert von 24 Stunden enthält.

Schreibt

- Stellen Sie sicher, dass der Zeitstempel der eingehenden Daten nicht vor der für den Speicherspeicher konfigurierten Datenspeicherung liegt und nicht nach dem in definierten future Aufnahmezeitraum. Kontingente Das Senden von Daten mit einem Zeitstempel außerhalb dieser Grenzen führt dazu, dass die Daten von Timestream zurückgewiesen werden, LiveAnalytics es sei denn, Sie aktivieren Magnetspeicher-Schreibvorgänge für Ihre Tabelle. Wenn Sie Magnetspeicher-Schreibvorgänge aktivieren, stellen Sie sicher, dass der Zeitstempel für eingehende Daten nicht vor der für den Magnetspeicher konfigurierten Datenspeicherung liegt.
- Wenn Sie erwarten, dass Daten zu spät ankommen, aktivieren Sie Magnetic Store Writes für Ihre Tabelle. Auf diese Weise können Daten mit Zeitstempeln aufgenommen werden, die außerhalb der Aufbewahrungsfrist Ihres Speicherspeichers, aber dennoch innerhalb der Aufbewahrungsfrist Ihres Magnetspeichers liegen. Sie können dies festlegen, indem Sie das `EnableMagneticStoreWrites` Kennzeichen in Ihrer Tabelle aktualisieren. `MagneticStoreWritesProperties` Diese Eigenschaft ist standardmäßig falsch. Beachten Sie, dass Schreibvorgänge in den Magnetspeicher nicht sofort für Abfragen verfügbar sind. Sie werden innerhalb von 6 Stunden verfügbar sein.
- Richten Sie Workloads mit hohem Durchsatz auf den Speicherspeicher aus, indem Sie sicherstellen, dass die Zeitstempel der aufgenommenen Daten innerhalb der Aufbewahrungsgrenzen des Speicherspeichers liegen. Schreibvorgänge in den Magnetspeicher sind auf eine maximale Anzahl von aktiven Magnetspeicherpartitionen beschränkt, die gleichzeitig von einer Datenbank aufgenommen werden können. Sie finden diese `ActiveMagneticStorePartitions` Metrik in. CloudWatch Um die Anzahl der aktiven Magnetspeicherpartitionen zu reduzieren, sollten Sie versuchen, die Anzahl der Serien und die Dauer der gleichzeitigen Aufnahme von Magnetspeichern zu reduzieren.
- Beim Senden von Daten an Timestream for sollten Sie mehrere Datensätze in einer einzigen Anfrage stapeln LiveAnalytics, um die Leistung der Datenaufnahme zu optimieren.
  - Es ist von Vorteil, Datensätze aus derselben Zeitreihe und Datensätze mit demselben Kennzahlnamen zu stapeln.
  - Kombinieren Sie so viele Datensätze wie möglich in einer einzigen Anfrage, solange die Anfragen innerhalb der in definierten Servicegrenzen liegenKontingente.
  - Verwenden Sie nach Möglichkeit gemeinsame Attribute, um die Kosten für Datenübertragung und Datenaufnahme zu reduzieren. Weitere Informationen finden Sie unter. WriteRecords API

- Wenn Sie beim Schreiben von Daten in Timestream for auf partielle clientseitige Fehler stoßen, können Sie den Stapel von Datensätzen LiveAnalytics, die nicht aufgenommen werden konnten, erneut senden, nachdem Sie die Ursache für die Ablehnung behoben haben.
- Nach Zeitstempeln geordnete Daten weisen eine bessere Schreibleistung auf.
- Amazon Timestream for LiveAnalytics ist so konzipiert, dass es automatisch an die Anforderungen Ihrer Anwendung angepasst wird. Wenn Timestream for notes LiveAnalytics bei Schreibanforderungen aus Ihrer Anwendung Spitzenwerte erreicht, kann es in Ihrer Anwendung zu einer gewissen anfänglichen Drosselung des Speicherspeichers kommen. Wenn in Ihrer Anwendung eine Drosselung des Speicherspeichers auftritt, senden Sie weiterhin Daten mit derselben (oder höheren) Geschwindigkeit LiveAnalytics an Timestream, damit Timestream for automatisch skaliert werden kann, um den Anforderungen Ihrer Anwendung LiveAnalytics gerecht zu werden. Wenn Sie eine Drosselung des Magnetspeichers feststellen, sollten Sie die Geschwindigkeit der Aufnahme von Magnetspeichern verringern, bis die Anzahl der Magnetspeicher sinkt. `ActiveMagneticStorePartitions`

Batch-Laden

Bewährte Methoden für das Laden von Batches werden unter beschriebenBewährte Methoden zum Laden von Batch.

Abfragen

Im Folgenden finden Sie empfohlene bewährte Methoden für Abfragen mit Amazon Timestream for LiveAnalytics.

- Geben Sie nur die Kennzahlen- und Dimensionsnamen an, die für die Abfrage unbedingt erforderlich sind. Das Hinzufügen von überflüssigen Spalten erhöht die Anzahl der Datenscans, was sich auf die Leistung von Abfragen auswirkt.
- Bevor Sie Ihre Abfrage in der Produktion einsetzen, empfehlen wir Ihnen, die Abfrageergebnisse zu überprüfen, um sicherzustellen, dass die räumliche und zeitliche Bereinigung optimal ist. Weitere Informationen finden Sie unter Verwenden von Abfrageerkenntnissen zur Optimierung von Abfragen in Amazon Timestream.
- Wenn möglich, übertragen Sie die Datenberechnung auf Timestream, um die integrierten Aggregate und Skalarfunktionen in der Klausel und gegebenenfalls in der SELECT Klausel zu LiveAnalytics verwenden, um die WHERE Abfrageleistung zu verbessern und die Kosten zu senken. Siehe SELECT und Aggregationsfunktionen.

- Verwenden Sie nach Möglichkeit Näherungsfunktionen. Verwenden Sie z. B. APPROX _ DISTINCT anstelle von COUNT (DISTINCTcolumn_name), um die Abfrageleistung zu optimieren und die Abfragekosten zu senken. Siehe Aggregationsfunktionen.
- Verwenden Sie einen CASE Ausdruck, um komplexe Aggregationen durchzuführen, anstatt mehrmals aus derselben Tabelle auszuwählen. Siehe Die CASE Aussage.
- Fügen Sie nach Möglichkeit einen Zeitraum in die WHERE Klausel Ihrer Abfrage ein. Dadurch werden die Leistung und die Kosten der Abfrage optimiert. Wenn Sie beispielsweise nur die Daten der letzten Stunde in Ihrem Datensatz benötigen, fügen Sie ein Zeitprädikat wie time > ago (1h) hinzu. Siehe SELECT und Intervall und Dauer.
- Wenn eine Abfrage auf eine Teilmenge von Kennzahlen in einer Tabelle zugreift, nehmen Sie die Kennzahlnamen immer in die WHERE Klausel der Abfrage auf.
- Verwenden Sie nach Möglichkeit den Gleichheitsoperator, wenn Sie Dimensionen und Kennzahlen in der WHERE Klausel einer Abfrage vergleichen. Ein Gleichheitsprädikat für Dimensionen und Kennzahlnamen ermöglicht eine verbesserte Abfrageleistung und geringere Abfragekosten.
- Vermeiden Sie nach Möglichkeit die Verwendung von Funktionen in der WHERE Klausel, um die Kosten zu optimieren.
- Verwenden Sie die LIKE Klausel nicht mehrfach. Verwenden Sie stattdessen reguläre Ausdrücke, wenn Sie in einer Zeichenfolgenspalte nach mehreren Werten filtern. Siehe Funktionen für reguläre Ausdrücke.
- Verwenden Sie nur die erforderlichen Spalten in der GROUP BY-Klausel einer Abfrage.
- Wenn das Abfrageergebnis in einer bestimmten Reihenfolge vorliegen muss, geben Sie diese Reihenfolge explizit in der ORDER BY-Klausel der äußersten Abfrage an. Wenn Ihr Abfrageergebnis keine Reihenfolge erfordert, vermeiden Sie die Verwendung einer ORDER BY-Klausel, um die Abfrageleistung zu verbessern.
- Verwenden Sie eine LIMIT Klausel, wenn Sie nur die ersten N Zeilen in Ihrer Abfrage benötigen.
- Wenn Sie eine ORDER BY-Klausel verwenden, um sich die oberen oder unteren N Werte anzusehen, verwenden Sie eine LIMIT Klausel, um die Abfragekosten zu reduzieren.
- Verwenden Sie das Paginierungstoken aus der zurückgegebenen Antwort, um die Abfrageergebnisse abzurufen. Weitere Informationen finden Sie unter Abfrage.
- Wenn Sie mit der Ausführung einer Abfrage begonnen haben und feststellen, dass die Abfrage nicht die gewünschten Ergebnisse zurückgibt, brechen Sie die Abfrage ab, um Kosten zu sparen. Weitere Informationen finden Sie unter CancelQuery.
- Wenn Ihre Anwendung gedrosselt wird, senden Sie weiterhin Daten mit derselben Geschwindigkeit LiveAnalytics an Amazon Timestream, damit Amazon Timestream automatisch skalieren kann,

LiveAnalytics um den Anforderungen Ihrer Anwendung an den Abfragedurchsatz gerecht zu werden.

- Wenn die Anforderungen Ihrer Anwendungen an die Parallelität von Abfragen die Standardgrenzwerte von Timestream für überschreiten, erhöhen Sie das Limit bei Kontakt. LiveAnalytics AWS Support

Geplante Abfragen

Geplante Abfragen helfen Ihnen dabei, Ihre Dashboards zu optimieren, indem sie einige aggregierte Statistiken für die gesamte Flotte vorab berechnen. Es liegt also auf der Hand, sich die Frage zu stellen, wie Sie anhand Ihres Anwendungsfalls ermitteln, welche Ergebnisse vorab berechnet werden sollen und wie Sie diese in einer abgeleiteten Tabelle gespeicherten Ergebnisse zur Erstellung Ihres Dashboards verwenden können. Der erste Schritt in diesem Prozess besteht darin, zu ermitteln, welche Panels vorab berechnet werden sollen. Im Folgenden finden Sie einige allgemeine Richtlinien:

- Berücksichtigen Sie die Anzahl der Byte, die von den Abfragen gescannt wurden, die zum Füllen der Panels verwendet werden, die Häufigkeit, mit der das Dashboard neu geladen wird, und die Anzahl der gleichzeitigen Benutzer, die diese Dashboards laden würden. Sie sollten mit den Dashboards beginnen, die am häufigsten geladen werden und große Datenmengen scannen. Die ersten beiden Dashboards im Beispiel für ein aggregiertes Dashboard sowie das Aggregat-Dashboard im Drilldown-Beispiel sind gute Beispiele für solche Dashboards.
- Überlegen Sie, welche Berechnungen wiederholt verwendet werden. Es ist zwar möglich, für jedes Panel und jeden Variablenwert, der im Panel verwendet wird, eine geplante Abfrage zu erstellen, aber Sie können Ihre Kosten und die Anzahl der geplanten Abfragen erheblich optimieren, indem Sie nach Möglichkeiten suchen, mit einer Berechnung die für mehrere Panels erforderlichen Daten vorab zu berechnen.
- Berücksichtigen Sie die Häufigkeit Ihrer geplanten Abfragen, um die materialisierten Ergebnisse in der abgeleiteten Tabelle zu aktualisieren. Sie sollten analysieren, wie oft ein Dashboard aktualisiert wird, im Vergleich zu dem Zeitfenster, das in einem Dashboard abgefragt wird, und mit der Zeiteinteilung, die bei der Vorberechnung verwendet wurde, sowie mit den Bedienfeldern in den Dashboards verglichen werden. Wenn beispielsweise ein Dashboard, das stündliche Aggregate für die letzten Tage darstellt, nur einmal in ein paar Stunden aktualisiert wird, sollten Sie Ihre geplanten Abfragen so konfigurieren, dass sie nur einmal alle 30 Minuten oder eine Stunde aktualisiert werden. Wenn Sie dagegen über ein Dashboard verfügen, das Aggregationen pro Minute grafisch darstellt und etwa jede Minute aktualisiert wird, möchten Sie, dass Ihre geplanten Abfragen die Ergebnisse jede Minute oder einige Minuten aktualisieren.

- Überlegen Sie, welche Abfragemuster mithilfe von geplanten Abfragen weiter optimiert werden können (sowohl im Hinblick auf die Abfragekosten als auch auf die Abfragelatenz). Zum Beispiel bei der Berechnung der eindeutigen Dimensionswerte, die häufig als Variablen in Dashboards verwendet werden, oder bei der Rückgabe des letzten von einem Sensor ausgegebenen Datenpunkts oder des ersten Datenpunkts, der von einem Sensor nach einem bestimmten Datum ausgegeben wurde, usw. Einige dieser Beispielmuster werden in diesem Leitfaden behandelt.

Die obigen Überlegungen werden sich erheblich auf Ihre Einsparungen auswirken, wenn Sie Ihr Dashboard auf die Abfrage der abgeleiteten Tabellen umstellen, auf die Aktualität der Daten in Ihren Dashboards und auf die Kosten, die durch die geplanten Abfragen entstehen.

Client-Anwendungen und unterstützte Integrationen

Führen Sie Ihre Client-Anwendung in derselben Region wie Timestream aus, LiveAnalytics um Netzwerklatenzen und Datenübertragungskosten zu reduzieren. Weitere Hinweise zur Zusammenarbeit mit anderen Diensten finden Sie unter. Arbeiten mit anderen -Services Im Folgenden finden Sie einige weitere hilfreiche Links.

- Bewährte Methoden für die AWS Entwicklung mit dem AWS SDK for Java
- Bewährte Methoden für die Arbeit mit AWS Lambda Funktionen
- Bewährte Methoden für Amazon Managed Service für Apache Flink
- Bewährte Methoden für die Erstellung von Dashboards in Grafana

Allgemeines

- Stellen Sie sicher, dass Sie das AWS Well-Architected Framework befolgen, wenn Sie Timestream for verwenden. LiveAnalytics Dieses Whitepaper bietet Anleitungen zu bewährten Verfahren in den Bereichen betriebliche Exzellenz, Sicherheit, Zuverlässigkeit, Leistungseffizienz und Kostenoptimierung.

Messung und Kostenoptimierung

Mit Amazon Timestream for zahlen Sie nur für das LiveAnalytics, was Sie tatsächlich nutzen. Timestream für LiveAnalytics Zähler getrennt für Schreibvorgänge, gespeicherte Daten und durch Abfragen gescannte Daten. Der Preis für jede Messgröße ist auf der Preisseite angegeben. Sie

können Ihre monatliche Rechnung mit dem Amazon Timestream for LiveAnalytics Pricing Calculator schätzen.

In diesem Abschnitt wird beschrieben, wie die Messung von Schreibvorgängen, Speichern und Abfragen in Timestream for funktioniert. LiveAnalytics Beispielszenarien und Berechnungen werden ebenfalls bereitgestellt. Darüber hinaus ist eine Liste mit bewährten Methoden zur Kostenoptimierung enthalten. Sie können unten ein Thema auswählen:

Themen

Schreibt

Die Schreibgröße jedes Zeitreihenereignisses wird als Summe der Größe des Zeitstempels und eines oder mehrerer Dimensionsnamen, Dimensionswerte, Kennzahlnamen und Kennzahlwerte berechnet. Die Größe des Zeitstempels beträgt 8 Byte. Die Größe von Dimensionsnamen, Dimensionswerten und Kennzahlnamen entspricht der Länge der UTF -8 codierten Byte der Zeichenfolge, die jeden Dimensionsnamen, Dimensionswert und Kennzahlnamen darstellt. Die Größe des Kennzahlwerts hängt vom Datentyp ab. Sie beträgt 1 Byte für den booleschen Datentyp, 8 Byte für Bigint und Double und die Länge der UTF -8 codierten Byte für Zeichenketten. Jeder Schreibvorgang wird in Einheiten von 1 KiB gezählt.

Im Folgenden finden Sie zwei Beispielberechnungen:

Themen

Berechnung der Schreibgröße eines Zeitreihenereignisses

Stellen Sie sich ein Zeitreihenereignis vor, das die CPU Auslastung einer EC2 Instanz darstellt, wie unten dargestellt:

Zeit	Region	az	vpc	Hostname	measure_name	Messwert::doppelt
1602983435238563000	us-east-1	1 d	vpc-1a2b3c4d	Host-24GJU	CPU-Auslastung	35,0

Die Schreibgröße des Zeitreihenereignisses kann wie folgt berechnet werden:

- Zeit = 8 Byte
- erste Dimension = 15 Byte (`region+us-east-1`)
- zweite Dimension = 4 Byte (`az+1d`)
- dritte Dimension = 15 Byte (`vpc+vpc-1a2b3c4d`)
- vierte Dimension = 18 Byte (`hostname+host-24Gju`)
- Name der Maßnahme = 15 Byte (`cpu_utilization`)
- Wert der Kennzahl = 8 Byte

Schreibgröße des Zeitreihenereignisses = 83 Byte

Berechnung der Anzahl der Schreibvorgänge

Stellen Sie sich nun 100 EC2 Instanzen vor, ähnlich der unter beschriebenen InstanzBerechnung der Schreibgröße eines Zeitreihenereignisses, die alle 5 Sekunden Metriken aussenden. Die Gesamtzahl der monatlichen Schreibvorgänge für die EC2 Instances hängt davon ab, wie viele Zeitreihenereignisse pro Schreibvorgang existieren und ob beim Batching von Zeitreihenereignissen gemeinsame Attribute verwendet werden. Ein Beispiel für die Berechnung der gesamten monatlichen Schreibvorgänge wird für jedes der folgenden Szenarien bereitgestellt:

Themen

- Ein Zeitreihenereignis pro Schreibvorgang
- Stapeln von Zeitreihenereignissen in einem Schreibvorgang
- Batchverarbeitung von Zeitreihenereignissen und Verwendung gemeinsamer Attribute bei einem Schreibvorgang

Ein Zeitreihenereignis pro Schreibvorgang

Wenn jeder Schreibvorgang nur ein Zeitreihenereignis enthält, werden die gesamten monatlichen Schreibvorgänge wie folgt berechnet:

- 100 Zeitreihenereignisse = 100 Schreibvorgänge alle 5 Sekunden
- x 12 Schreibvorgänge/Minute = 1.200 Schreibvorgänge
- x 60 Minuten/Stunde = 72.000 Schreibvorgänge
- x 24 Stunden/Tag = 1.728.000 Schreibvorgänge
- x 30 Tage/Monat = 51.840.000 Schreibvorgänge

Gesamtzahl der monatlichen Schreibvorgänge = 51.840.000

Stapeln von Zeitreihenereignissen in einem Schreibvorgang

Da jeder Schreibvorgang in Einheiten von 1 KB gemessen wird, kann ein Schreibvorgang einen Stapel von 12 Zeitreihenereignissen (998 Byte) enthalten. Die Gesamtzahl der monatlichen Schreibvorgänge wird wie folgt berechnet:

- 100 Zeitreihenereignisse = 9 Schreibvorgänge (12 Zeitreihenereignisse pro Schreibvorgang) alle 5 Sekunden
- x 12 Schreibvorgänge/Minute = 108 Schreibvorgänge
- x 60 Minuten/Stunde = 6.480 Schreibvorgänge
- x 24 Stunden/Tag = 155.520 Schreibvorgänge
- x 30 Tage/Monat = 4.665.600 Schreibvorgänge

Gesamtzahl der monatlichen Schreibvorgänge = 4.665.600

Batchverarbeitung von Zeitreihenereignissen und Verwendung gemeinsamer Attribute bei einem Schreibvorgang

Wenn Region, az, vpc und Kennzahlname in 100 EC2 Instanzen gleich sind, können die gemeinsamen Werte nur einmal pro Schreibvorgang angegeben werden und werden als gemeinsame Attribute bezeichnet. In diesem Fall beträgt die Größe der gemeinsamen Attribute 52 Byte und die Größe der Zeitreihenereignisse 27 Byte. Da jeder Schreibvorgang in Einheiten von 1 KiB gemessen wird, kann ein Schreibvorgang 36 Zeitreihenereignisse und gemeinsame Attribute enthalten, und die Gesamtzahl der monatlichen Schreibvorgänge wird wie folgt berechnet:

- 100 Zeitreihenereignisse = 3 Schreibvorgänge (36 Zeitreihenereignisse pro Schreibvorgang) alle 5 Sekunden
- x 12 Schreibvorgänge/Minute = 36 Schreibvorgänge
- x 60 Minuten/Stunde = 2.160 Schreibvorgänge
- x 24 Stunden/Tag = 51.840 Schreibvorgänge
- x 30 Tage/Monat = 1.555.200 Schreibvorgänge

Gesamtzahl der monatlichen Schreibvorgänge = 1.555.200

> **Note**
>
> Aufgrund der Verwendung von Batchverarbeitung, gemeinsamen Attributen und der Rundung der Schreibvorgänge auf Einheiten von 1 KB kann die Speichergröße der Zeitreihenereignisse von der Schreibgröße abweichen.

Speicher

Die Speichergröße jedes Zeitreihenereignisses im Speicher und im Magnetspeicher wird als Summe der Größe des Zeitstempels, der Dimensionsnamen, der Dimensionswerte, der Messnamen und der Messwerte berechnet. Die Größe des Zeitstempels beträgt 8 Byte. Die Größe von Dimensionsnamen, Dimensionswerten und Kennzahlnamen entspricht der Länge der UTF -8 codierten Byte jeder Zeichenfolge, die den Dimensionsnamen, den Dimensionswert und den Kennzahlnamen darstellt. Die Größe des Kennzahlwerts hängt vom Datentyp ab. Sie beträgt 1 Byte für boolesche Datentypen, 8 Byte für Bigint und Double und die Länge der UTF -8 codierten Byte für Zeichenketten. Jede Kennzahl wird als separater Datensatz in Amazon Timestream für gespeichert LiveAnalytics, d. h. wenn Ihr Zeitreihenereignis vier Kennzahlen hat, werden vier Datensätze für dieses Zeitreihenereignis gespeichert.

Betrachtet man das Beispiel eines Zeitreihenereignisses, das die CPU Auslastung einer EC2 Instance darstellt (sieheBerechnung der Schreibgröße eines Zeitreihenereignisses), wird die Speichergröße des Zeitreihenereignisses wie folgt berechnet:

- Zeit = 8 Byte
- erste Dimension = 15 Byte (`region+us-east-1`)
- zweite Dimension = 4 Byte (`az+1d`)

- dritte Dimension = 15 Byte (`vpc+vpc-1a2b3c4d`)
- vierte Dimension = 18 Byte (`hostname+host-24Gju`)
- Name der Maßnahme = 15 Byte (`cpu_utilization`)
- Wert der Kennzahl = 8 Byte

Speichergröße des Zeitreihenereignisses = 83 Byte

> **Note**
>
> Der Speicher wird in GB-Stunden und der Magnetspeicher in GB-Monat gemessen.

Abfragen

Abfragen werden auf der Grundlage der Dauer der von Ihrer Anwendung genutzten Timestream-Recheneinheiten (TCUs) in TCU Stunden berechnet, wie auf der Amazon Timestream Timestream-Preisseite angegeben. Amazon Timestream for LiveAnalytics 'Query Engine löscht irrelevante Daten während der Verarbeitung einer Abfrage. Abfragen mit Prognosen und Prädikaten, einschließlich Zeiträumen, Kennzahlnamen und/oder Dimensionsnamen, ermöglichen es der Engine zur Abfrageverarbeitung, eine beträchtliche Datenmenge zu bereinigen und tragen so zur Senkung der Abfragekosten bei.

Kostenoptimierung

Um die Kosten für Schreibvorgänge, Speicherung und Abfragen zu optimieren, verwenden Sie die folgenden bewährten Methoden mit Amazon Timestream für LiveAnalytics:

- Batch Sie mehrere Zeitreihenereignisse pro Schreibvorgang, um die Anzahl der Schreibanforderungen zu reduzieren.
- Erwägen Sie die Verwendung von Datensätzen mit mehreren Messwerten, die es Ihnen ermöglichen, mehrere Zeitreihenwerte in einer einzigen Schreibanforderung zu schreiben und Ihre Daten kompakter zu speichern. Dadurch werden die Anzahl der Schreibanforderungen sowie die Datenspeicher- und Abfragekosten reduziert.
- Verwenden Sie beim Batching gemeinsame Attribute, um mehr Zeitreihenereignisse pro Schreibvorgang zu stapeln und so die Anzahl der Schreibanforderungen weiter zu reduzieren.
- Stellen Sie die Datenspeicherung des Speicherspeichers so ein, dass sie den Anforderungen Ihrer Anwendung für die Verarbeitung spät eingehender Daten entspricht. Bei spät eintreffenden Daten

handelt es sich um eingehende Daten, deren Zeitstempel vor der aktuellen Uhrzeit und außerhalb der Aufbewahrungsfrist des Speicherspeichers liegt.

- Stellen Sie die Datenspeicherung des Magnetspeichers so ein, dass sie Ihren langfristigen Datenspeicheranforderungen entspricht.
- Geben Sie beim Schreiben von Abfragen nur die Kennzahlen- und Dimensionsnamen an, die für die Abfrage unbedingt erforderlich sind. Das Hinzufügen von überflüssigen Spalten erhöht die Anzahl der Datenscans und damit auch die Abfragekosten. Es wird empfohlen, die Abfrageergebnisse zu überprüfen, um die Effizienz der Bereinigung der enthaltenen Dimensionen und Kennzahlen zu beurteilen.
- Geben Sie nach Möglichkeit einen Zeitraum in die WHERE Klausel Ihrer Abfrage ein. Wenn Sie beispielsweise nur die Daten der letzten Stunde in Ihrem Datensatz benötigen, fügen Sie ein Zeitprädikat wie `time > ago(1h)` hinzu.
- Wenn eine Abfrage auf eine Teilmenge von Kennzahlen in einer Tabelle zugreift, nehmen Sie die Kennzahlnamen immer in die WHERE Klausel der Abfrage auf.
- Wenn Sie mit der Ausführung einer Abfrage begonnen haben und feststellen, dass die Abfrage nicht die gewünschten Ergebnisse zurückgibt, brechen Sie die Abfrage ab, um Kosten zu sparen.

Überwachung mit Amazon CloudWatch

Sie können Timestream für die LiveAnalytics Nutzung von Amazon überwachen CloudWatch, das Rohdaten von Timestream sammelt und zu lesbaren LiveAnalytics Metriken verarbeitet. near-real-time Diese Statistiken werden zwei Wochen lang aufgezeichnet, damit Sie auf Verlaufsinformationen zugreifen können und einen besseren Überblick darüber erhalten, wie Ihre Webanwendung oder der Service ausgeführt werden. Standardmäßig wird Timestream für LiveAnalytics Metrikdaten automatisch CloudWatch in Zeitabständen von 1 Minute oder 15 Minuten gesendet. Weitere Informationen finden Sie unter Was ist Amazon CloudWatch? im CloudWatch Amazon-Benutzerhandbuch.

Themen

- Wie verwende ich Timestream für LiveAnalytics Metriken?
- Timestream für LiveAnalytics Metriken und Dimensionen
- Erstellung von CloudWatch Alarmen zur Überwachung von Timestream LiveAnalytics

Wie verwende ich Timestream für LiveAnalytics Metriken?

Die von Timestream für gemeldeten Metriken LiveAnalytics bieten Informationen, die Sie auf unterschiedliche Weise analysieren können. In der folgenden Liste finden Sie einige häufige Verwendungszwecke für die Metriken. Es handelt sich dabei um Vorschläge für den Einstieg und nicht um eine umfassende Liste.

Wie kann ich ...?	Relevante Metriken
`How can I determine if any system errors occurred?`	Sie können überwachen`SystemErrors` , ob Anfragen zu einem Serverfehlercode geführt haben. In der Regel sollte diese Metrik gleich Null sein. Ist dies nicht der Fall, sollten Sie das genauer untersuchen.
`How can I monitor the amount of data in the memory store?`	Sie können eine Überwachung `MemoryCumulativeBytesMetered` über den angegebenen Zeitraum durchführen, um die Menge der im Speicher gespeicherten Daten in Byte zu überwachen. Diese Metrik wird stündlich ausgegeben, und Sie können die in einem Konto gespeicherten Byte sowie die Datenbankgranularität verfolgen. Der Speicher wird in GB-Stunden gemessen (die Kosten für das Speichern von 1 GB an Daten für eine Stunde). Wenn Sie also den Stundenwert von `MemoryCumulativeBytesMetered` mit den Preisen für GB-Stunden in Ihrer Region multiplizieren, erhalten Sie die pro Stunde anfallenden Kosten. Dimensionen: Vorgang (Speicher), DatabaseName Name der Metrik
`How can I monitor the amount of data in the magnetic store?`	Sie können das `MagneticCumulativeBytesMetered` System über den angegebenen Zeitraum überwachen, um die Menge der im Magnetspeicher gespeicherten Daten in Byte zu überwachen. Diese Metrik wird stündlich ausgegeben, und Sie können die in einem Konto gespeicherten Byte sowie die Datenbankgranularität verfolgen. Der Speicherplatz wird in GB pro Monat abgerechnet (die Kosten für die Speicherung von 1 GB an Daten für einen Monat). Wenn Sie also den Stundenwert von `MagneticCumulativeBytesMetered` mit den

Wie kann ich ...?	Relevante Metriken
	Preisen für GB-Monat in Ihrer Region multiplizieren, erhalten Sie die pro Stunde anfallenden Kosten. Wenn der Wert von beispielsweise 107374182400 Byte (100 GB) `MagneticCumulativeBytesMetered` ist, dann entspricht die stündliche Gebühr für 1 GB Daten im Magnetspeicher = (0,03) (US-East-1-Preis)/(30,4*24). Wenn Sie diesen Wert mit dem in GB multiplizieren, erhalten Sie ~0,004 $ für diese Stunde. `MagneticCumulativeBytesMetered` Dimensionen: Vorgang (Speicher), Name der Metrik DatabaseName
`How can I monitor the data scanned by queries?`	Sie können die Daten `CumulativeBytesMetered` über den angegebenen Zeitraum überwachen, um die Daten zu überwachen, die von Abfragen (in Byte) gescannt wurden, die an Timestream gesendet wurden. LiveAnalytics Diese Metrik wird nach der Ausführung der Abfrage ausgegeben, und Sie können die gescannten Daten mit Konto- und Datenbank granularität verfolgen. Sie können die Abfragekosten für einen bestimmten Zeitraum berechnen, indem Sie den Wert der Metrik mit den Preisen pro gescanntem GB in Ihrer Region multiplizieren. Die von geplanten Abfragen gescannten Byte werden in dieser Metrik berücksichtigt. Dimensionen: Vorgang (Abfrage) DatabaseName, Name der Metrik

Wie kann ich ...?	Relevante Metriken
How can I monitor the data scanned by scheduled queries?	Sie können die Daten `CumulativeBytesMetered` über den angegebenen Zeitraum überwachen, um die Daten zu überwachen, die durch geplante Abfragen (in Byte) gescannt wurden, die von Timestream for LiveAnalytics ausgeführt werden. Diese Metrik wird nach der Ausführung der Abfrage ausgegeben, und Sie können die gescannten Daten mit Konto- und Datenbankgranularität verfolgen. Sie können die Abfragekosten für einen bestimmten Zeitraum berechnen, indem Sie den Wert der Metrik mit den Preisen pro gescanntem GB in Ihrer Region multiplizieren. **Note** Die gemessenen Byte werden auch in der Abfrage berücksichtigt. `CumulativeBytesMetered` Dimensionen: Operation (TriggeredScheduledQuery) DatabaseName, Metrikname

Wie kann ich ...?	Relevante Metriken
How can I monitor the number of records ingested?	Sie können die Überwachung `NumberOfRecords` über den angegebenen Zeitraum durchführen, um die Anzahl der aufgenommenen Datensätze zu überwachen. Sie können die in einem Konto gespeicherten Byte sowie die Datenbankgranularität verfolgen. Sie können diese Metrik auch verwenden, um die Schreibvorgänge von geplanten Abfragen zu überwachen, wenn Abfrageergebnisse in eine separate Tabelle geschrieben werden. Bei Verwendung von wird die `WriteRecords` API Metrik für jede `WriteRecords` Anforderung ausgegeben, wobei die CloudWatch Operations-Dimension wie folgt lautet`WriteRecords` . Bei Verwendung der Option `BatchLoad` oder `ScheduledQuery` APIs wird die Metrik in vom Dienst festgelegten Intervallen ausgegeben, bis die Aufgabe abgeschlossen ist. Die CloudWatch Operations-Dimension für diese Metrik ist entweder `BatchLoad` oder`ScheduledQuery` , je nachdem, welche verwendet API wird. Dimensionen: Operation (WriteRecords, BatchLoad, oder ScheduledQuery) DatabaseName, Metrikname

Wie kann ich ...?	Relevante Metriken
How can I monitor the cost of records ingested?	Sie können die Anzahl der aufgenommenen Byte`CumulativeBytesMetered`, für die Kosten anfallen, überwachen, um sie zu überwachen. Sie können sowohl die in einem Konto gespeicherten Byte als auch anhand der Datenbankgranularität verfolgen. Aufgenommene Datensätze werden in kumulativen Byte gemessen. Wenn Sie den Wert `CumulativeBytesMetered` von mit den Preisen für Schreibvorgänge in Ihrer Region multiplizieren, erhalten Sie die anfallenden Aufnahmekosten. Bei Verwendung von wird diese Metrik für jede `WriteRecords` Anfrage ausgegeben `WriteRecords` API, wobei die CloudWatch Operations-Dimension wie folgt lautet. `WriteRecords` Bei Verwendung der Option `BatchLoad` oder `ScheduledQuery` API wird die Metrik in vom Dienst festgelegten Intervallen ausgegeben, bis die Aufgabe abgeschlossen ist. Die CloudWatch Operationsdimension für diese Metrik ist `BatchLoad` oder, `ScheduledQuery` je nachdem, welche verwendet API wird.. Dimensionen: Operation (WriteRecords, BatchLoad, oder ScheduledQuery) DatabaseName, Metrikname

Wie kann ich ...?	Relevante Metriken
How can I monitor the Timestream Compute Units (TCUs) used in my account?	Sie können eine Überwachung QueryTCU über den angegeben en Zeitraum durchführen, um die Recheneinheiten zu überwachen, die für die Abfrage-Arbeitslast im Konto verbrauch t werden. Diese Metrik wird mit den maximalen und minimalen Recheneinheiten für jede Minute während des aktiven Abfrage-Workloads vom Konto ausgegeben. Einheiten: Count Gültige Statistiken: Minimum, Maximum Metrisch: ResourceCount Abmessungen: Service: Timestream Resource: QueryTCU,Type: Resource, Class: OnDemand

Timestream für LiveAnalytics Metriken und Dimensionen

Wenn Sie mit Timestream for interagieren LiveAnalytics, sendet es die folgenden Metriken und Dimensionen an Amazon CloudWatch. Alle Metriken werden aggregiert und jede Minute gemeldet. Sie können die folgenden Verfahren verwenden, um die Metriken für Timestream anzuzeigen. LiveAnalytics

So zeigen Sie Metriken mit der Konsole an CloudWatch

Metriken werden zunächst nach dem Service-Namespace und anschließend nach den verschiedenen Dimensionskombinationen in den einzelnen Namespaces gruppiert.

1. Öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Ändern Sie, falls erforderlich, die Region. Wählen Sie in der Navigationsleiste die Region aus, in der sich Ihre AWS Ressourcen befinden. Weitere Informationen finden Sie unter AWS -Service-Endpunkte.
3. Wählen Sie im Navigationsbereich Metriken aus.
4. Wählen Sie unter dem Tab Alle Metriken AWS/Timestream for LiveAnalytics.

Um Metriken anzuzeigen, verwenden Sie AWS CLI

- Geben Sie als Eingabeaufforderung den folgenden Befehl ein.

```
aws cloudwatch list-metrics --namespace "AWS/Timestream"
```

Dimensionen für Timestream für Metriken LiveAnalytics

Die Metriken für Timestream for LiveAnalytics werden anhand der Werte für das Konto, den Tabellennamen oder den Vorgang qualifiziert. Sie können die CloudWatch Konsole verwenden, um Timestream für LiveAnalytics Daten in einer der Dimensionen in der folgenden Tabelle abzurufen:

Dimension	Beschreibung
`DatabaseName`	Diese Dimension beschränkt die Daten auf einen bestimmten Timestream für die LiveAnalytics Datenbank. Bei diesem Wert kann es sich um eine beliebige Datenbank in der aktuellen Region und im aktuellen Konto handeln AWS
`Operation`	Diese Dimension beschränkt die Daten auf einen der LiveAnaly tics Timestream-Werte für Operationen wie`Storage`, `WriteRecords BatchLoad` , oder`ScheduledQuery` . Eine Liste der verfügbaren Werte finden Sie in der Timestream for LiveAnalytics API Query-Referenz.
`TableName`	Diese Dimension beschränkt die Daten auf eine bestimmte Tabelle in einer Timestream-Datenbank. LiveAnalyticss

Important

`CumulativeBytesMetered`, `UserErrors` und `SystemErrors` Metriken haben nur die `Operation` Dimension. `SuccessfulRequestLatency`Metriken haben immer eine `Operation` Dimension, können aber auch die `TableName` Dimensionen `DatabaseName` und haben, abhängig vom Wert von`Operation`. Das liegt daran, dass Timestream für

Operationen LiveAnalytics auf Tabellenebene `DatabaseName` und `TableName` als Dimensionen hat, Operationen auf Kontoebene jedoch nicht.

Timestream für Metriken LiveAnalytics

> **Note**
>
> Amazon CloudWatch aggregiert den gesamten folgenden Timestream für LiveAnalytics Metriken in Intervallen von einer Minute.

Allgemeine Metriken

Metrik	Beschreibung
`SuccessfulRequestLatency`	Die erfolgreichen Anfragen an Timestream für LiveAnalytics den angegebenen Zeitraum. SuccessfulRequestLatency kann zwei verschiedene Arten von Informationen bereitstellen: • Die verstrichene Zeit für erfolgreiche Anfragen (Minimum, Maximum, Summe oder Durchschnitt). • Die Anzahl erfolgreicher Anforderungen (SampleCount). SuccessfulRequestLatency spiegelt nur Aktivitäten innerhalb von Timestream für wider LiveAnalytics und berücksichtigt nicht die Netzwerklatenz oder clientseitige Aktivitäten. Einheiten: `Milliseconds` Dimensionen • `DatabaseName`

Metrik	Beschreibung
	• TableName • Operation Gültige Statistiken: • Minimum • Maximum • Average • SampleCount • P10 • p50 • p90 • p95 • p99

Schreib- und Speichermetriken

Metrik	Beschreibung
MagneticStoreRejectedRecordCount	Die Anzahl der im Magnetspeicher geschrieb enen Datensätze, die asynchron zurückgew iesen wurden. Dies kann passieren, wenn der neue Datensatz eine Version hat, die niedriger als die aktuelle Version ist, oder wenn der neue Datensatz eine Version hat, die der aktuellen Version entspricht, aber andere Daten enthält. Einheiten: Count Dimensionen • DatabaseName • TableName

Metrik	Beschreibung
	• `Operation` Gültige Statistiken: • `Sum` • `SampleCount`
`MagneticStoreRejectedUpload UserFailures`	Die Anzahl der von Magnetic Store abgelehnten Datensatzberichte, die aufgrund von Benutzerfehlern nicht hochgeladen wurden. Dies kann auf nicht korrekt konfigurierte IAM Berechtigungen oder auf einen gelöschten S3-Bucket zurückzuführen sein. Einheiten: `Count` Dimensionen • `DatabaseName` • `TableName` • `Operation` Gültige Statistiken: • `Sum` • `SampleCount`

Metrik	Beschreibung
`MagneticStoreRejectedUpload SystemFailures`	Die Anzahl der von Magnetic Store abgelehnten Datensatzberichte, die aufgrund von Systemfehlern nicht hochgeladen wurden. Einheiten: `Count` Dimensionen • `DatabaseName` • `TableName` • `Operation` Gültige Statistiken: • `Sum` • `SampleCount`
`ActiveMagneticStorePartitions`	Die Anzahl der Magnetspeicherpartitionen, die zu einem bestimmten Zeitpunkt aktiv Daten aufnehmen. Einheiten: `Count` Dimensionen • `DatabaseName` • `Operation` Gültige Statistiken: • `Sum` • `SampleCount`

Metrik	Beschreibung
`MagneticStorePendingRecordsLatency`	Der älteste Schreibvorgang in einen Magnetspeicher, der nicht für Abfragen verfügbar ist. In den Magnetspeicher geschriebene Datensätze können innerhalb von 6 Stunden abgefragt werden. Einheiten: `Milliseconds` Dimensionen • `DatabaseName` • `TableName` • `Operation` Gültige Statistiken: • `Minimum` • `Maximum` • `Average` • `SampleCount` • `P10` • `p50` • `p90` • `p95` • `p99`

Metrik	Beschreibung
`MemoryCumulativeBytesMetered`	Die Menge der im Speicher gespeicherten Daten in Byte Einheiten: `Bytes` Maße: `Operation` Gültige Statistiken: • `Average`
`MagneticCumulativeBytesMetered`	Die im Magnetspeicher gespeicherte Datenmenge in Byte Einheiten: `Bytes` Maße: `Operation` Gültige Statistiken: • `Average`
`CumulativeBytesMetered`	Die durch die Aufnahme in Timestream gemessene Datenmenge in Byte. LiveAnalytics Einheiten: `Bytes` Maße: `Operation` Gültige Statistiken: `Sum`
`NumberOfRecords`	Die Anzahl der in Timestream aufgenommenen Datensätze für. LiveAnalytics Einheiten: `Count` Maße: `Operation` Gültige Statistiken: `Sum`

Metriken abfragen

Metrik	Beschreibung
`CumulativeBytesMetered`	Die Datenmenge, für die Abfragen, die an Timestream gesendet wurden, gescannt wurde LiveAnalytics, in Byte. Einheiten: `Bytes` Maße: `Operation` Gültige Statistiken: • `Sum`
`ResourceCount`	Die Timestream-Recheneinheiten (TCUs), die für den Abfrage-Workload im Konto verbrauch t wurden. Diese Metrik wird mit den maximalen und minimalen Recheneinheiten für jede Minute während des aktiven Abfrage-Workloads vom Konto ausgegeben. Einheiten: `Count` Gültige Statistiken: Minimum, Maximum Abmessungen: `Service: Timestrea m Resource: QueryTCU`,`Type: Resource`, `Class: OnDemand`

Fehlermetriken

Metrik	Beschreibung
`SystemErrors`	Die Anfragen an Timestream LiveAnaly tics dafür generieren SystemError während des angegebenen Zeitraums eine. A weist SystemError normalerweise auf einen internen Dienstfehler hin.

Metrik	Beschreibung
	Einheiten: `Count` Maße: `Operation` Gültige Statistiken: • `Sum` • `SampleCount`
`UserErrors`	Anfragen an Timestream zu LiveAnalytics diesem Thema erzeugen während des angegebenen Zeitraums einen InvalidRequest Fehler. Ein weist InvalidRequest normalerw eise auf einen clientseitigen Fehler hin, z. B. eine ungültige Kombination von Parametern, einen Versuch, eine nicht vorhandene Tabelle zu aktualisieren, oder eine falsche Anforderu ngssignatur. UserErrors stellt die Summe der ungültigen Anfragen für die aktuelle AWS Region und das aktuelle Konto dar. AWS Einheiten: `Count` Maße: `Operation` Gültige Statistiken: • `Sum` • `SampleCount`

Important

Nicht alle Statistiken, wie `Average` oder `Sum` sind für jede Metrik anwendbar. Alle diese Werte sind jedoch über den Timestream für die LiveAnalytics Konsole, über die CloudWatch Konsole oder AWS SDKs für alle Metriken verfügbar. AWS CLI

Erstellung von CloudWatch Alarmen zur Überwachung von Timestream LiveAnalytics

Sie können einen CloudWatch Amazon-Alarm für Timestream erstellen LiveAnalytics , der eine Amazon Simple Notification Service (AmazonSNS) -Nachricht sendet, wenn sich der Status des Alarms ändert. Ein Alarm überwacht eine Metrik über einen bestimmten, von Ihnen definierten Zeitraum. Der Alarm führt eine oder mehrere Aktionen durch, basierend auf dem Wert der Metrik im Vergleich zu einem bestimmten Schwellenwert in einer Reihe von Zeiträumen. Die Aktion ist eine Benachrichtigung, die an ein SNS Amazon-Thema oder eine Auto Scaling Scaling-Richtlinie gesendet wird.

Alarme lösen nur Aktionen für anhaltende Statusänderungen aus. CloudWatch Alarme lösen keine Aktionen aus, nur weil sie sich in einem bestimmten Zustand befinden. Der Status muss sich geändert haben und für eine festgelegte Anzahl an Zeiträumen aufrechterhalten worden sein.

Weitere Informationen zum Erstellen von CloudWatch Alarmen finden Sie unter Verwenden von Amazon CloudWatch Alarms im CloudWatch Amazon-Benutzerhandbuch.

Fehlerbehebung

Dieser Abschnitt enthält Informationen zur Fehlerbehebung bei Timestream für LiveAnalytics.

Themen

- Umgang mit Drosseln WriteRecords
- Umgang mit abgelehnten Datensätzen
- Fehlerbehebung UNLOAD von Timestream für LiveAnalytics
- Timestream für LiveAnalytics bestimmte Fehlercodes

Umgang mit Drosseln WriteRecords

Ihre Speicherspeicher-Schreibanforderungen an Timestream können gedrosselt werden, wenn Timestream skaliert wird, um sich an die Datenaufnahmeanforderungen Ihrer Anwendung anzupassen. Wenn Ihre Anwendungen auf Ausnahmen bei der Drosselung stoßen, müssen Sie weiterhin Daten mit demselben (oder einem höheren) Durchsatz senden, damit Timestream automatisch an die Anforderungen Ihrer Anwendung angepasst werden kann.

Ihre Magnetspeicher-Schreibanforderungen an Timestream können gedrosselt werden, wenn die maximale Anzahl von Magnetspeicherpartitionen, die Daten empfangen, erreicht ist. Es wird eine Drosselungsmeldung angezeigt, in der Sie aufgefordert werden, die Cloudwatch-Metrik

für diese Datenbank zu überprüfen. `ActiveMagneticStorePartitions` Es kann bis zu 6 Stunden dauern, bis diese Drosselung behoben ist. Um diese Drosselung zu vermeiden, sollten Sie den Speicherspeicher für Datenaufnahme-Workloads mit hohem Durchsatz verwenden. Bei der Aufnahme durch Magnetspeicher können Sie die Aufnahme gezielt in weniger Partitionen durchführen, indem Sie die Anzahl der Datenreihen und die Dauer der Aufnahme einschränken

Weitere Informationen zu bewährten Methoden für die Datenaufnahme finden Sie unter. Schreibt

Umgang mit abgelehnten Datensätzen

Wenn Timestream Datensätze ablehnt, erhalten Sie eine `RejectedRecordsException` mit Einzelheiten zur Ablehnung. Weitere Informationen zum Extrahieren dieser Informationen aus der Antwort finden Sie unter Behandlung von Schreibfehlern. WriteRecords

Alle Ablehnungen werden in diese Antwort aufgenommen, mit Ausnahme von Aktualisierungen des Magnetspeichers, bei denen die Version des neuen Datensatzes kleiner oder gleich der Version des vorhandenen Datensatzes ist. In diesem Fall aktualisiert Timestream den vorhandenen Datensatz mit der höheren Version nicht. Timestream lehnt den neuen Datensatz mit einer niedrigeren oder gleichwertigen Version ab und schreibt diese Fehler asynchron in Ihren S3-Bucket. Um diese asynchronen Fehlerberichte zu erhalten, sollten Sie die `MagneticStoreRejectedDataLocation` Eigenschaft in `MagneticStoreWriteProperties` Ihrer Tabelle festlegen.

Fehlerbehebung UNLOAD von Timestream für LiveAnalytics

Im Folgenden finden Sie Anleitungen zur Fehlerbehebung im Zusammenhang mit dem UNLOAD Befehl.

Kategorie	Fehlermeldung	Fehlerbehebung
Länge des S3-Schlüssels	UNLOADDer Schlüssel der Ergebnisdatei, wenn das im Ziel angegebene S3-Präfix [%s] verwendet wird, überschreitet die zulässige S3-Schlüssellänge. Weitere Informationen finden Sie in der Dokumentation.	Beim Exportieren von Abfrageergebnissen mithilfe der `UNLOAD` Anweisung überschreitet die S3-Schlüssellänge, die sich aus der Summe der Länge des S3-Bucket-Namens und des Präfixes zusammensetzt, die maximal unterstützte

Kategorie	Fehlermeldung	Fehlerbehebung
		S3-Schlüssellänge. Wir empfehlen, die Länge Ihres Präfix- oder Bucket-Namens zu reduzieren.
	UNLOADDer Schlüssel der Ergebnisdatei wird bei Verwendung von partition ed_by [%s] die zulässige S3-Schlüssellänge überschreiten. Weitere Informationen finden Sie in der Dokumentation.	Beim Exportieren von Abfrageergebnissen mithilfe der UNLOAD Anweisung überschreitet die Länge des S3-Schlüssels mithilfe der partitioned_by-Spalte die maximal unterstützte S3-Schlüssellänge. Wir empfehlen, mit einer alternati ven Spalte zu partitionieren oder die Länge der partition ed_column zu reduzieren (falls möglich).

Kategorie	Fehlermeldung	Fehlerbehebung
	UNLOADDer Schlüssel der Ergebnisdatei wird bei Verwendung des S3-Präfixes [%s] zusammen mit partition ed_by [%s] die zulässige S3-Schlüssellänge überschreiten. Weitere Informationen finden Sie in der Dokumentation.	Beim Exportieren von Abfrageergebnissen mithilfe der UNLOAD Anweisung überschreitet die Länge des S3-Schlüssels, die sich aus der Summe der Länge des S3-Bucket-Namens, des Präfixes und des partitioned_by-Spaltennamens zusammensetzt, die maximal unterstützte S3-Schlüssellänge. Wir empfehlen, Ihr Präfix und die Länge Ihres Bucket-Namens zu reduzieren oder eine alternative Spalte zur Partitionierung Ihrer Daten zu verwenden.
	Der generierte S3-Objekt schlüssel: %s ist zu lang. Weitere Informationen finden Sie in der Dokumentation.	Bei der Verarbeitung Ihrer Abfrage mithilfe der UNLOAD Anweisung überschreitet einer der Werte in der partitionierten Spalte die maximal unterstützte S3-Schlüssellänge. Die Partitionsspalte und der Wert befinden sich im generierten Objektschlüssel.

Kategorie	Fehlermeldung	Fehlerbehebung
S3 drosselt	Wir haben festgestellt, dass Amazon S3 die Schreibvorgänge vom Befehl aus UNLOAD drosselt. Weitere Informationen finden Sie in der Amazon Timestream-Dokumentation	Die S3-Dokumentation finden Sie hier. Die API S3-Anrufrate könnte gedrosselt werden, wenn mehrere Leser/Schreiber auf denselben Ordner zugreifen. Bitte überprüfen Sie das Anrufvolumen im angegebenen Bucket. Wenn Sie denselben Bucket für mehrere gleichzeitige `UNLOAD` Abfragen verwenden, versuchen Sie, verschiedene Buckets für dasselbe zu verwenden. Wenn Sie denselben Bucket für mehrere andere Operationen als Timestream für verwenden, sollten Sie erwägen LiveAnalytics `UNLOAD`, die `UNLOAD` Ergebnisse in einen separaten Bucket zu verschieben.

Timestream für LiveAnalytics bestimmte Fehlercodes

Dieser Abschnitt enthält die spezifischen Fehlercodes für Timestream für. LiveAnalytics

Timestream für Schreibfehler LiveAnalytics API

InternalServerException

- HTTPStatuscode: 500

ThrottlingException

- HTTPStatuscode: 429

ValidationException

HTTPStatuscode: 400

ConflictException

HTTPStatuscode: 409

AccessDeniedException

Sie haben keinen ausreichenden Zugriff zum Durchführen dieser Aktion.

HTTPStatuscode: 403

ServiceQuotaExceededException

HTTPStatuscode: 402

ResourceNotFoundException

HTTPStatuscode: 404

RejectedRecordsException

HTTPStatuscode: 419

InvalidEndpointException

HTTPStatuscode: 421

Timestream für LiveAnalytics Abfragefehler API

ValidationException

HTTPStatuscode: 400

QueryExecutionException

HTTPStatuscode: 400

ConflictException

HTTPStatuscode: 409

ThrottlingException

HTTPStatuscode: 429

InternalServerException

HTTPStatuscode: 500

InvalidEndpointException

HTTPStatuscode: 421

Kontingente

In diesem Thema werden aktuelle Kontingente, auch als Limits bezeichnet, in Amazon Timestream für LiveAnalytics beschrieben. Jedes Kontingent gilt pro Region, sofern nicht anders angegeben.

Themen

Standardkontingente

Die folgende Tabelle enthält den Timestream für LiveAnalytics Kontingente und die Standardwerte.

displayName	Beschreibung	defaultValue
Datenbanken pro Konto	Die maximale Anzahl von Datenbanken, die Sie pro AWS-Konto erstellen können.	500
Tabellen pro Konto	Die maximale Anzahl von Tabellen, die Sie pro erstellen können AWS-Konto.	50000

displayName	Beschreibung	defaultValue
Drosselrate für CRUD APIs	Die maximale Anzahl von Create/Update/List/Describe /Delete database/table/sch eduled API Abfrageanfragen, die pro Sekunde und Konto in der aktuellen Region zulässig sind.	1
Geplante Abfragen pro Konto	Die maximale Anzahl von geplanten Abfragen, die Sie pro erstellen können AWS-Konto.	10000
Maximale Anzahl aktiver Magnetspeicherpartitionen	Die maximale Anzahl aktiver Magnetspeicherpartitionen pro Datenbank. Eine Partition kann nach der Aufnahme bis zu sechs Stunden aktiv bleiben.	250
maxQueryTCU	Die maximale Anzahl an AbfragenTCUs, die Sie für Ihr Konto festlegen können.	1000

Service Limits

Die folgende Tabelle enthält den Timestream für LiveAnalytics Service-Limits und die Standardwerte. Informationen zum Bearbeiten der Datenspeicherung für eine Tabelle von der Konsole aus finden Sie unter Tabelle bearbeiten.

displayName	Beschreibung	defaultValue
Künftiger Erfassungszeitraum in Minuten	Die maximale Vorlaufzeit (in Minuten) für Ihre Zeitreihe ndaten im Vergleich zur	15

displayName	Beschreibung	defaultValue
	aktuellen Systemzeit. Wenn der future Aufnahmezeitraum beispielsweise 15 Minuten beträgt, akzeptiert Timestrea m for LiveAnalytics Daten, die der aktuellen Systemzeit bis zu 15 Minuten voraus sind.	
Minimale Aufbewahr ungsdauer für In-Memory-Speicher in Stunden	Die minimale Dauer (in Stunden), für die Daten pro Tabelle im Magnetspeicher aufbewahrt werden müssen.	1
Maximale Aufbewahr ungsdauer für In-Memory-Speicher in Stunden	Die maximale Dauer (in Stunden), für die Daten pro Tabelle im Magnetspeicher aufbewahrt werden können.	8766
Minimale Aufbewahr ungsdauer für Magnetspeicher in Tagen	Die minimale Dauer (in Tagen), für die Daten pro Tabelle im Magnetspeicher aufbewahrt werden müssen.	1
Maximale Aufbewahr ungsdauer für Magnetspeicher in Tagen	Die maximale Dauer (in Tagen), für die Daten im Magnetspeicher aufbewahrt werden können. Dieser Wert entspricht 200 Jahren.	73000
Standard-Aufbewahr ungsdauer für Magnetspeicher in Tagen	Der Standardwert (in Tagen), für den Daten pro Tabelle im Magnetspeicher aufbewahrt werden. Dieser Wert entsprich t 200 Jahren.	73000

displayName	Beschreibung	defaultValue
Standardmäßige Aufbewahrungsdauer für den Speicherspeicher in Stunden	Die Standarddauer (in Stunden), für die Daten im Speicherspeicher aufbewahrt werden.	6
Dimensionen pro Tabelle	Die maximale Anzahl der Dimensionen pro Tabelle.	128
Kennzahlnamen pro Tabelle	Die maximale Anzahl eindeutiger Kennzahlnamen pro Tabelle.	8192
Größe des Wertpaares aus Dimensionsname und -wert pro Serie	Die maximale Größe des Wertpaares aus Dimensionsname und -wert pro Serie.	2 Kilobyte
Maximale Datensatzgröße	Die maximale Größe eines Datensatzes.	2 Kilobyte
Datensätze pro WriteRecords API Anfrage	Die maximale Anzahl von Datensätzen in einer WriteRecords API Anfrage.	100
Länge des Dimensionsnamens	Die maximale Anzahl von Byte für einen Dimensionsnamen.	60 Byte
Länge des Kennzahlnamens	Die maximale Anzahl von Byte für einen Measure-Namen.	256 Byte
Länge des Datenbanknamens	Die maximale Anzahl von Byte für einen Datenbanknamen.	256 Byte
Länge des Tabellnamens	Die maximale Anzahl von Byte für einen Tabellennamen.	256 Byte

displayName	Beschreibung	defaultValue
QueryString Länge in KiB	Die maximale Länge (in KiB) einer Abfragezeichenfolge in UTF -8 codierten Zeichen für eine Abfrage.	256
Ausführungsdauer für Abfragen in Stunden	Die maximale Ausführun gsdauer (in Stunden) für eine Abfrage. Abfragen, die länger dauern, führen zu einem Timeout.	1
Einblicke in Abfragen	Die maximale Anzahl von API Query-Anfragen, die mit aktivierten Query Insights pro Sekunde und Konto in der aktuellen Region zulässig sind.	1
Metadatengröße für Abfrageer gebnis	Die maximale Meta-Date ngröße für ein Abfrageer gebnis.	100 Kilobyte
Datengröße für Abfrageer gebnis	Die maximale Datengröße für ein Abfrageergebnis.	5 Gigabyte
Kennzahlen pro Datensatz mit mehreren Kennzahlen	Maximale Anzahl der Kennzahlen pro Datensatz mit mehreren Kennzahlen.	256
Kennzahlenwertgröße pro Datensatz mit mehreren Kennzahlen	Maximale Kennzahle nwertgröße pro Datensatz mit mehreren Kennzahlen.	2048

displayName	Beschreibung	defaultValue
Eindeutige Kennzahlen in Datensätzen mit mehreren Kennzahlen pro Tabelle	Die eindeutigen Kennzahlen in allen Datensätzen mit mehreren Kennzahlen, die in einer einzigen Tabelle definiert sind.	1024
Timestream-Recheneinheiten (TCUs) pro Konto	Das Standardmaximum TCUs pro Konto.	200

Unterstützte Datentypen

In der folgenden Tabelle werden die unterstützten Datentypen für Kennzahl- und Dimensionswerte beschrieben.

Beschreibung	Timestream für den Wert LiveAnalytics
Unterstützte Datentypen für Kennzahlwerte.	Big Int, Double, Zeichenfolge, Boolean, Timestamp MULTI
Unterstützte Datentypen für Dimensionswerte.	String

Batch-Laden

Die aktuellen Kontingente, auch als Grenzwerte bezeichnet, beim Laden von Batches lauten wie folgt.

Beschreibung	Timestream für den Wert LiveAnalytics
Max. Größe der Batch-Load-Aufgabe	Die maximale Größe von Batch-Load-Aufgaben darf 100 GB nicht überschreiten.
Anzahl der Dateien	Eine Batch-Load-Aufgabe kann nicht mehr als 100 Dateien enthalten .

Beschreibung	Timestream für den Wert LiveAnalytics
Maximale Dateigröße	Die maximale Dateigröße in einer Batch-Load-Task darf 5 GB nicht überschreiten.
CSVGröße der Dateizeile	Eine Zeile in einer CSV Datei darf 16 MB nicht überschreiten. Dies ist ein fester Grenzwert, der nicht erhöht werden kann.
Aktive Batch-Load-Aufgaben	Eine Tabelle kann nicht mehr als 5 aktive Batch-Load-Tasks haben und ein Konto darf nicht mehr als 10 aktive Batch-Load-Tasks haben. Timestream for LiveAnalytics drosselt neue Batch-Load-Aufgaben, bis mehr Ressourcen verfügbar sind.

Benennungseinschränkungen:

In der folgenden Tabelle werden Einschränkungen bei der Benennung beschrieben.

Beschreibung	Timestream für den Wert LiveAnalytics
Die maximale Länge eines Dimensionsnamens.	60 Byte
Die maximale Länge eines Kennzahlnamens.	256 Byte
Die maximale Länge eines Tabellen- oder Datenbank namens.	256 Byte
Tabelle und Datenbank name	• Wir empfehlen Ihnen, nicht zu verwendenSystemkennungen. • Kann a-z A-Z 0-9 _ (Unterstrich) - (Bindestrich) enthalten. (Punkt). • Alle Namen müssen mit UTF -8 kodiert sein und Groß- und Kleinschreibung beachten.

Beschreibung	Timestream für den Wert LiveAnalytics
	Note Tabellen- und Datenbanknamen werden unter Verwendung der Binärdarstellung UTF -8 verglichen. Das bedeutet, dass beim ASCII Zeichenvergleich zwischen Groß- und Kleinschr eibung unterschieden wird.
Name der Maßnahme	• Darf kein ':' Systemkennungen oder einen Doppelpunkt enthalten. • Darf nicht mit einem reservierten Präfix (`ts_`,`measure_value`) beginnen. **Note** Tabellen- und Datenbanknamen werden unter Verwendung der Binärdarstellung UTF -8 verglichen. Das bedeutet, dass beim ASCII Zeichenvergleich zwischen Groß- und Kleinschr eibung unterschieden wird.
Name der Dimension	• Darf keinen Doppelpunkt ':' oder ein doppeltes Anführungszeichen („) enthaltenSystemkennungen. • Darf nicht mit einem reservierten Präfix (`ts_`,`measure_value`) beginnen. • Darf keine Unicode-Zeichen [0,31] enthalten, die hier oder „\ u2028" oder „\ u2029" aufgeführt sind. **Note** Dimensions- und Kennzahlnamen werden unter Verwendun g der binären Darstellung -8 verglichen. UTF Das bedeutet, dass beim ASCII Zeichenvergleich zwischen Groß- und Kleinschreibung unterschieden wird.

Beschreibung	Timestream für den Wert LiveAnalytics
Alle Spaltennamen	Spaltennamen können nicht dupliziert werden. Da Datensätze mit mehreren Kennzahlen Dimensionen und Kennzahlen als Spalten darstellen, kann der Name einer Dimension nicht mit dem Namen einer Kennzahl identisch sein. Bei den Namen muss die Groß- und Kleinschreibung beachtet werden.

Reservierte Schlüsselwörter

Alle der folgenden sind reservierte Schlüsselwörter:

- ALTER
- AND
- AS
- BETWEEN
- BY
- 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
- IN
- INNER
- INSERT
- INTERSECT
- INTO
- IS
- JOIN
- LEFT
- LIKE
- LOCALTIME
- LOCALTIMESTAMP
- NATURAL
- NORMALIZE
- NOT

- NULL
- ON
- ODER
- ORDER
- OUTER
- PREPARE
- RECURSIVE
- RIGHT
- ROLLUP
- SELECT
- TABLE
- THEN
- TRUE
- UESCAPE
- UNION
- UNNEST
- USING
- VALUES
- WHEN
- WHERE
- WITH

Systemkennungen

Wir behalten uns die Spaltennamen „measure_value“, „ts_non_existent_col“ und „time“ als Timestream für Systemkennungen vor. LiveAnalytics Außerdem dürfen Spaltennamen nicht mit „ts_“ oder „measure_name“ beginnen. Bei Systemkennungen wird zwischen Groß- und Kleinschreibung unterschieden. Identifier wurden anhand der binären Darstellung von UTF -8 verglichen. Das bedeutet, dass beim Vergleich von Bezeichnern zwischen Groß- und Kleinschreibung unterschieden wird.

> **Note**
>
> Systemkennungen dürfen nicht für Dimensions- oder Kennzahlnamen verwendet werden. Wir empfehlen, keine Systemkennungen für Datenbank- oder Tabellennamen zu verwenden.

UNLOAD

Informationen zu den Einschränkungen im Zusammenhang mit dem `UNLOAD` Befehl finden Sie unter Verwenden UNLOAD zum Exportieren von Abfrageergebnissen aus Timestream nach S3.

Referenz zur Abfragesprache

> **Note**
>
> Diese Referenz zur Abfragesprache enthält die folgende Drittanbieterdokumentation der Trino Software Foundation (ehemals Presto Software Foundation), die unter der Apache License, Version 2.0, lizenziert ist. Sie dürfen diese Datei nur in Übereinstimmung mit dieser Lizenz verwenden. Eine Kopie der Apache-Lizenz, Version 2.0, finden Sie auf der Apache-Website.

Timestream for LiveAnalytics unterstützt eine umfangreiche Abfragesprache für die Arbeit mit Ihren Daten. Die verfügbaren Datentypen, Operatoren, Funktionen und Konstrukte finden Sie weiter unten.

In diesem Abschnitt können Sie auch sofort mit der Abfragesprache von Timestream beginnen. Beispielabfragen

Themen

- Mathematische Operatoren
- Mathematische Funktionen
- Zeichenfolgenoperatoren
- Zeichenfolgenfunktionen
- Array-Operatoren
- Array-Funktionen
- Bitweise-Funktionen
- Funktionen für reguläre Ausdrücke
- Operatoren für Datum und Uhrzeit
- Funktionen für Datum und Uhrzeit
- Aggregationsfunktionen
- Fensterfunktionen
- Beispielabfragen

Unterstützte Datentypen

LiveAnalyticsDie Abfragesprache von Timestream for unterstützt die folgenden Datentypen.

> **Note**
>
> Datentypen, die für Schreibvorgänge unterstützt werden, werden unter Datentypen beschrieben.

Datentyp	Beschreibung
`int`	Stellt eine 32-Bit-Ganzzahl dar.
`bigint`	Stellt eine 64-Bit-Ganzzahl mit Vorzeichen dar.
`boolean`	Einer der beiden Wahrheitswerte der Logik, `True` und`False`.
`double`	Stellt einen 64-Bit-Datentyp mit variabler Genauigkeit dar. Implementiert den IEEEStandard 754 für binäre Gleitkomma-Arithmetik.

Datentyp	Beschreibung
	Note Die Abfragesprache dient zum Lesen von Daten. Es gibt Funktionen für `Infinity` und `NaN` Doppelwerte, die in Abfragen verwendet werden können. Sie können diese Werte jedoch nicht in Timestream schreiben.
`varchar`	Zeichendaten variabler Länge mit einer maximalen Größe von 2 KB.
`array[T,...]`	Enthält ein oder mehrere Elemente eines angegebenen Datentyps *T*wobei *T* kann jeder der in Timestream unterstüt zten Datentypen sein.
`row(T,...)`	Enthält ein oder mehrere benannte Felder des Datentyps *T*. Die Felder können von jedem Datentyp sein, der von Timestream unterstützt wird, und auf sie wird mit dem Referenzoperator „Punktfeld“ zugegriffen: `.`
`date`	Stellt ein Datum im Formular dar*YYYY*`-`*MM*`-`*DD*. Wo *YYYY* ist das Jahr, *MM* ist der Monat und *DD* ist jeweils der Tag. Der unterstützte Bereich liegt zwischen `1970-01-01` und`2262-04-11` . Beispiel: `1971-02-03`

Datentyp	Beschreibung
`time`	Stellt die Tageszeit in dar UTC. Der `time` Datentyp wird in der Form *HH*`.`*MM*`.`*SS*`.`*sssssssss* `.` Unterstützt Nanosekun dengenauigkeit dargestellt. Beispiel: `17:02:07.496000000`
`timestamp`	Stellt eine Zeitinstanz mit einer Genauigkeit von Nanosekun den dar. UTC *YYYY*`-`*MM*`-`*DD* *hh*`:`*mm*`:`*ss*`.`*sssssssss* Query unterstützt Zeitstempel im Bereich bis`1677-09-21 00:12:44.000000000` `.2262-04-11 23:47:16.854775807`

Datentyp	Beschreibung

`interval`	Stellt ein Zeitintervall als Zeichenkettenliteral dar*X*`t`, das aus zwei Teilen besteht, *X* and *t*.
	
	X ist ein numerischer Wert, der größer oder gleich ist`0`, und *t* ist eine Zeiteinheit wie Sekunde oder Stunde. Die Einheit ist nicht pluralisiert. Die Zeiteinheit *t* is muss eines der folgenden Zeichenkettenliterale sein:
	
	• `nanosecond`
	• `microsecond`
	• `millisecond`
	• `second`
	• `minute`
	• `hour`
	• `day`
	• `ns`(dasselbe wie`nanosecond`)
	• `us`(genauso wie`microsecond`)
	• `ms`(genauso wie`millisecond`)
	• `s`(genauso wie`second`)
	• `m`(genauso wie`minute`)
	• `h`(genauso wie`hour`)
	• `d`(genauso wie`day`)
	
	Beispiele:
	
	`17s`
	
	`12second`
	
	`21hour`

Datentyp	Beschreibung
	`2d`
`timeseries[row(timestamp, T,...)]`	Stellt die Werte einer Messgröße dar, die über ein Zeitintervall als aus `row` Objekten `array` zusammengesetzt aufgezeichnet wurden. Jeder `row` enthält einen `timestamp` und einen oder mehrere Messwerte des Datentyps *T*wobei *T* kann einer der folgenden Werte sein: `bigint``boolean`,`double`, oder`varchar`. Die Zeilen sind in aufsteigender Reihenfolge nach sortiert. `timestamp` Der Zeitreihen-Datentyp stellt die Werte einer Kennzahl im Zeitverlauf dar.
`unknown`	Stellt Nulldaten dar.

Integrierte Zeitreihenfunktionalität

Timestream for LiveAnalytics bietet integrierte Zeitreihenfunktionen, die Zeitreihendaten als erstklassiges Konzept behandeln.

Die integrierte Zeitreihenfunktion kann in zwei Kategorien unterteilt werden: Ansichten und Funktionen.

Im Folgenden finden Sie Informationen zu den einzelnen Konstrukten.

Themen

- Ansichten von Zeitreihen
- Funktionen für Zeitreihen

Ansichten von Zeitreihen

Timestream for LiveAnalytics unterstützt die folgenden Funktionen zur Umwandlung Ihrer Daten in den `timeseries` Datentyp:

Themen

- CREATE_TIME_SERIES
- UNNEST

CREATE_TIME_SERIES

CREATE_ TIME _ SERIES ist eine Aggregationsfunktion, die alle Rohmessungen einer Zeitreihe (Zeit- und Messwerte) verwendet und einen Zeitreihendatentyp zurückgibt. Die Syntax dieser Funktion lautet wie folgt:

```
CREATE_TIME_SERIES(time, measure_value::<data_type>)
```

Dabei `<data_type>` handelt es sich um den Datentyp des Messwerts. Dabei kann es sich um Bigint, Boolean, Double oder Varchar handeln. Der zweite Parameter darf nicht Null sein.

Betrachten Sie die CPU Nutzung von EC2 Instances, die in einer Tabelle mit dem Namen Metrics gespeichert sind, wie unten dargestellt:

Zeit	Region	az	vpc	instance_id	measure_name	Messwert::doppelt
2019-12-04 19:00:00.000000000	us-east-1	us-ost-1d	vpc-1a2b3c4d	i-1234567890abcdef0	CPU-Auslastung	35,0
2019-12-04 19:00:01.000000000	us-east-1	us-ost-1d	vpc-1a2b3c4d	i-1234567890abcdef0	CPU-Auslastung	38,2
2019-12-04 19:00:02.000000000	us-east-1	us-ost-1d	vpc-1a2b3c4d	i-1234567890abcdef0	CPU-Auslastung	45,3
2019-12-04 19:00:00.000000000	us-east-1	us-ost-1d	vpc-1a2b3c4d	i-1234567890abcdef1	CPU-Auslastung	54,1

Zeit	Region	az	vpc	instance_id	measure_name	Messwert::doppelt
2019-12-04 19:00:01.000 000000	us-east-1	us-ost-1d	vpc-1a2b3c4d	i-1234567890abcdef1	CPU-Auslastung	42,5
2019-12-04 19:00:02.000 000000	us-east-1	us-ost-1d	vpc-1a2b3c4d	i-1234567890abcdef1	CPU-Auslastung	33,7

Die Abfrage ausführen:

```
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
```

gibt alle Serien zurück, die einen Messwert haben`cpu_utilization`. In diesem Fall haben wir zwei Serien:

Region	az	vpc	instance_id	cpu_utilization
us-east-1	us-ost-1d	vpc-1a2b3c4d	i-1234567890abcdef0	[{Zeit: 2019-12-04 19:00:00.000 000000, measure_value: :double: 35.0}, {time: 2019-12-04 19:00:01.000 000000, measure_value: :double: 38.2}, {time:

Region	az	vpc	instance_id	cpu_utilization
				2019-12-0 4 19:00:02. 000 000000, measure_v alue: :double: 45,3}]
us-east-1	us-east-1d	vpc-1a2b3c4d	i-1234567 890abcdef1	[{Uhrzeit: 2019-12-0 4 19:00:00. 000 000000, measure_v alue: :double: 35.1}, {time: 2019-12-0 4 19:00:01. 000 000000, measure_v alue: :double: 38,5}, {time: 2019-12-0 4 19:00:02. 000 000000, measure_v alue: :double: 45.7}]

UNNEST

UNNESTist eine Tabellenfunktion`timeseries`, mit der Sie Daten in das flache Modell transformieren können. Die Syntax ist wie folgt:

UNNEST`timeseries`wandelt a in zwei Spalten um, nämlich `time` und`value`. Sie können Aliase auch mit verwenden, UNNEST wie unten gezeigt:

```
UNNEST(timeseries) AS <alias_name> (time_alias, value_alias)
```

wo <alias_name> ist der Alias für die flache Tabelle, time_alias ist der Alias für die time Spalte und value_alias ist der Alias für die value Spalte.

Stellen Sie sich zum Beispiel das Szenario vor, in dem einige EC2 Instances in Ihrer Flotte so konfiguriert sind, dass sie Metriken in einem Intervall von 5 Sekunden ausgeben, andere wiederum Metriken in einem Intervall von 15 Sekunden ausgeben und Sie die durchschnittlichen Metriken für alle Instances mit einer Granularität von 10 Sekunden für die letzten 6 Stunden benötigen. Um diese Daten zu erhalten, transformieren Sie Ihre Metriken mithilfe von CREATE_ TIME _ SERIES in das Zeitreihenmodell. Anschließend können Sie INTERPOLATE_ verwendenLINEAR, um die fehlenden Werte mit einer Genauigkeit von 10 Sekunden abzurufen. Als Nächstes transformieren Sie die Daten wieder in das flache Modell UNNEST, indem Sie die Durchschnittsmetriken für alle Instanzen verwenden und dann verwenden, AVGum sie abzurufen.

```
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
```

Die obige Abfrage demonstriert die Verwendung von UNNESTmit einem Alias. Im Folgenden finden Sie ein Beispiel für dieselbe Abfrage ohne Verwendung eines Alias für 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
```

Funktionen für Zeitreihen

Amazon Timestream for LiveAnalytics unterstützt Zeitreihenfunktionen wie Ableitungen, Integrale und Korrelationen sowie andere, um tiefere Einblicke aus Ihren Zeitreihendaten zu gewinnen. Dieser Abschnitt enthält Nutzungsinformationen für jede dieser Funktionen sowie Beispielabfragen. Wählen Sie unten ein Thema aus, um mehr zu erfahren.

Themen

- Interpolationsfunktionen
- Abgeleitete Funktionen
- Integrale Funktionen
- Korrelationsfunktionen
- Funktionen filtern und reduzieren

Interpolationsfunktionen

Wenn in Ihren Zeitreihendaten Werte für Ereignisse zu bestimmten Zeitpunkten fehlen, können Sie die Werte dieser fehlenden Ereignisse mithilfe von Interpolation schätzen. Amazon Timestream unterstützt vier Varianten der Interpolation: lineare Interpolation, kubische Spline-Interpolation, Locf-Interpolation (Last Observation Carried Forward) und konstante Interpolation. Dieser Abschnitt enthält Informationen zur Verwendung von Timestream für Interpolationsfunktionen sowie Beispielabfragen. LiveAnalytics

Nutzungsinformationen

Funktion	Ausgabedatentyp	Beschreibung
`interpolate_linear (timeseries, array[timestamp])`	Zeitreihen	Füllt fehlende Daten mit linearer Interpolation aus.

Funktion	Ausgabedatentyp	Beschreibung
`interpolate_linear(timeseries, timestamp)`	double	Füllt fehlende Daten mit linearer Interpolation aus.
`interpolate_spline_cubic(timeseries, array[timestamp])`	Zeitreihen	Füllt fehlende Daten mithilfe der kubischen Spline-Interpolation aus.
`interpolate_spline_cubic(timeseries, timestamp)`	double	Füllt fehlende Daten mithilfe der kubischen Spline-Interpolation aus.
`interpolate_locf(timeseries, array[timestamp])`	Zeitreihen	Füllt fehlende Daten mit dem zuletzt abgetasteten Wert aus.
`interpolate_locf(timeseries, timestamp)`	double	Füllt fehlende Daten mit dem zuletzt abgetasteten Wert aus.
`interpolate_fill(timeseries, array[timestamp], double)`	Zeitreihen	Füllt fehlende Daten mit einem konstanten Wert aus.
`interpolate_fill(timeseries, timestamp, double)`	double	Füllt fehlende Daten mit einem konstanten Wert aus.

Abfragebeispiele

Example

Ermittelt die durchschnittliche CPU Auslastung in Intervallen von 30 Sekunden für einen bestimmten EC2 Host in den letzten 2 Stunden und füllt die fehlenden Werte mit linearer Interpolation aus:

```
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

Ermitteln Sie die durchschnittliche CPU Auslastung in Intervallen von 30 Sekunden für einen bestimmten EC2 Host in den letzten 2 Stunden und füllen Sie die fehlenden Werte durch Interpolation auf der Grundlage der letzten übertragenen Beobachtung aus:

```
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)
```

Abgeleitete Funktionen

Derivate werden verwendet, um die Änderungsrate für eine bestimmte Kennzahl zu berechnen und können verwendet werden, um proaktiv auf ein Ereignis zu reagieren. Nehmen wir zum Beispiel an, Sie berechnen die Ableitung der CPU Auslastung von EC2 Instances in den letzten 5 Minuten und stellen eine signifikante positive Ableitung fest. Dies kann auf eine erhöhte Auslastung Ihres Workloads hindeuten, sodass Sie möglicherweise entscheiden, mehr EC2 Instances hochzufahren, um Ihren Workload besser bewältigen zu können.

Amazon Timestream unterstützt zwei Varianten von abgeleiteten Funktionen. Dieser Abschnitt enthält Informationen zur Verwendung von Timestream für LiveAnalytics abgeleitete Funktionen sowie Beispielabfragen.

Nutzungsinformationen

Funktion	Ausgabedatentyp	Beschreibung
`derivative_linear( timeseries, interval)`	Zeitreihen	Berechnet die Ableitung jedes Punktes in der `timeseries` für den angegebenen Wert. `interval`
`non_negative_deriv ative_linear(times eries, interval)`	Zeitreihen	Wie`derivative_linear( timeseries, interval)` , gibt aber nur positive Werte zurück.

Abfragebeispiele

Example

Ermitteln Sie die Änderungsrate der CPU Auslastung in den letzten 1 Stunde alle 5 Minuten:

```
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

Berechnen Sie die Rate der Zunahme von Fehlern, die durch einen oder mehrere Microservices verursacht werden:

```
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
```

Integrale Funktionen

Sie können Integrale verwenden, um die Fläche unter der Kurve pro Zeiteinheit für Ihre Zeitreihenereignisse zu ermitteln. Nehmen wir zum Beispiel an, Sie verfolgen die Anzahl der Anfragen, die Ihre Anwendung pro Zeiteinheit erhält. In diesem Szenario können Sie die Integralfunktion verwenden, um das Gesamtvolumen der Anfragen zu ermitteln, die in einem bestimmten Intervall über einen bestimmten Zeitraum bearbeitet wurden.

Amazon Timestream unterstützt eine Variante integraler Funktionen. Dieser Abschnitt enthält Informationen zur Verwendung der Funktion Timestream for LiveAnalytics Integral sowie Beispielabfragen.

Nutzungsinformationen

Funktion	Ausgabedatentyp	Beschreibung

`integral_trapezoidal(timeseries(double))`
 	
`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))`
	double	Nähert sich dem Integral gemäß dem `interval day to second` für den angegebenen Wert angegebenen Wert unter Verwendung der `timeseries` Trapezregel an. Der Parameter Intervall von Tag zu Sekunde ist optional und die Standardeinstellung ist. `1s` Weitere Hinweise zu Intervallen finden Sie unterIntervall und Dauer.

Abfragebeispiele

Example

Berechnen Sie das Gesamtvolumen der Anfragen, die in der letzten Stunde alle fünf Minuten von einem bestimmten Host bearbeitet wurden:

```
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
```

Korrelationsfunktionen

Bei zwei Zeitreihen ähnlicher Länge liefern Korrelationsfunktionen einen Korrelationskoeffizienten, der erklärt, wie sich die beiden Zeitreihen im Zeitverlauf entwickeln. Der Korrelationskoeffizient reicht von `-1.0` bis`1.0`. `-1.0`gibt an, dass die beiden Zeitreihen mit derselben Geschwindigkeit in entgegengesetzte Richtungen tendieren. `1.0` Bedeutet hingegen, dass die beiden Zeitreihen mit derselben Geschwindigkeit in dieselbe Richtung tendieren. Der Wert von `0` bedeutet, dass keine Korrelation zwischen den beiden Zeitreihen besteht. Wenn beispielsweise der Ölpreis steigt und der Aktienkurs eines Ölunternehmens steigt, weisen der Trend des Ölpreisanstiegs und des Preisanstiegs des Ölunternehmens einen positiven Korrelationskoeffizienten auf. Ein hoher positiver Korrelationskoeffizient würde darauf hindeuten, dass sich die beiden Preise mit ähnlicher Geschwindigkeit entwickeln. In ähnlicher Weise ist der Korrelationskoeffizient zwischen Anleihekursen und Anleiherenditen negativ, was darauf hindeutet, dass diese beiden Werte im Laufe der Zeit in die entgegengesetzte Richtung tendieren.

Amazon Timestream unterstützt zwei Varianten von Korrelationsfunktionen. Dieser Abschnitt enthält Informationen zur Verwendung von Timestream für LiveAnalytics Korrelationsfunktionen sowie Beispielabfragen.

Nutzungsinformationen

Funktion	Ausgabedatentyp	Beschreibung
`correlate_pearson( timeseries, timeseries)`	double	Berechnet den Korrelati onskoeffizienten von Pearson für die beiden. `timeserie s` Die Zeitreihen müssen dieselben Zeitstempel haben.

Funktion	Ausgabedatentyp	Beschreibung
`correlate_spearman (timeseries, timeseries)`	double	Berechnet den Spearman-Korrelationskoeffizienten für die beiden. `timeseries` Die Zeitreihen müssen dieselben Zeitstempel haben.

Abfragebeispiele

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
```

Funktionen filtern und reduzieren

Amazon Timestream unterstützt Funktionen zur Durchführung von Filter- und Reduktionsoperationen mit Zeitreihendaten. Dieser Abschnitt enthält Informationen zur Verwendung von Timestream für LiveAnalytics Filter- und Reduktionsfunktionen sowie Beispielabfragen.

Nutzungsinformationen

Funktion	Ausgabedatentyp	Beschreibung
`filter(timeseries(T), function(T, Boolean))`	Zeitreihe (T)	Konstruiert eine Zeitreihe aus einer Eingabezeitreihe und verwendet dabei Werte, für die der übergebene `function` Wert zurückgegeben wird. `true`
`reduce(timeseries(T), initialState S, inputFunction(S, T, S), outputFunction(S, R))`	R	Gibt einen einzelnen Wert zurück, reduziert gegenüber der Zeitreihe. Das `inputFunction` wird der Reihe nach für jedes Element in der Zeitreihe aufgerufen. Übernimmt zusätzlich zum aktuellen Element den aktuellen Status (anfänglich `initialState`) und gibt den neuen Status zurück. inputFunction Das `outputFunction` wird aufgerufen, um den Endzustand in den Ergebniswert umzuwandeln. Das `outputFunction` kann eine Identitätsfunktion sein.

Abfragebeispiele

Example

Konstruieren Sie eine Zeitreihe der CPU Auslastung eines Hosts und filtern Sie Punkte mit einem Messwert von mehr als 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

Konstruieren Sie eine Zeitreihe der CPU Auslastung eines Hosts und bestimmen Sie die quadratische Summe der Messwerte:

```
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

Konstruieren Sie eine Zeitreihe der CPU Auslastung eines Hosts und bestimmen Sie den Anteil der Stichproben, die über dem CPU Schwellenwert liegen:

```
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
```

SQLUnterstützung

Timestream for LiveAnalytics unterstützt einige gängige SQL Konstrukte. Sie können weiter unten mehr lesen.

Themen

- SELECT
- Unterstützung für Unterabfragen
- SHOWAussagen
- DESCRIBEAussagen
- UNLOAD

SELECT

SELECTAnweisungen können verwendet werden, um Daten aus einer oder mehreren Tabellen abzurufen. Die Abfragesprache von Timestream unterstützt die folgende Syntax für SELECTAnweisungen:

```
[ 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 ] ]
```

where

- `function (expression)`ist eine der unterstützten Fensterfunktionen.
- `partition_expr_list`ist:

```
expression | column_name [, expr_list ]
```

- `order_list`ist:

```
expression | column_name [ ASC | DESC ]
[ NULLS FIRST | NULLS LAST ]
[, order_list ]
```

- `frame_clause`eist:

```
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`mist einer von:

```
table_name [ [ AS ] alias [ ( column_alias [, ...] ) ] ]
from_item join_type from_item [ ON join_condition | USING ( join_column [, ...] ) ]
```

- `join_type`eist einer von:

```
[ INNER ] JOIN
LEFT [ OUTER ] JOIN
RIGHT [ OUTER ] JOIN
```

```
FULL [ OUTER ] JOIN
```

- `grouping_element`ist einer von:

```
()
expression
```

Unterstützung für Unterabfragen

Timestream unterstützt Unterabfragen in `EXISTS` und Prädikate. `IN` Das `EXISTS` Prädikat bestimmt, ob eine Unterabfrage Zeilen zurückgibt. Das `IN` Prädikat bestimmt, ob die von der Unterabfrage erzeugten Werte mit den Werten oder dem Ausdruck einer IN-Klausel übereinstimmen. Die Timestream-Abfragesprache unterstützt korrelierte und andere Unterabfragen.

```
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

SHOWAussagen

Mithilfe des `SHOW DATABASES` Kontoauszugs können Sie alle Datenbanken in einem Konto einsehen. Die Syntax ist wie folgt:

```
SHOW DATABASES [LIKE pattern]
```

wobei die `LIKE` Klausel verwendet werden kann, um Datenbanknamen zu filtern.

Mithilfe der `SHOW TABLES` Anweisung können Sie alle Tabellen in einem Konto anzeigen. Die Syntax ist wie folgt:

```
SHOW TABLES [FROM database] [LIKE pattern]
```

wobei die `FROM` Klausel zum Filtern von Datenbanknamen und die `LIKE` Klausel zum Filtern von Tabellennamen verwendet werden kann.

Mithilfe der `SHOW MEASURES` Anweisung können Sie alle Kennzahlen für eine Tabelle anzeigen. Die Syntax ist wie folgt:

```
SHOW MEASURES FROM database.table [LIKE pattern]
```

wobei die `FROM` Klausel zur Angabe des Datenbank- und Tabellennamens verwendet wird und die `LIKE` Klausel zum Filtern von Kennzahlnamen verwendet werden kann.

DESCRIBEAussagen

Sie können die Metadaten für eine Tabelle mithilfe der `DESCRIBE` Anweisung anzeigen. Die Syntax ist wie folgt:

```
DESCRIBE database.table
```

wo `table` enthält den Tabellennamen. Die Describe-Anweisung gibt die Spaltennamen und Datentypen für die Tabelle zurück.

UNLOAD

Timestream for LiveAnalytics unterstützt einen `UNLOAD` Befehl als Erweiterung seiner SQL Unterstützung. Datentypen, die von unterstützt werden, `UNLOAD` werden unter beschriebenUnterstützte Datentypen. Die `unknown` Typen `time` und gelten nicht für`UNLOAD`.

```
UNLOAD (SELECT statement)
 TO 's3://bucket-name/folder'
 WITH ( option = expression [, ...] )
```

wo Option ist

```
{ 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>'
}
```

SELECTAussage

Die Abfrageanweisung, die verwendet wird, um Daten aus einem oder mehreren Timestreams für LiveAnalytics Tabellen auszuwählen und abzurufen.

```
(SELECT column 1, column 2, column 3 from database.table
      where measure_name = "ABC" and timestamp between ago (1d) and now() )
```

TO-Klausel

```
TO 's3://bucket-name/folder'
```

or

```
TO 's3://access-point-alias/folder'
```

Die TO Klausel in der UNLOAD Anweisung gibt das Ziel für die Ausgabe der Abfrageergebnisse an. Sie müssen den vollständigen Pfad angeben, einschließlich entweder des Amazon S3-Bucket-Namens oder Amazon S3 access-point-alias mit dem Speicherort des Ordners auf Amazon S3, in den Timestream for die LiveAnalytics Ausgabedateiobjekte schreibt. Der S3-Bucket sollte demselben Konto gehören und sich in derselben Region befinden. Zusätzlich zum Abfrageergebnissatz LiveAnalytics schreibt Timestream for die Manifest- und Metadatendateien in den angegebenen Zielordner.

PARTITIONED_BY-Klausel

```
partitioned_by = ARRAY [col_name[,…] , (default: none)
```

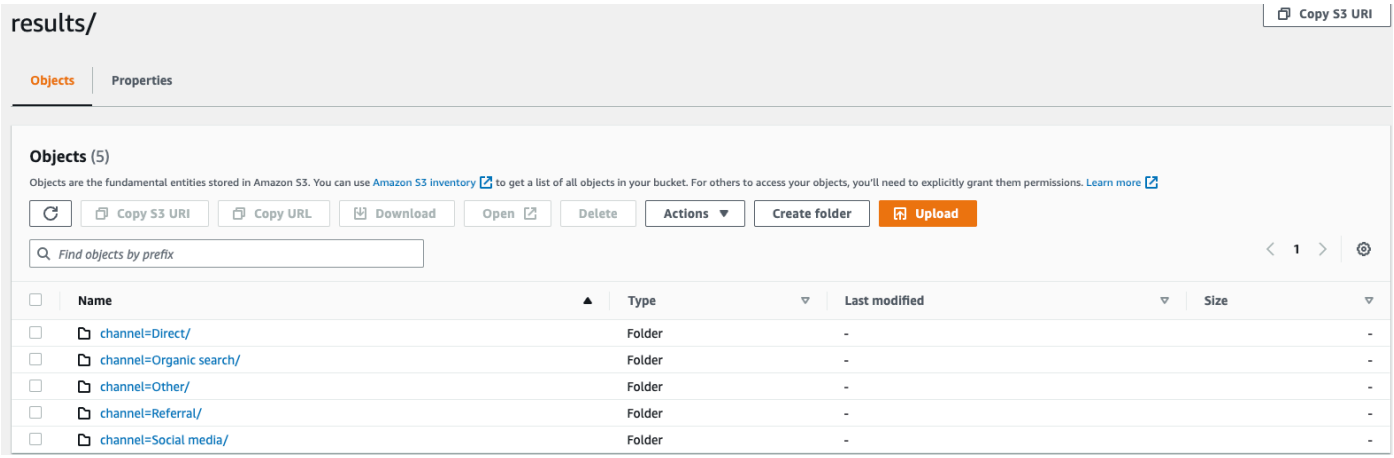
Die `partitioned_by` Klausel wird in Abfragen verwendet, um Daten auf granularer Ebene zu gruppieren und zu analysieren. Wenn Sie Ihre Abfrageergebnisse in den S3-Bucket exportieren, können Sie wählen, ob Sie die Daten anhand einer oder mehrerer Spalten in der Auswahlabfrage partitionieren möchten. Bei der Partitionierung der Daten werden die exportierten Daten basierend auf der Partitionsspalte in Teilmengen unterteilt, und jede Teilmenge wird in einem separaten Ordner gespeichert. Innerhalb des Ergebnisordners, der Ihre exportierten Daten enthält, wird automatisch ein Unterordner erstellt`folder/results/partition column = partition value/`. Beachten Sie jedoch, dass partitionierte Spalten nicht in der Ausgabedatei enthalten sind.

`partitioned_by`ist keine obligatorische Klausel in der Syntax. Wenn Sie die Daten ohne Partitionierung exportieren möchten, können Sie die Klausel in der Syntax ausschließen.

Example

Angenommen, Sie überwachen die Clickstream-Daten Ihrer Website und haben 5 Verkehrskanäle`direct`, nämlich, `Social Media` `Organic Search``Other`, und. `Referral` Beim Exportieren der Daten können Sie wählen, ob Sie die Daten mithilfe der Spalte `Channel` partitionieren möchten. In Ihrem Datenordner befinden sich fünf Ordner mit jeweils ihrem jeweiligen Kanalnamen. `s3://bucketname/results/channel=Social Media/.` In diesem Ordner finden Sie beispielsweise die Daten aller Kunden, die über den `Social Media` Kanal auf Ihre Website gelangt sind. `s3://bucketname/results` In ähnlicher Weise werden Sie andere Ordner für die verbleibenden Kanäle haben.

Exportierte Daten, partitioniert nach Kanalspalten



FORMAT

```
format = [ '{ CSV | PARQUET }' , default: CSV
```

Die Schlüsselwörter zur Angabe des Formats der Abfrageergebnisse, die in Ihren S3-Bucket geschrieben werden. Sie können die Daten entweder als kommagetrennten Wert (CSV) mit einem Komma (,) als Standardtrennzeichen oder im Apache Parquet-Format, einem effizienten offenen spaltenbasierten Speicherformat für Analysen, exportieren.

COMPRESSION

```
compression = [ '{ GZIP | NONE }' ], default: GZIP
```

Sie können die exportierten Daten mithilfe des Komprimierungsalgorithmus komprimieren GZIP oder sie dekomprimieren lassen, indem Sie die Option angeben. `NONE`

ENCRYPTION

```
encryption = [ '{ SSE_KMS | SSE_S3 }' ], default: SSE_S3
```

Die Ausgabedateien auf Amazon S3 werden mit der von Ihnen ausgewählten Verschlüsselungsoption verschlüsselt. Zusätzlich zu Ihren Daten werden auch die Manifest- und Metadatendateien auf der Grundlage der von Ihnen ausgewählten Verschlüsselungsoption verschlüsselt. Wir unterstützen derzeit die SSE _S3- und SSE KMS _-Verschlüsselung. SSE_S3 ist eine serverseitige Verschlüsselung, bei der Amazon S3 die Daten mit der 256-Bit-Verschlüsselung des Advanced Encryption Standard () verschlüsselt. AES SSE_ KMS ist eine serverseitige Verschlüsselung zur Verschlüsselung von Daten mit vom Kunden verwalteten Schlüsseln.

KMS_KEY

```
kms_key = '<string>'
```

KMSDer Schlüssel ist ein vom Kunden definierter Schlüssel zur Verschlüsselung exportierter Abfrageergebnisse. KMSDer Schlüssel wird sicher vom AWS Key Management Service (AWS KMS) verwaltet und zum Verschlüsseln von Datendateien auf Amazon S3 verwendet.

FIELD_DELIMITER

```
field_delimiter ='<character>' , default: (,)
```

Beim Exportieren der Daten im CSV Format gibt dieses Feld ein einzelnes ASCII Zeichen an, das zur Trennung von Feldern in der Ausgabedatei verwendet wird, z. B. ein senkrechter Strich (|), ein Komma (,) oder ein Tabulatorzeichen (/t). Das Standardtrennzeichen für CSV Dateien ist ein Komma. Wenn ein Wert in Ihren Daten das gewählte Trennzeichen enthält, wird das Trennzeichen mit einem Anführungszeichen in Anführungszeichen gesetzt. Wenn der Wert in Ihren Daten beispielsweise Folgendes enthält`Time,stream`, wird dieser Wert wie in den exportierten Daten `"Time,stream"` in Anführungszeichen gesetzt. Das von Timestream für verwendete Anführungszeichen LiveAnalytics sind doppelte Anführungszeichen („).

Vermeiden Sie es, das Wagenrücklaufzeichen (ASCII`13`0D, Hexadezimalzahl, Text '\ r') oder das Zeilenumbruchzeichen (ASCII10, Hexadezimalzahl 0A, Text '\n') als 'anzugeben, `FIELD_DELIMITER` wenn Sie Header in die aufnehmen möchtenCSV, da dies viele Parser daran hindert, die Header in der resultierenden Ausgabe korrekt zu analysieren. CSV

ESCAPED_VON

```
escaped_by = '<character>', default: (\)
```

Beim Exportieren der Daten im CSV Format gibt dieses Feld das Zeichen an, das in der in den S3-Bucket geschriebenen Datendatei als Escape-Zeichen behandelt werden soll. Die Flucht erfolgt in den folgenden Szenarien:

1. Wenn der Wert selbst das Anführungszeichen („) enthält, wird er mit einem Escape-Zeichen maskiert. Wenn der Wert beispielsweise lautet`Time"stream`, wobei (\) das konfigurierte Escape-Zeichen ist, dann wird er als maskiert`Time\"stream`.
2. Wenn der Wert das konfigurierte Escape-Zeichen enthält, wird es maskiert. Wenn der Wert beispielsweise ist`Time\stream`, wird er als maskiert`Time\\stream`.

> **Note**
>
> Wenn die exportierte Ausgabe komplexe Datentypen wie Arrays, Zeilen oder Zeitreihen enthält, wird sie als Zeichenfolge serialisiert. JSON Im Folgenden sehen Sie ein Beispiel.

Datentyp	Tatsächlicher Wert	Wie der Wert im CSV Format [serialisierte JSON Zeichenfolge] maskiert wird
Array	`[ 23,24,25 ]`	`"[23,24,25]"`
Zeile	`( x=23.0, y=hello )`	`"{\"x\":23.0,\"y\":\"hello\"}"`
Zeitreihen	`[ ( 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'
```

Wenn Sie die Daten im CSV Format exportieren, können Sie in diesem Feld Spaltennamen als erste Zeile der exportierten CSV Datendateien angeben.

Die akzeptierten Werte sind „wahr“ und „falsch“ und der Standardwert ist „falsch“. Optionen zur Texttransformation, wie z. B. `escaped_by` und, `field_delimiter` gelten auch für Überschriften.

> **Note**
>
> Beim Einbeziehen von Kopfzeilen ist es wichtig, dass Sie kein Wagenrücklaufzeichen (ASCII13, Hex 0D, Text '\ r') oder ein Zeilenumbruchzeichen (ASCII10, Hex 0A, Text '\n') als 'auswählen`FIELD_DELIMITER`, da dies viele Parser daran hindert, die Header in der resultierenden Ausgabe korrekt zu analysieren. CSV

MAX_FILE_SIZE

```
max_file_size = 'X[MB|GB]' , default: '78GB'
```

Dieses Feld gibt die maximale Größe der Dateien an, die die `UNLOAD` Anweisung in Amazon S3 erstellt. Die `UNLOAD` Anweisung kann mehrere Dateien erstellen, aber die maximale Größe jeder in Amazon S3 geschriebenen Datei entspricht ungefähr der in diesem Feld angegebenen Größe.

Der Wert des Felds muss zwischen 16 MB und einschließlich 78 GB liegen. Sie können ihn als Ganzzahl wie `12GB` oder in Dezimalzahlen wie `0.5GB` oder angeben. `24.7MB` Der Standardwert ist 78 GB.

Die tatsächliche Dateigröße wird ungefähr angegeben, wenn die Datei geschrieben wird, sodass die tatsächliche Maximalgröße möglicherweise nicht exakt der von Ihnen angegebenen Zahl entspricht.

Logische Operatoren

Timestream for LiveAnalytics unterstützt die folgenden logischen Operatoren.

Operator	Beschreibung	Beispiel
AND	Wahr, wenn beide Werte wahr sind	a AND b
ODER	Stimmt, wenn einer der Werte wahr ist	a ODER b
NOT	Stimmt, wenn der Wert falsch ist	NOTein

- Das Ergebnis eines AND Vergleichs kann sein, NULL ob eine oder beide Seiten des NULL Ausdrucks
- Wenn mindestens eine Seite eines AND Operators ist, wird FALSE der Ausdruck zu FALSE ausgewertet.
- Das Ergebnis eines OR Vergleichs kann seinNULL, wenn eine oder beide Seiten des NULL Ausdrucks
- Wenn mindestens eine Seite eines OR Operators ist, wird TRUE der Ausdruck zu TRUE ausgewertet.
- Die logische Ergänzung von NULL istNULL.

Die folgende Wahrheitstabelle zeigt den Umgang mit NULL in AND undOR:

A	B	A und b	A oder b
Null	Null	Null	Null
false	Null	false	Null
Null	false	false	Null
true	Null	Null	true
Null	true	Null	true
false	false	false	false
true	false	false	true
false	true	false	true
true	true	true	true

Die folgende Wahrheitstabelle zeigt den Umgang mit NULL inNOT:

A	Kein
Null	Null

A	Kein
true	false
false	true

Vergleichsoperatoren

Timestream for LiveAnalytics unterstützt die folgenden Vergleichsoperatoren.

Operator	Beschreibung
<	kleiner als
>	größer als
<=	kleiner als oder gleich
>=	größer als oder gleich
=	Gleich
<>	Ungleich
!=	Ungleich

Note

- Der `BETWEEN` Operator testet, ob ein Wert innerhalb eines angegebenen Bereichs liegt. Die Syntax ist wie folgt:

  ```
  BETWEEN min AND max
  ```

  Das Vorhandensein von `NULL` in einer `BETWEEN` `NOT BETWEEN` Oder-Anweisung führt dazu, dass die Anweisung als 0 bewertet wird`NULL`.
- `IS NULL` und `IS NOT NULL` Operatoren testen, ob ein Wert Null (undefiniert) ist. Die Verwendung von `NULL` with `IS NULL` ergibt „Wahr“.

- In SQL steht ein NULL Wert für einen unbekannten Wert.

Vergleichsfunktionen

Timestream for LiveAnalytics unterstützt die folgenden Vergleichsfunktionen.

Themen

am größten ()

Die Funktion greatest () gibt den größten der angegebenen Werte zurück. Sie gibt zurückNULL, ob einer der angegebenen Werte zutrifftNULL. Die Syntax ist wie folgt.

```
greatest(value1, value2, ..., valueN)
```

am wenigsten ()

Die Funktion least () gibt den kleinsten der angegebenen Werte zurück. Sie gibt zurück, NULL ob einer der angegebenen Werte NULL Die Syntax ist wie folgt.

```
least(value1, value2, ..., valueN)
```

ALL(), ANY () und SOME ()

Die SOME QuantifiziererALL, ANY und können zusammen mit Vergleichsoperatoren auf folgende Weise verwendet werden.

Expression	Bedeutung
A = ALL (...)	Wird als wahr ausgewertet, wenn A allen Werten entspricht.

Expression	Bedeutung
A <> ALL (...)	Wird als wahr ausgewertet, wenn A mit keinem Wert übereinstimmt.
A < ALL (...)	Wird als wahr ausgewertet, wenn A kleiner als der kleinste Wert ist.
A = ANY (...)	Wird als wahr ausgewertet, wenn A einem der Werte entspricht.
A <> ANY (...)	Wird als wahr ausgewertet, wenn A nicht mit einem oder mehreren Werten übereinstimmt.
A < ANY (...)	Wird als wahr ausgewertet, wenn A kleiner als der größte Wert ist.

Beispiele und Nutzungshinweise

> **Note**
>
> Bei der Verwendung von `ALL` `ANY` oder `VALUES` sollte das Schlüsselwort verwendet werden`SOME`, wenn es sich bei den Vergleichswerten um eine Liste von Literalen handelt.

Beispiel: **`ANY()`**

Ein Beispiel für `ANY()` in einer Abfrageanweisung wie folgt.

```
SELECT 11.7 = ANY (VALUES 12.0, 13.5, 11.7)
```

Eine alternative Syntax für dieselbe Operation lautet wie folgt.

```
SELECT 11.7 = ANY (SELECT 12.0 UNION ALL SELECT 13.5 UNION ALL SELECT 11.7)
```

In diesem Fall `ANY()` ergibt das Ergebnis. `True`

Beispiel: **`ALL()`**

Ein Beispiel für `ALL()` in einer Abfrageanweisung wie folgt.

```
SELECT 17 < ALL (VALUES 19, 20, 15);
```

Eine alternative Syntax für dieselbe Operation lautet wie folgt.

```
SELECT 17 < ALL (SELECT 19 UNION ALL SELECT 20 UNION ALL SELECT 15);
```

In diesem Fall `ALL()` ergibt das Ergebnis. `False`

Beispiel: **`SOME()`**

Ein Beispiel für `SOME()` in einer Abfrageanweisung wie folgt.

```
SELECT 50 >= SOME (VALUES 53, 77, 27);
```

Eine alternative Syntax für dieselbe Operation lautet wie folgt.

```
SELECT 50 >= SOME (SELECT 53 UNION ALL SELECT 77 UNION ALL SELECT 27);
```

In diesem Fall `SOME()` ergibt das Ergebnis. `True`

Bedingte Ausdrücke

Timestream for LiveAnalytics unterstützt die folgenden bedingten Ausdrücke.

Themen

Die CASE Aussage

Die CASEAnweisung durchsucht jeden Wertausdruck von links nach rechts, bis sie einen Wert findet, der dem Wert entsprichtexpression. Wenn eine Übereinstimmung gefunden wird, wird das

Ergebnis für den entsprechenden Wert zurückgegeben. Wenn keine Übereinstimmung gefunden wird, wird das Ergebnis der ELSE Klausel zurückgegeben, sofern es existiert; andernfalls null wird es zurückgegeben. Die Syntax ist wie folgt:

```
CASE expression
    WHEN value THEN result
    [ WHEN ... ]
    [ ELSE result ]
END
```

Timestream unterstützt auch die folgende Syntax für CASEAnweisungen. In dieser Syntax wertet das Formular „durchsucht“ jede boolesche Bedingung von links nach rechts aus, bis eine vorliegt, true und gibt das passende Ergebnis zurück. Wenn keine Bedingungen vorliegentrue, wird das Ergebnis der ELSE Klausel zurückgegeben, sofern es existiert; andernfalls null wird es zurückgegeben. Im Folgenden finden Sie die alternative Syntax:

```
CASE
    WHEN condition THEN result
    [ WHEN ... ]
    [ ELSE result ]
END
```

Die IF-Anweisung

Die IF-Anweisung bewertet eine Bedingung als wahr oder falsch und gibt den entsprechenden Wert zurück. Timestream unterstützt die folgenden zwei Syntaxdarstellungen für IF:

```
if(condition, true_value)
```

Diese Syntax wertet aus und gibt zurück, true_value ob die Bedingung erfüllt isttrue; andernfalls null wird sie zurückgegeben und true_value nicht ausgewertet.

```
if(condition, true_value, false_value)
```

Diese Syntax wertet aus und gibt zurück, true_value ob die Bedingung erfüllt isttrue, andernfalls wird ausgewertet und zurückgegeben. false_value

Beispiele

```
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	_Spalte 3
example 1	- null	example 3 true	example 4 false

Die Aussage COALESCE

COALESCEgibt den ersten Wert in einer Argumentliste zurück, der nicht Null ist. Die Syntax ist wie folgt:

```
coalesce(value1, value2[,...])
```

Die NULLIF Aussage

Die IF-Anweisung bewertet eine Bedingung als wahr oder falsch und gibt den entsprechenden Wert zurück. Timestream unterstützt die folgenden zwei Syntaxdarstellungen für IF:

NULLIFgibt null zurück, wenn `value1` gleich`value2`; andernfalls wird zurückgegeben. `value1` Die Syntax ist wie folgt:

```
nullif(value1, value2)
```

Die TRY Aussage

Die TRYFunktion wertet einen Ausdruck aus und behandelt bestimmte Arten von Fehlern, indem sie zurückgibt`null`. Die Syntax ist wie folgt:

```
try(expression)
```

Konvertierungs-Funktionen

Timestream for LiveAnalytics unterstützt die folgenden Konvertierungsfunktionen.

Themen

cast()

Die Syntax der Cast-Funktion zur expliziten Umwandlung eines Werts in einen Typ lautet wie folgt.

```
cast(value AS type)
```

try_cast ()

Timestream for unterstützt LiveAnalytics auch die Funktion try_cast, die Cast ähnelt, aber null zurückgibt, wenn die Übertragung fehlschlägt. Die Syntax ist wie folgt.

```
try_cast(value AS type)
```

Mathematische Operatoren

Timestream for LiveAnalytics unterstützt die folgenden mathematischen Operatoren.

Operator	Beschreibung
+	Addition
-	Subtraktion
*	Multiplikation
/	Division (eine Ganzzahldivision führt eine Kürzung durch)
%	Modul (Rest)

Mathematische Funktionen

Timestream for LiveAnalytics unterstützt die folgenden mathematischen Funktionen.

Funktion	Ausgabedatentyp	Beschreibung
Bauchmuskeln (x)	[dasselbe wie Eingabe]	Gibt den absoluten Wert von x zurück.
cbrt (x)	double	Gibt die Kubikwurzel von x zurück.
Decke (x) oder Decke (x)	[wie Eingabe]	Gibt x aufgerundet auf die nächste Ganzzahl zurück.
Grad (x)	double	Konvertiert den Winkel x im Bogenmaß in Grad.
e ()	double	Gibt die Konstante Eulersche Zahl zurück.
exp (x)	double	Gibt die Eulersche Zahl potenziert mit x zurück.
Stockwerk (x)	[dasselbe wie Eingabe]	Gibt x abgerundet auf die nächste Ganzzahl zurück.
from_base (Zeichenfolge, Radix)	bigint	Gibt den Wert einer Zeichenfolge zurück, die als Basiszahl interpretiert wird.
ln (x)	double	Gibt den natürlichen Logarithmus von x zurück.
log2 (x)	double	Gibt den Logarithmus zur Basis 2 von x zurück.
log10 (x)	double	Gibt den Logarithmus zur Basis 10 von x zurück.
mod (n, m)	[dasselbe wie Eingabe]	Gibt den Modul (Rest) von n geteilt durch m zurück.

Funktion	Ausgabedatentyp	Beschreibung
pi ()	double	Gibt die Konstante Pi zurück.
pow (x, p) oder power (x, p)	double	Gibt x potenziert mit p zurück.
Bogenmaß (x)	double	Konvertiert den Winkel x in Grad in Radiant.
rand () oder random ()	double	Gibt einen pseudozufälligen Wert im Bereich 0,0 — 1,0 zurück.
zufällig (n)	[dasselbe wie Eingabe]	Gibt eine Pseudozufallszahl zwischen 0 und n (ausschli eßlich) zurück.
rund (x)	[dasselbe wie Eingabe]	Gibt x auf die nächste Ganzzahl gerundet zurück.
rund (x, d)	[dasselbe wie Eingabe]	Gibt x auf d Dezimalstellen gerundet zurück.

Funktion	Ausgabedatentyp	Beschreibung
Zeichen (x)	[dasselbe wie Eingabe]	Gibt die Signumfunktion von x zurück, das heißt: • 0, wenn das Argument 0 ist • 1, wenn das Argument größer als 0 ist • -1, wenn das Argument kleiner als 0 ist. Bei doppelten Argumenten gibt die Funktion zusätzlich Folgendes zurück: • NaN, wenn das Argument NaN ist • 1, wenn das Argument +Infinity ist • -1, wenn das Argument -Infinity ist.
sqrt (x)	double	Gibt die Quadratwurzel von x zurück.
to_base (x, radix)	varchar	Gibt die Basisradix-Darstellung von x zurück.
kürzen (x)	double	Gibt x auf eine Ganzzahl gerundet zurück, indem Ziffern nach dem Dezimalpunkt weggelassen werden.
acos (x)	double	Gibt den Arkuskosinus von x zurück.

Funktion	Ausgabedatentyp	Beschreibung
asin (x)	double	Gibt den Arkussinus von x zurück.
atan (x)	double	Gibt den Arkustangens von x zurück.
atan2 (y, x)	double	Gibt den Bogentangens von y/x zurück.
cos (x)	double	Gibt den Kosinus von x zurück.
cosh (x)	double	Gibt den hyperbolischen Kosinus von x zurück.
sin (x)	double	Gibt den Sinus von x zurück.
tan (x)	double	Gibt den Tangens von x zurück.
tanh (x)	double	Gibt den hyperbolischen Tangens von x zurück.
unendlich ()	double	Gibt die Konstante zurück, die positive Unendlichkeit darstellt.
is_endlich (x)	boolesch	Ermitteln Sie, ob x endlich ist.
ist_unendlich (x)	boolesch	Ermitteln Sie, ob x unendlich ist.
is_nan (x)	boolesch	Ermitteln Sie, ob x ist. not-a-number
Mann ()	double	Gibt die Konstante zurück, die repräsentiert not-a-number.

Zeichenfolgenoperatoren

Timestream for LiveAnalytics unterstützt den | | Operator zum Verketten einer oder mehrerer Zeichenketten.

Zeichenfolgenfunktionen

> **Note**
>
> Es wird davon ausgegangen, dass der Eingabedatentyp dieser Funktionen varchar ist, sofern nicht anders angegeben.

Funktion	Ausgabedatentyp	Beschreibung
chr (n)	varchar	Gibt den Unicode-Codepunkt n als Varchar zurück.
Codepunkt (x)	Ganzzahl	Gibt den Unicode-Codepunkt des einzigen Zeichens von str zurück.
concat (x1,..., xN)	varchar	Gibt die Verkettung von x1, x2,..., xN zurück.
hamming_distance (x1, x2)	bigint	Gibt die Hamming-Distanz von x1 und x2 zurück, d. h. die Anzahl der Positionen, an denen sich die entsprech enden Zeichen unterscheiden. Beachten Sie, dass die beiden Varchar-Eingaben dieselbe Länge haben müssen.
Länge (x)	bigint	Gibt die Länge von x in Zeichen zurück.
levenshtein_distance (x1, x2)	bigint	Gibt die Levenshtein-Bearbe itungsdistanz von x1 und x2

Funktion	Ausgabedatentyp	Beschreibung
		zurück, d. h. die Mindestan zahl von Änderungen einzelner Zeichen (Einfügun gen, Löschungen oder Ersetzungen), die erforderlich sind, um x1 in x2 zu ändern.
niedriger (x)	varchar	Konvertiert x in Kleinbuch staben.
load (x1, Bigint-Größe, x2)	varchar	Fügt x1 nach links ein, um die Größe von Zeichen mit x2 zu ändern. Wenn die Größe kleiner als die Länge von x1 ist, wird das Ergebnis auf die Größe der Zeichen gekürzt. Größe darf nicht negativ sein und x2 darf nicht leer sein.
ltrim (x)	varchar	Entfernt führende Leerzeichen aus x.
ersetzt (x1, x2)	varchar	Entfernt alle Instanzen von x2 aus x1.
ersetzt (x1, x2, x3)	varchar	Ersetzt alle Instanzen von x2 durch x3 in x1.
Rückwärts (x)	varchar	Gibt x mit den Zeichen in umgekehrter Reihenfolge zurück.

Funktion	Ausgabedatentyp	Beschreibung
road (x1, Bigint-Größe, x2)	varchar	Drücken Sie x1 nach rechts, um die Größe von Zeichen mit x2 zu ändern. Wenn die Größe kleiner als die Länge von x1 ist, wird das Ergebnis auf die Größe der Zeichen gekürzt. Größe darf nicht negativ sein und x2 darf nicht leer sein.
rtrim (x)	varchar	Entfernt abschließende Leerzeichen aus x.
geteilt (x1, x2)	array(varchar)	Teilt x1 am Trennzeichen x2 und gibt ein Array zurück.
Split (x1, x2, Bigint-Limit)	array(varchar)	Teilt x1 auf das Trennzeichen x2 und gibt ein Array zurück. Das letzte Element im Array enthält immer alles, was im Bereich x1 noch übrig ist. Der Grenzwert muss eine positive Zahl sein.
split_part (x1, x2, bigint pos)	varchar	Teilt x1 am Trennzeichen x2 auf und gibt das Varchar-F eld an Pos zurück. Feldindiz es beginnen mit 1. Wenn pos größer als die Anzahl der Felder ist, wird Null zurückgeg eben.
strpos (x1, x2)	bigint	Gibt die Startposition der ersten Instanz von x2 in x1 zurück. Positionen beginnen mit 1. Wenn nicht gefunden, wird 0 zurückgegeben.

Funktion	Ausgabedatentyp	Beschreibung
strpos (x1, x2, Bigint-Instanz)	bigint	Gibt die Position der N-ten Instanz von x2 in x1 zurück. Die Instanz muss eine positive Zahl sein. Positionen beginnen mit 1. Wenn nicht gefunden, wird 0 zurückgegeben.
strrpos (x1, x2)	bigint	Gibt die Startposition der letzten Instanz von x2 in x1 zurück. Positionen beginnen mit 1. Wenn nicht gefunden, wird 0 zurückgegeben.
strrpos (x1, x2, Bigint-Instanz)	bigint	Gibt die Position der N-ten Instanz von x2 in x1 zurück, beginnend am Ende von x1. Instanz muss eine positive Zahl sein. Positionen beginnen mit 1. Wenn nicht gefunden, wird 0 zurückgegeben.
Position (x2 IN x1)	bigint	Gibt die Startposition der ersten Instanz von x2 in x1 zurück. Positionen beginnen mit 1. Wenn nicht gefunden, wird 0 zurückgegeben.
substr (x, Bigint, Start)	varchar	Gibt den Rest von x von der Startposition start zurück. Positionen beginnen mit 1. Eine negative Startposition wird als relativ zum Ende von x interpretiert.

Funktion	Ausgabedatentyp	Beschreibung
substr (x, bigint start, bigint len)	varchar	Gibt eine Teilzeichenfolge von x mit der Länge len von der Startposition start zurück. Positionen beginnen mit 1. Eine negative Startposition wird als relativ zum Ende von x interpretiert.
trimmen (x)	varchar	Entfernt führende und nachfolgende Leerzeichen aus x.
oberes (x)	varchar	Konvertiert x in Großbuchs taben.

Array-Operatoren

Timestream for LiveAnalytics unterstützt die folgenden Array-Operatoren.

Operator	Beschreibung
[]	Greifen Sie auf ein Element eines Arrays zu, bei dem der erste Index bei 1 beginnt.
||	Verketten Sie ein Array mit einem anderen Array oder Element desselben Typs.

Array-Funktionen

Timestream for LiveAnalytics unterstützt die folgenden Array-Funktionen.

Funktion	Ausgabedatentyp	Beschreibung
array_distinct (x)	Array	Entferne doppelte Werte aus dem Array x.

Funktion	Ausgabedatentyp	Beschreibung
		`SELECT array_distinct(ARRAY[1,2,2,3])` Beispielergebnis: `[ 1,2,3 ]`
array_intersect (x, y)	Array	Gibt ein Array der Elemente im Schnittpunkt von x und y zurück, ohne Duplikate. `SELECT array_intersect(ARRAY[1,2,3], ARRAY[3,4,5])` Beispielergebnis: `[ 3 ]`
array_union (x, y)	Array	Gibt ein Array der Elemente in der Vereinigung von x und y zurück, ohne Duplikate. `SELECT array_union(ARRAY[1,2,3], ARRAY[3,4,5])` Beispielergebnis: `[ 1,2,3,4,5 ]`
array_except (x, y)	Array	Gibt ein Array von Elementen in x, aber nicht in y, ohne Duplikate zurück. `SELECT array_except(ARRAY[1,2,3], ARRAY[3,4,5])` Beispielergebnis: `[ 1,2 ]`

Funktion	Ausgabedatentyp	Beschreibung
array_join (x, delimiter, null_replacement)	varchar	Verkettet die Elemente des angegebenen Arrays mithilfe des Trennzeichens und einer optionalen Zeichenfolge, um Nullen zu ersetzen. `SELECT array_joi n(ARRAY[1,2,3], ';', '')` Beispielergebnis: `1;2;3`
array_max (x)	dasselbe wie Array-Elemente	Gibt den Maximalwert des Eingabe-Arrays zurück. `SELECT array_max (ARRAY[1,2,3])` Beispielergebnis: `3`
array_min (x)	dasselbe wie Array-Elemente	Gibt den Mindestwert des Eingabe-Arrays zurück. `SELECT array_min (ARRAY[1,2,3])` Beispielergebnis: 1

Funktion	Ausgabedatentyp	Beschreibung
array_position (x, Element)	bigint	Gibt die Position des ersten Vorkommens des Elements in Array x zurück (oder 0, falls es nicht gefunden wurde). `SELECT array_position(ARRAY[3,4,5,9], 5)` Beispielergebnis: 3
array_remove (x, Element)	Array	Entferne alle Elemente, die dem Element entsprechen, aus dem Array x. `SELECT array_remove(ARRAY[3,4,5,9], 4)` Beispielergebnis: `[ 3,5,9 ]`
array_sort (x)	Array	Sortiert das Array x und gibt es zurück. Die Elemente von x müssen sortierbar sein. Null-Elemente werden am Ende des zurückgegebenen Arrays platziert. `SELECT array_sort(ARRAY[6,8,2,9,3])` Beispielergebnis: `[ 2,3,6,8,9 ]`

Funktion	Ausgabedatentyp	Beschreibung
arrays_overlap (x, y)	boolesch	Testet, ob die Arrays x und y irgendwelche Elemente gemeinsam haben, die nicht Null sind. Gibt Null zurück, wenn es keine gemeinsamen Elemente gibt, die nicht Null sind, aber eines der Arrays Null enthält. `SELECT arrays_overlap(ARRAY[6,8,2,9,3], ARRAY[6,8])` Beispielergebnis: `true`
Kardinalität (x)	bigint	Gibt die Größe des Arrays x zurück. `SELECT cardinality(ARRAY[6,8,2,9,3])` Beispielergebnis: 5
concat (Matrix1, Matrix2,..., MatrixN)	Array	Verkettet die Arrays array1, array2,..., arrayN. `SELECT concat(ARRAY[6,8,2,9,3], ARRAY[11,32], ARRAY[6,8,2,0,14])` Beispielergebnis: `[ 6,8,2,9,3,11,32,6,8,2,0,14 ]`

Funktion	Ausgabedatentyp	Beschreibung
element_at (Array (E), Index)	E	Gibt das Element eines Arrays am angegebenen Index zurück. Wenn der Index < 0 ist, greift element_at auf Elemente vom letzten bis zum ersten zu. `SELECT element_at(ARRAY[6,8,2,9,3], 1)` Beispielergebnis: 6
wiederholen (Element, Anzahl)	Array	Wiederhole das Element, um es mal zu zählen. `SELECT repeat(1, 3)` Beispielergebnis: `[ 1,1,1 ]`
rückwärts (x)	Array	Gibt ein Array zurück, das die umgekehrte Reihenfolge von Array x hat. `SELECT reverse(ARRAY[6,8,2,9,3])` Beispielergebnis: `[ 3,9,2,8,6 ]`

Funktion	Ausgabedatentyp	Beschreibung
Reihenfolge (Start, Stopp)	Array (großer Ganzzahl)	Generiert eine Folge von ganzen Zahlen von Anfang bis Ende, wobei sie um 1 erhöht wird, wenn Start kleiner oder gleich Stopp ist, andernfalls -1. `SELECT sequence(3, 8)` Beispielergebnis: `[ 3,4,5,6,7,8 ]`
Reihenfolge (Start, Stopp, Schritt)	Array (großer Ganzzahl)	Generiert eine Folge von ganzen Zahlen von Anfang bis Ende, die schrittweise inkrementiert wird. `SELECT sequence(3, 15, 2)` Beispiel für ein Ergebnis: `[ 3,5,7,9,11,13,15 ]`

Funktion	Ausgabedatentyp	Beschreibung

Reihenfolge (Start, Stopp)	Array (Zeitstempel)	Generiert eine Sequenz von Zeitstempeln vom Startdatum bis zum Enddatum, wobei die Reihenfolge um 1 Tag erhöht wird.
		`SELECT sequence( '2023-04-02 19:26:12.941000000', '2023-04-06 19:26:12.941000000', 1d)`
		Beispielergebnis:
		`[ 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 ]`

Funktion	Ausgabedatentyp	Beschreibung

Reihenfolge (Start, Stopp, Schritt)	Array (Zeitstempel)	Generiert eine Sequenz von Zeitstempeln vom Start bis zum Ende, die schrittweise inkrementiert werden. Der Datentyp des Schritts ist Intervall.
		`SELECT sequence( '2023-04-02 19:26:12.941000000', '2023-04-10 19:26:12.941000000', 2d)`
		Beispielergebnis:
		`[ 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 ]`

mischen (x)	Array	Generiert eine zufällige Permutation des angegebenen Arrays x.
		`SELECT shuffle(ARRAY[6,8,2,9,3])`
		Beispielergebnis:
		`[ 6,3,2,9,8 ]`

Funktion	Ausgabedatentyp	Beschreibung
Schnitt (x, Start, Länge)	Array	Unterteilt das Array x, beginnend am Indexstart (oder am Ende, falls Start negativ ist) mit einer Länge von length. `SELECT slice(ARRAY[6,8,2,9,3], 1, 3)` Beispielergebnis: `[ 6,8,2 ]`
zip (Matrix1, Matrix2 [,...])	array (Zeile)	Fügt die angegebenen Arrays elementweise zu einem einzigen Array von Zeilen zusammen. Wenn die Argumente eine ungleichmäßige Länge haben, werden fehlende Werte mit aufgefüllt. NULL `SELECT zip(ARRAY[6,8,2,9,3], ARRAY[15, 24])` Beispielergebnis: `[ ( 6, 15 ),( 8, 24 ),( 2, - ),( 9, - ),( 3, - ) ]`

Bitweise-Funktionen

Timestream for LiveAnalytics unterstützt die folgenden bitweisen Funktionen.

Funktion	Ausgabedatentyp	Beschreibung

bit_count (Bigint, Bigint)	bigint (Zweierkomplement)	Gibt die Anzahl der Bits im ersten Bigint-Parameter zurück, wobei der zweite Parameter eine Ganzzahl mit Bit-Vorzeichen wie 8 oder 64 ist.
		`SELECT bit_count(19, 8)`
		Beispielergebnis: `3`
		`SELECT bit_count(19, 2)`
		Beispielergebnis: `Number must be represent able with the bits specified. 19 can not be represented with 2 bits`

bitwise_and (bigint, bigint)	bigint (Zweierkomplement)	Gibt den bitweisen Wert AND der Bigint-Parameter zurück. `SELECT bitwise_and(12, 7)` Beispielergebnis: `4`
bitwise_not (bigint)	bigint (Zweierkomplement)	Gibt den bitweisen Wert NOT des Bigint-Parameters zurück. `SELECT bitwise_not(12)` Beispielergebnis: `-13`

Funktion	Ausgabedatentyp	Beschreibung
bitwise_or (bigint, bigint)	bigint (Zweierkomplement)	Gibt das bitweise ODER der Bigint-Parameter zurück. `SELECT bitwise_or(12, 7)` Beispielergebnis: 15
bitwise_xor (bigint, bigint)	bigint (Zweierkomplement)	Gibt den bitweisen Wert XOR der Bigint-Parameter zurück. `SELECT bitwise_xor(12, 7)` Beispielergebnis: 11

Funktionen für reguläre Ausdrücke

Die Funktionen für reguläre Ausdrücke in Timestream for LiveAnalytics unterstützen die Java-Muster-Syntax. Timestream for LiveAnalytics unterstützt die folgenden Funktionen für reguläre Ausdrücke.

Funktion	Ausgabedatentyp	Beschreibung
regexp_extract_all (Zeichenfolge, Muster)	array(varchar)	Gibt die Teilzeichenfolge (n) zurück, denen das reguläre Ausdrucksmuster in der Zeichenfolge entspricht. `SELECT regexp_extract_all('example expect complex', 'ex\w')` Beispielergebnis: `[ exa,exp ]`

Funktion	Ausgabedatentyp	Beschreibung
regexp_extract_all (Zeichenfolge, Muster, Gruppe)	array(varchar)	Findet alle Vorkommen des Musters mit regulären Ausdrücken in einer Zeichenfolge und gibt die Gruppe mit der aufnehmenden Gruppennummer zurück. `SELECT regexp_extract_all('example expect complex', '(ex)(\w)', 2)` Beispielergebnis: `[ a,p ]`
regexp_extract (Zeichenfolge, Muster)	varchar	Gibt die erste Teilzeichenfolge zurück, der dem regulären Ausdrucksmuster in der Zeichenfolge entspricht. `SELECT regexp_extract('example expect', 'ex\w')` Beispielergebnis: `exa`

Funktion	Ausgabedatentyp	Beschreibung
regexp_extract (Zeichenfolge, Muster, Gruppe)	varchar	Findet das erste Vorkommen des Musters mit regulären Ausdrücken in einer Zeichenfolge und gibt die Gruppe mit der aufnehmenden Gruppennummer zurück. `SELECT regexp_extract('example expect', '(ex)(\w)', 2)` Beispielergebnis: a

Funktion	Ausgabedatentyp	Beschreibung
regexp_like (Zeichenfolge, Muster)	boolesch	Wertet das Muster eines regulären Ausdrucks aus und bestimmt, ob es in einer Zeichenfolge enthalten ist. Diese Funktion ähnelt dem LIKE Operator, außer dass das Muster nur in einer Zeichenfolge enthalten sein muss und nicht der gesamten Zeichenfolge entsprechen muss. Mit anderen Worten, sie führt eher eine Contains-Operation als eine Match-Operation aus. Sie können die gesamte Zeichenfolge zuordnen, indem Sie das Muster mit ^ und $ verankern. `SELECT regexp_like('example', 'ex')` Beispielergebnis: `true`
regexp_replace (Zeichenfolge, Muster)	varchar	Entfernt jede Instanz der Teilzeichenfolge, der dem Muster des regulären Ausdrucks entspricht, aus der Zeichenfolge. `SELECT regexp_replace('example expect', 'expect')` Beispielergebnis: `example`

Funktion	Ausgabedatentyp	Beschreibung
regexp_replace (Zeichenfolge, Muster, Ersatz)	varchar	Ersetzt jede Instanz der Teilzeichenfolge, der dem Regex-Muster in der Zeichenfolge entspricht, durch Ersetzung. Erfassende Gruppen können stattdessen referenziert werden, indem $g für eine nummerierte Gruppe oder $ {name} für eine benannte Gruppe verwendet wird. Ein Dollarzeichen ($) kann in die Ersetzung aufgenommen werden, indem es mit einem umgekehrten Schrägstrich (\ $) maskiert wird. `SELECT regexp_replace('example expect', 'expect', 'surprise')` Beispielergebnis: `example surprise`

Funktion	Ausgabedatentyp	Beschreibung
regexp_replace (Zeichenfolge, Muster, Funktion)	varchar	Ersetzt mithilfe der Funktion jede Instanz der Teilzeich enfolge, auf die das reguläre Ausdrucksmuster in der Zeichenfolge zutrifft. Die Lambda-Ausdrucksfunktion wird für jede Übereinst immung aufgerufen, wobei die Erfassungsgruppen als Array übergeben werden. Die Erfassung von Gruppennu mmern beginnt bei eins; es gibt keine Gruppe für den gesamten Treffer (falls Sie diese benötigen, setzen Sie den gesamten Ausdruck in Klammern). `SELECT regexp_re place('example', '(\w)', x -> upper(x[1 ]))` Beispiel für ein Ergebnis: `EXAMPLE`

Funktion	Ausgabedatentyp	Beschreibung
regexp_split (Zeichenfolge, Muster)	array(varchar)	Teilt eine Zeichenfolge anhand des Musters für reguläre Ausdrücke auf und gibt ein Array zurück. Am Ende stehende leere Zeichenketten werden beibehalten. `SELECT regexp_split('example', 'x')` Beispielergebnis: `[ e,ample ]`

Operatoren für Datum und Uhrzeit

> **Note**
>
> Timestream for unterstützt LiveAnalytics keine negativen Zeitwerte. Jede Operation, die zu einer negativen Zeit führt, führt zu einem Fehler.

Timestream for LiveAnalytics unterstützt die folgenden Operationen für `timestampsdates`, und`intervals`.

Operator	Beschreibung
+	Addition
-	Subtraktion

Themen

- Operationen
- Addition

- Subtraktion

Operationen

Der Ergebnistyp einer Operation basiert auf den Operanden. Intervallliterale wie `1day` und `3s` können verwendet werden.

```
SELECT date '2022-05-21' + interval '2' day
```

```
SELECT date '2022-05-21' + 2d
```

```
SELECT date '2022-05-21' + 2day
```

Beispielergebnis für jedes: `2022-05-23`

Zu den Intervalleinheiten gehören `second` `minutehour`,`day`,`week`,,`month`, und`year`. In einigen Fällen sind jedoch nicht alle anwendbar. Zum Beispiel können Sekunden, Minuten und Stunden nicht zu einem Datum hinzugefügt oder von diesem subtrahiert werden.

```
SELECT interval '4' year + interval '2' month
```

Beispiel für ein Ergebnis: `4-2`

```
SELECT typeof(interval '4' year + interval '2' month)
```

Beispielergebnis: `interval year to month`

Der Ergebnistyp von Intervalloperationen kann von den Operanden `'interval day to second'` abhängen `'interval year to month'` oder sein. Intervalle können zu und addiert oder von diesen subtrahiert werden. `dates` `timestamps` Ein `date` oder `timestamp` kann jedoch nicht zu einem Oder addiert oder von diesem subtrahiert werden. `date` `timestamp` Informationen zu Intervallen oder Dauern im Zusammenhang mit Datumsangaben oder Zeitstempeln finden Sie unter `date_diff` und verwandte Funktionen unter. Intervall und Dauer

Addition

Example

```
SELECT date '2022-05-21' + interval '2' day
```

Beispiel für ein Ergebnis: `2022-05-23`

Example

```
SELECT typeof(date '2022-05-21' + interval '2' day)
```

Beispielergebnis: `date`

Example

```
SELECT interval '2' year + interval '4' month
```

Beispielergebnis: `2-4`

Example

```
SELECT typeof(interval '2' year + interval '4' month)
```

Beispielergebnis: `interval year to month`

Subtraktion

Example

```
SELECT timestamp '2022-06-17 01:00' - interval '7' hour
```

Beispielergebnis: `2022-06-16 18:00:00.000000000`

Example

```
SELECT typeof(timestamp '2022-06-17 01:00' - interval '7' hour)
```

Beispielergebnis: `timestamp`

Example

```
SELECT interval '6' day - interval '4' hour
```

Beispielergebnis: `5 20:00:00.000000000`

Example

```
SELECT typeof(interval '6' day - interval '4' hour)
```

Beispielergebnis: `interval day to second`

Funktionen für Datum und Uhrzeit

> **Note**
>
> Timestream for unterstützt LiveAnalytics keine negativen Zeitwerte. Jede Operation, die zu einer negativen Zeit führt, führt zu einem Fehler.

Timestream for LiveAnalytics verwendet die UTC Zeitzone für Datum und Uhrzeit. Timestream unterstützt die folgenden Funktionen für Datum und Uhrzeit.

Themen

Allgemeines und Konvertierung

Timestream for LiveAnalytics unterstützt die folgenden allgemeinen Funktionen und Konvertierungsfunktionen für Datum und Uhrzeit.

Funktion	Ausgabedatentyp	Beschreibung
aktuelles_Datum	date	Gibt das aktuelle Datum in zurück. UTC Es werden keine Klammern verwendet. `SELECT current_date` Beispielergebnis: `2022-07-07` **Note** Dies ist auch ein reserviertes Schlüsselwort. Eine Liste der reservierten Schlüsselwörter finden Sie unterReservierte Schlüsselwörter.
current_time	time	Gibt die aktuelle Uhrzeit in zurück. UTC Es werden keine Klammern verwendet. `SELECT current_time` Beispielergebnis: `17:41:52.827000000` **Note** Dies ist auch ein reserviertes Schlüsselwort. Eine Liste der reservierten Schlüsselwörter finden Sie

Funktion	Ausgabedatentyp	Beschreibung
		unterReservierte Schlüsselwörter.
current_timestamp oder now ()	Zeitstempel	Gibt den aktuellen Zeitstempel in zurück. UTC `SELECT current_timestamp` Beispielergebnis: `2022-07-07 17:42:32.939000000` **Note** Dies ist auch ein reserviertes Schlüsselwort. Eine Liste der reservierten Schlüsselwörter finden Sie unterReservierte Schlüsselwörter.
current_timezone ()	varchar Der Wert wird '.' sein UTC	Timestream verwendet die UTC Zeitzone für Datum und Uhrzeit. `SELECT current_timezone()` Beispielergebnis: `UTC`

Funktion	Ausgabedatentyp	Beschreibung
Datum (varchar (x)), Datum (Zeitstempel)	date	`SELECT date(TIMESTAMP '2022-07-07 17:44:43.771000000')` Beispielergebnis: `2022-07-07`
last_day_of_month (Zeitstempel), last_day_of_month (Datum)	date	`SELECT last_day_of_month(TIMESTAMP '2022-07-07 17:44:43.771000000')` Beispiel für ein Ergebnis: `2022-07-31`
from_iso8601_timestamp (Zeichenfolge)	Zeitstempel	Analysiert den 8601-Zeitstempel in ein internes Zeitstempelformat. ISO `SELECT from_iso8601_timestamp('2022-06-17T08:04:05.000000000+05:00')` Beispielergebnis: `2022-06-17 03:04:05.000000000`

Funktion	Ausgabedatentyp	Beschreibung
from_iso8601_date (Zeichenfolge)	date	Analysiert die ISO 8601-Datumszeichenfolge in ein internes Zeitstempelformat für 00:00:00 des angegebenen Datums. UTC `SELECT from_iso8601_date('2022-07-17')` Beispielergebnis: `2022-07-17`
to_iso8601 (Zeitstempel), to_iso8601 (Datum)	varchar	Gibt eine Zeichenfolge im 8601-Format für die Eingabe zurück. ISO `SELECT to_iso8601(from_iso8601_date('2022-06-17'))` Beispielergebnis: `2022-06-17`
from_milliseconds (bigint)	Zeitstempel	`SELECT from_milliseconds(1)` Beispielergebnis: `1970-01-01 00:00:00.001000000`

Funktion	Ausgabedatentyp	Beschreibung
from_nanoseconds (bigint)	Zeitstempel	`select from_nanoseconds(300000001)` Beispielergebnis: `1970-01-01 00:00:00.300000001`
from_unixtime (doppelt)	Zeitstempel	Gibt einen Zeitstempel zurück, der der angegebenen Unixtime entspricht. `SELECT from_unixtime(1)` Beispielergebnis: `1970-01-01 00:00:01.000000000`

Funktion	Ausgabedatentyp	Beschreibung
Ortszeit	time	Gibt die aktuelle Uhrzeit in UTC zurück. Es werden keine Klammern verwendet. `SELECT localtime` Beispielergebnis: `17:58:22.654000000` **Note** Dies ist auch ein reserviertes Schlüsselwort. Eine Liste der reservierten Schlüsselwörter finden Sie unterReservierte Schlüsselwörter.

Funktion	Ausgabedatentyp	Beschreibung
lokaler Zeitstempel	Zeitstempel	Gibt den aktuellen Zeitstempel in zurück. UTC Es werden keine Klammern verwendet. `SELECT localtimestamp` Beispielergebnis: `2022-07-07 17:59:04.368000000` **Note** Dies ist auch ein reserviertes Schlüsselwort. Eine Liste der reservierten Schlüsselwörter finden Sie unterReservierte Schlüsselwörter.

to_milliseconds (Intervall von Tag bis Sekunde), to_milliseconds (Zeitstempel)	bigint	`SELECT to_milliseconds(INTERVAL '2' DAY + INTERVAL '3' HOUR)`
		Beispielergebnis: `183600000`
		`SELECT to_milliseconds(TIMESTAMP '2022-06-17 17:44:43.771000000')`
		Beispielergebnis: `1655487883771`

Funktion	Ausgabedatentyp	Beschreibung

to_nanoseconds (Intervall von Tag zu Sekunde), to_nanoseconds (Zeitstempel)	bigint	`SELECT to_nanoseconds(INTERVAL '2' DAY + INTERVAL '3' HOUR)`
		
		Beispielergebnis: `183600000000000`
		
		`SELECT to_nanoseconds(TIMESTAMP '2022-06-17 17:44:43.771000678')`
		
		Beispielergebnis: `1655487883771000678`

to_unixtime (Zeitstempel)	double	Gibt Unixtime für den angegebenen Zeitstempel zurück. `SELECT to_unixtime('2022-06-17 17:44:43.771000000')` Beispielergebnis: `1.655487883771E9`

Funktion	Ausgabedatentyp	Beschreibung
date_trunc (Einheit, Zeitstempel)	Zeitstempel	Gibt den auf eine Einheit gekürzten Zeitstempel zurück, wobei Einheit eins von [Sekunde, Minute, Stunde, Tag, Woche, Monat, Quartal oder Jahr] ist. `SELECT date_trunc('minute', TIMESTAMP '2022-06-17 17:44:43.771000000')` Beispiel für ein Ergebnis: `2022-06-17 17:44:00.000000000`

Intervall und Dauer

Timestream for LiveAnalytics unterstützt die folgenden Intervall- und Dauerfunktionen für Datum und Uhrzeit.

Funktion	Ausgabedatentyp	Beschreibung
date_add (Einheit, Bigint, Datum), date_add (Einheit, Bigint, Zeit), date_add (varchar (x), bigint, timestamp)	Zeitstempel	Fügt einen Bigint von Einheiten hinzu, wobei Einheit eine von [Sekunde, Minute, Stunde, Tag, Woche, Monat, Quartal oder Jahr] ist. `SELECT date_add('hour', 9, TIMESTAMP '2022-06-17 00:00:00')`

Funktion	Ausgabedatentyp	Beschreibung
		Beispielergebnis: `2022-06-17 09:00:00.000000000`
date_diff (Einheit, Datum, Datum), date_diff (Einheit, Zeit, Uhrzeit), date_diff (Einheit, Zeitstempel, Zeitstempel)	bigint	Gibt einen Unterschied zurück, wobei die Einheit eins von [Sekunde, Minute, Stunde, Tag, Woche, Monat, Quartal oder Jahr] ist. `SELECT date_diff('day', DATE '2020-03-01', DATE '2020-03-02')` Beispiel für ein Ergebnis: 1

parse_duration (Zeichenfolge)	Intervall	Analysiert die Eingabeze ichenfolge, um ein Äquivalent zurückzugeben. `interval`
		`SELECT parse_duration('42.8ms')`
		Beispielergebnis: `0 00:00:00.042800000`
		`SELECT typeof(parse_duration('42.8ms'))`
		Beispielergebnis: `interval day to second`

Funktion	Ausgabedatentyp	Beschreibung
bin (Zeitstempel, Intervall)	Zeitstempel	Rundet den Integer-Wert des `timestamp` Parameters auf das nächste Vielfache des Integer-Werts des `interval` Parameters ab. Die Bedeutung dieses Rückgabewerts ist möglicher weise nicht offensichtlich. Er wird mithilfe der Ganzzahl-Arithmetik berechnet, indem zuerst die Ganzzahl mit dem Zeitstempel durch die Ganzzahl für das Intervall dividiert und dann das Ergebnis mit der Intervallzahl multipliziert wird. Wenn man bedenkt, dass ein Zeitstempel einen UTC Zeitpunkt als Anzahl von Sekundenbruchteilen angibt, die seit der POSIX Epoche (1. Januar 1970) vergangen sind, stimmt der Rückgabewert selten mit Kalendereinheiten überein. Wenn Sie beispiels weise ein Intervall von 30 Tagen angeben, werden alle Tage seit der Epoche in 30-Tage-Inkremente unterteilt, und der Beginn des letzten 30-Tage-Inkrements wird zurückgegeben, was keinen

Funktion	Ausgabedatentyp	Beschreibung

		Bezug zu Kalendermonaten hat.
		
		Hier sind einige Beispiele:
		
		`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`

vor (Intervall)	Zeitstempel	Gibt den Wert zurück, der current_timestamp `interval` entspricht. `SELECT ago(1d)` Beispielergebnis: `2022-07-06 21:08:53.245000000`

Funktion	Ausgabedatentyp	Beschreibung
Intervallliterale wie 1h, 1d und 30m	Intervall	Intervallliterale sind praktisch für parse_duration (string). Zum Beispiel ist 1d identisch zu `parse_duration('1d')` . Dies ermöglicht die Verwendung der Literale überall dort, wo ein Intervall verwendet wird. Beispiel: `ago(1d)` und `bin(<timestamp> , 1m)`.

Einige Intervallliterale dienen als Abkürzung für parse_duration. Zum Beispiel,,, und geben 1d jeweils `parse_duration('1day')` zurück `1dayparse_duration('1d')`, wo sich der Typ befindet. `1 00:00:00.000000000` `interval day to second` Leerzeichen sind in dem Format zulässig, das für bereitgestellt wurde`parse_duration`. Gibt zum Beispiel `parse_duration('1day')` auch zurück`00:00:00.000000000`. `1 day`Ist aber kein Intervall wörtlich.

Die Einheiten, auf die sich beziehen, `interval day to second` sind ns, Nanosekunde, us, Mikrosekunde, ms, Millisekunde, s, Sekunde, m, Minute, h, Stunde, d und Tag.

Es gibt auch. `interval year to month` Die Einheiten, die sich auf das Intervall von Jahr zu Monat beziehen, sind Y, Jahr und Monat. Gibt zum Beispiel `SELECT 1year` zurück`1-0`. `SELECT 12month`kehrt auch zurück`1-0`. `SELECT 8month`kehrt zurück`0-8`.

Obwohl die Einheit von `quarter` auch für einige Funktionen wie `date_trunc` und verfügbar `quarter` ist`date_add`, ist sie nicht als Teil eines Intervallliterals verfügbar.

Formatieren und Analysieren

Timestream for LiveAnalytics unterstützt die folgenden Formatierungs- und Analysefunktionen für Datum und Uhrzeit.

Funktion	Ausgabedatentyp	Beschreibung
date_format (Zeitstempel, varchar (x))	varchar	Weitere Hinweise zu den von dieser Funktion verwendeten

Funktion	Ausgabedatentyp	Beschreibung
		Formatbezeichnern finden Sie unter # 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')` Beispielergebnis: `2019-10-20 10:20:20`
date_parse (varchar (x), varchar (y))	Zeitstempel	Weitere Hinweise zu den von dieser Funktion verwendeten Formatbezeichnern finden Sie unter # 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')` Beispielergebnis: `2019-10-20 10:20:20.000000000`

Funktion	Ausgabedatentyp	Beschreibung
format_datetime (Zeitstempel, varchar (x))	varchar	Weitere Informationen zu der von dieser Funktion verwendeten Formatzeichenfolge finden Sie unter 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')` Beispielergebnis: `1968-01-13 12`
parse_datetime (varchar (x), varchar (y))	Zeitstempel	Weitere Informationen zu der von dieser Funktion verwendeten Formatzeichenfolge finden Sie unter 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')` Beispielergebnis: `2019-12-29 18:10:00.000000000`

Extraktion

Timestream for LiveAnalytics unterstützt die folgenden Extraktionsfunktionen für Datum und Uhrzeit. Die Extraktfunktion ist die Grundlage für die übrigen Komfortfunktionen.

Funktion	Datentyp der Ausgabe	Beschreibung
extract	bigint	Extrahiert ein Feld aus einem Zeitstempel, wobei das Feld einem der Werte [YEAR,,,, QUARTERMONTH, _OF_ WEEKDAY, DAY _OF_MONTH,, DAY _OF_WEEK, DAY _OF_DOW,, YEAR,DOY, YEAR oder] entspricht. WEEK YOW HOUR MINUTE SECOND `SELECT extract(YEAR FROM '2019-10-12 23:10:34.000000000')` Beispielergebnis: 2019
Tag (Zeitstempel), Tag (Datum), Tag (Intervall Tag bis Sekunde)	bigint	`SELECT day('2019-10-12 23:10:34.000000000')` Beispiel für ein Ergebnis: 12
day_of_month (timestamp), day_of_month (Datum), day_of_month (Intervall von Tag bis Sekunde)	bigint	`SELECT day_of_month('2019-10-12 23:10:34.000000000')` Beispiel für ein Ergebnis: 12

Funktion	Datentyp der Ausgabe	Beschreibung
day_of_week (Zeitstempel), day_of_week (Datum)	bigint	`SELECT day_of_week('2019-10-12 23:10:34.000000000')` Beispielergebnis: 6
day_of_year (Zeitstempel), day_of_year (Datum)	bigint	`SELECT day_of_year('2019-10-12 23:10:34.000000000')` Beispielergebnis: 285
dow (Zeitstempel), dow (Datum)	bigint	Alias für day_of_week
doy (Zeitstempel), doy (Datum)	bigint	Alias für day_of_year
Stunde (Zeitstempel), Stunde (Zeit), Stunde (Intervall Tag bis Sekunde)	bigint	`SELECT hour('2019-10-12 23:10:34.000000000')` Beispielergebnis: 23
Millisekunde (Zeitstempel), Millisekunde (Zeit), Millisekunde (Intervall von Tag zu Sekunde)	bigint	`SELECT millisecond('2019-10-12 23:10:34.000000000')` Beispielergebnis: 0
Minute (Zeitstempel), Minute (Zeit), Minute (Intervall von Tag zu Sekunde)	bigint	`SELECT minute('2019-10-12 23:10:34.000000000')` Beispiel für ein Ergebnis: 10

Funktion	Datentyp der Ausgabe	Beschreibung
Monat (Zeitstempel), Monat (Datum), Monat (Intervall von Jahr zu Monat)	bigint	`SELECT month('2019-10-12 23:10:34.000000000')` Beispielergebnis: `10`
Nanosekunde (Zeitstempel), Nanosekunde (Zeit), Nanosekunde (Intervall von Tag zu Sekunde)	bigint	`SELECT nanosecond(current_timestamp)` Beispiel für ein Ergebnis: `162000000`
Quartal (Zeitstempel), Quartal (Datum)	bigint	`SELECT quarter('2019-10-12 23:10:34.000000000')` Beispielergebnis: `4`
Sekunde (Zeitstempel), Sekunde (Zeit), Sekunde (Intervall von Tag zu Sekunde)	bigint	`SELECT second('2019-10-12 23:10:34.000000000')` Beispiel für ein Ergebnis: `34`
Woche (Zeitstempel), Woche (Datum)	bigint	`SELECT week('2019-10-12 23:10:34.000000000')` Beispiel für ein Ergebnis: `41`
week_of_year (Zeitstempel), week_of_year (Datum)	bigint	Alias für Woche

Funktion	Datentyp der Ausgabe	Beschreibung
Jahr (Zeitstempel), Jahr (Datum), Jahr (Intervall von Jahr zu Monat)	bigint	`SELECT year('2019-10-12 23:10:34.000000000')` Beispielergebnis: 2019
year_of_week (Zeitstempel), year_of_week (Datum)	bigint	`SELECT year_of_week('2019-10-12 23:10:34.000000000')` Beispielergebnis: 2019
yow (Zeitstempel), yow (Datum)	bigint	Alias für year_of_week

Aggregationsfunktionen

Timestream for LiveAnalytics unterstützt die folgenden Aggregatfunktionen.

Funktion	Ausgabedatentyp	Beschreibung
willkürlich (x)	[dasselbe wie Eingabe]	Gibt einen beliebigen Wert von x zurück, der ungleich Null ist, falls einer existiert. `SELECT arbitrary(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Beispielergebnis: 1
array_agg (x)	array< [dasselbe wie Eingabe]	Gibt ein Array zurück, das aus den X-Eingabeelementen erstellt wurde.

Funktion	Ausgabedatentyp	Beschreibung
		`SELECT array_agg(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Beispielergebnis: `[ 1,2,3,4 ]`
avg (x)	double	Gibt den Durchschnitt (arithmetisches Mittel) aller Eingabewerte zurück. `SELECT avg(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Beispielergebnis: `2.5`
bool_and (boolean) jeder (boolean)	boolesch	Gibt zurück, ob jeder Eingabewert istTRUE, andernfalls. TRUE FALSE `SELECT bool_and(t.c) FROM (VALUES true, true, false, true) AS t(c)` Beispielergebnis: `false`

Funktion	Ausgabedatentyp	Beschreibung
bool_or (boolean)	boolesch	Gibt zurück, TRUE ob ein Eingabewert ist, andernfalls. TRUE FALSE `SELECT bool_or(t.c) FROM (VALUES true, true, false, true) AS t(c)` Beispiel für ein Ergebnis: `true`
zählen (*) zählen (x)	bigint	count (*) gibt die Anzahl der Eingabezeilen zurück. count (x) gibt die Anzahl der Eingabewerte zurück, die nicht Null sind. `SELECT count(t.c) FROM (VALUES true, true, false, true) AS t(c)` Beispielergebnis: 4
count_if (x)	bigint	Gibt die Anzahl der TRUE Eingabewerte zurück. `SELECT count_if(t.c) FROM (VALUES true, true, false, true) AS t(c)` Beispielergebnis: 3

Funktion	Ausgabedatentyp	Beschreibung
geometric_mean (x)	double	Gibt den geometrischen Mittelwert aller Eingabewerte zurück. `SELECT geometric _mean(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Beispielergebnis: `2.213363839400643`
max_by (x, y)	[dasselbe wie x]	Gibt den Wert von x zurück, der dem Maximalwert von y für alle Eingabewerte zugeordnet ist. `SELECT max_by(t.c1, t.c2) FROM (VALUES (('a', 1)), (('b', 2)), (('c', 3)), (('d', 4))) AS t(c1, c2)` Beispielergebnis: d

Funktion	Ausgabedatentyp	Beschreibung
max_by (x, y, n)	Reihe< [same as x] >	Gibt n Werte von x zurück, die dem n-größten aller Eingabewerte von y zugeordnet sind, in absteigen der Reihenfolge von y. `SELECT max_by(t.c1, t.c2, 2) FROM (VALUES (('a', 1)), (('b', 2)), (('c', 3)), (('d', 4))) AS t(c1, c2)` Beispielergebnis: `[ d,c ]`
min_by (x, y)	[dasselbe wie x]	Gibt den Wert von x zurück, der dem Minimalwert von y für alle Eingabewerte zugeordnet ist. `SELECT min_by(t.c1, t.c2) FROM (VALUES (('a', 1)), (('b', 2)), (('c', 3)), (('d', 4))) AS t(c1, c2)` Beispielergebnis: a

Funktion	Ausgabedatentyp	Beschreibung
min_by (x, y, n)	Reihe< [same as x] >	Gibt n Werte von x zurück, die dem n kleinsten aller Eingabewerte von y zugeordnet sind, in aufsteige nder Reihenfolge von y. `SELECT min_by(t.c1, t.c2, 2) FROM (VALUES (('a', 1)), (('b', 2)), (('c', 3)), (('d', 4))) AS t(c1, c2)` Beispielergebnis: `[ a,b ]`
max (x)	[dasselbe wie Eingabe]	Gibt den Maximalwert aller Eingabewerte zurück. `SELECT max(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Beispielergebnis: `4`
max (x, n)	Reihe< [same as x] >	Gibt n größte Werte aller Eingabewerte von x zurück. `SELECT max(t.c, 2) FROM (VALUES 1, 2, 3, 4) AS t(c)` Beispielergebnis: `[ 4,3 ]`

Funktion	Ausgabedatentyp	Beschreibung
min (x)	[dasselbe wie Eingabe]	Gibt den Minimalwert aller Eingabewerte zurück. `SELECT min(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Beispielergebnis: 1
min (x, n)	Reihe< [same as x] >	Gibt n kleinste Werte aller Eingabewerte von x zurück. `SELECT min(t.c, 2) FROM (VALUES 1, 2, 3, 4) AS t(c)` Beispielergebnis: `[ 1,2 ]`
Summe (x)	[dasselbe wie Eingabe]	Gibt die Summe aller Eingabewerte zurück. `SELECT sum(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Beispielergebnis: `10`

Funktion	Ausgabedatentyp	Beschreibung
bitwise_and_agg (x)	bigint	Gibt den bitweisen Wert aller Eingabewerte in AND 2s-Komplement-Darstellung zurück. `SELECT bitwise_and_agg(t.c) FROM (VALUES 1, -3) AS t(c)` Beispielergebnis: 1
bitwise_or_agg (x)	bigint	Gibt das bitweise ODER aller Eingabewerte in 2s-Komplement-Darstellung zurück. `SELECT bitwise_or_agg(t.c) FROM (VALUES 1, -3) AS t(c)` Beispielergebnis: -3

Funktion	Ausgabedatentyp	Beschreibung
approx_distinct (x)	bigint	Gibt die ungefähre Anzahl verschiedener Eingabewe rte zurück. Diese Funktion liefert eine Näherung von count (DISTINCTx). Null wird zurückgegeben, wenn alle Eingabewerte Null sind. Diese Funktion sollte einen Standardfehler von 2,3% ergeben, was der Standarda bweichung der (annähernd normalen) Fehlerverteilung über alle möglichen Mengen entspricht. Sie garantiert keine Obergrenze für den Fehler für einen bestimmten Eingabesa tz. `SELECT approx_di stinct(t.c) FROM (VALUES 1, 2, 3, 4, 8) AS t(c)` Beispiel für ein Ergebnis: 5

Funktion	Ausgabedatentyp	Beschreibung
approx_distinct (x, e)	bigint	Gibt die ungefähre Anzahl unterschiedlicher Eingabewerte zurück. Diese Funktion liefert eine Näherung von count (DISTINCTx). Null wird zurückgegeben, wenn alle Eingabewerte Null sind. Diese Funktion sollte einen Standardfehler von nicht mehr als e erzeugen, was der Standardabweichung der (ungefähr normalen) Fehlerverteilung über alle möglichen Mengen entspricht. Sie garantiert keine Obergrenze für den Fehler für einen bestimmten Eingabesatz. Die aktuelle Implementierung dieser Funktion erfordert, dass e im Bereich von [0,0040625, 0,26000] liegt. `SELECT approx_distinct(t.c, 0.2) FROM (VALUES 1, 2, 3, 4, 8) AS t(c)` Beispielergebnis: 5

Funktion	Ausgabedatentyp	Beschreibung
approx_percentile (x, Prozent)	[dasselbe wie x]	Gibt das ungefähre Perzentil für alle Eingabewerte von x zum angegebenen Prozentsatz zurück. Der Prozentwert muss zwischen Null und Eins liegen und für alle Eingabezeilen konstant sein. `SELECT approx_percentile(t.c, 0.4) FROM (VALUES 1, 2, 3, 4) AS t(c)` Beispiel für ein Ergebnis: 2
approx_percentile (x, Prozentsätze)	Reihe< [same as x] >	Gibt das ungefähre Perzentil für alle Eingabewerte von x bei jedem der angegebenen Prozentsätze zurück. Jedes Element der Prozentsatzmatrix muss zwischen Null und Eins liegen, und die Matrix muss für alle Eingabezeilen konstant sein. `SELECT approx_percentile(t.c, ARRAY[0.1, 0.8, 0.8]) FROM (VALUES 1, 2, 3, 4) AS t(c)` Beispielergebnis: `[ 1,4,4 ]`

Funktion	Ausgabedatentyp	Beschreibung
approx_percentile (x, w, prozentual)	[dasselbe wie x]	Gibt das ungefähre gewichtete Perzentil für alle Eingabewerte von x zurück, wobei das Gewicht w pro Artikel beim Prozentsatz p verwendet wird. Das Gewicht muss ein ganzzahliger Wert von mindestens eins sein. Es handelt sich praktisch um eine Replikationszahl für den Wert x im Perzentilsatz. Der Wert von p muss zwischen Null und Eins liegen und für alle Eingabezeilen konstant sein. `SELECT approx_percentile(t.c, 1, 0.1) FROM (VALUES 1, 2, 3, 4) AS t(c)` Beispielergebnis: 1

Funktion	Ausgabedatentyp	Beschreibung
approx_percentile (x, w, Prozentsätze)	Reihe< [same as x] >	Gibt das ungefähre gewichtete Perzentil für alle Eingabewerte von x zurück, wobei das Gewicht w pro Element bei jedem der im Array angegebenen Prozentsätze verwendet wird. Bei der Gewichtung muss es sich um einen ganzzahligen Wert von mindestens eins handeln. Es handelt sich praktisch um eine Replikationszahl für den Wert x im Perzentilsatz. Jedes Element des Arrays muss zwischen Null und Eins liegen, und das Array muss für alle Eingabezeilen konstant sein. `SELECT approx_percentile(t.c, 1, ARRAY[0.1, 0.8, 0.8]) FROM (VALUES 1, 2, 3, 4) AS t(c)` Beispielergebnis: `[ 1,4,4 ]`

Funktion	Ausgabedatentyp	Beschreibung
approx_percentile (x, w, Prozentsatz, Genauigkeit)	[wie x]	Gibt das ungefähre gewichtete Perzentil für alle Eingabewerte von x zurück, wobei das Gewicht w pro Artikel als Prozentsatz p verwendet wird, wobei ein maximaler Rangfehler der Genauigkeit verwendet wird. Bei der Gewichtung muss es sich um einen ganzzahligen Wert von mindestens eins handeln. Es handelt sich praktisch um eine Replikationszahl für den Wert x im Perzentilsatz. Der Wert von p muss zwischen Null und Eins liegen und für alle Eingabezeilen konstant sein. Die Genauigkeit muss ein Wert größer als Null und kleiner als eins sein, und sie muss für alle Eingabezeilen konstant sein. `SELECT approx_percentile(t.c, 1, 0.1, 0.5) FROM (VALUES 1, 2, 3, 4) AS t(c)` Beispielergebnis: 1

Funktion	Ausgabedatentyp	Beschreibung
corr (y, x)	double	Gibt den Korrelationskoeffizienten der Eingabewerte zurück. `SELECT corr(t.c1, t.c2) FROM (VALUES ((1, 1)), ((2, 2)), ((3, 3)), ((4, 4))) AS t(c1, c2)` Beispielergebnis: `1.0`
covar_pop (y, x)	double	Gibt die Populationskovarianz der Eingabewerte zurück. `SELECT covar_pop(t.c1, t.c2) FROM (VALUES ((1, 1)), ((2, 2)), ((3, 3)), ((4, 4))) AS t(c1, c2)` Beispiel für ein Ergebnis: `1.25`
covar_samp (y, x)	double	Gibt die Stichprobenkovarianz der Eingabewerte zurück. `SELECT covar_samp(t.c1, t.c2) FROM (VALUES ((1, 1)), ((2, 2)), ((3, 3)), ((4, 4))) AS t(c1, c2)` Beispielergebnis: `1.6666666666666667`

Funktion	Ausgabedatentyp	Beschreibung
regr_intercept (y, x)	double	Gibt den linearen Regressio nsabschnitt der Eingabewe rte zurück. y ist der abhängige Wert. x ist der unabhängige Wert. `SELECT regr_inte rcept(t.c1, t.c2) FROM (VALUES ((1, 1)), ((2, 2)), ((3, 3)), ((4, 4))) AS t(c1, c2)` Beispiel für ein Ergebnis: `0.0`
regr_slope (y, x)	double	Gibt die lineare Regressio nssteigung der Eingabewerte zurück. y ist der abhängige Wert. x ist der unabhängige Wert. `SELECT regr_slope(t.c1, t.c2) FROM (VALUES ((1, 1)), ((2, 2)), ((3, 3)), ((4, 4))) AS t(c1, c2)` Beispielergebnis: `1.0`

Funktion	Ausgabedatentyp	Beschreibung
Schiefe (x)	double	Gibt die Schiefe aller Eingabewerte zurück. `SELECT skewness(t.c1) FROM (VALUES 1, 2, 3, 4, 8) AS t(c1)` Beispielergebnis: `0.8978957037987335`
stddev_pop (x)	double	Gibt die Populationsstandardabweichung aller Eingabewerte zurück. `SELECT stddev_pop(t.c1) FROM (VALUES 1, 2, 3, 4, 8) AS t(c1)` Beispielergebnis: `2.4166091947189146`
stddev_samp (x) stddev (x)	double	Gibt die Standardabweichung der Stichprobe aller Eingabewerte zurück. `SELECT stddev_samp(t.c1) FROM (VALUES 1, 2, 3, 4, 8) AS t(c1)` Beispielergebnis: `2.7018512172212592`

Funktion	Ausgabedatentyp	Beschreibung
var_pop (x)	double	Gibt die Populationsvarianz aller Eingabewerte zurück. `SELECT var_pop(t.c1) FROM (VALUES 1, 2, 3, 4, 8) AS t(c1)` Beispiel für ein Ergebnis: `5.840000000000001`
var_samp (x) Varianz (x)	double	Gibt die Stichprobenvarianz aller Eingabewerte zurück. `SELECT var_samp(t.c1) FROM (VALUES 1, 2, 3, 4, 8) AS t(c1)` Beispielergebnis: `7.300000000000001`

Fensterfunktionen

Fensterfunktionen führen Berechnungen für mehrere Zeilen des Abfrageergebnisses durch. Sie werden nach der HAVING Klausel, aber vor der ORDER BY-Klausel ausgeführt. Das Aufrufen einer Fensterfunktion erfordert eine spezielle Syntax, bei der die OVER Klausel zur Angabe des Fensters verwendet wird. Ein Fenster besteht aus drei Komponenten:

- Die Partitionsspezifikation, die die Eingabezeilen in verschiedene Partitionen unterteilt. Dies entspricht der Art und Weise, wie die GROUP BY-Klausel Zeilen für Aggregatfunktionen in verschiedene Gruppen unterteilt.
- Die Sortierspezifikation, die die Reihenfolge bestimmt, in der Eingabezeilen von der Fensterfunktion verarbeitet werden.
- Der Fensterrahmen, der ein verschiebbares Fenster mit Zeilen angibt, die von der Funktion für eine bestimmte Zeile verarbeitet werden sollen. Wenn der Rahmen nicht angegeben ist, wird standardmäßig der Wert verwendet RANGE UNBOUNDEDPRECEDING, was derselbe ist wie

RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENTROW. Dieser Frame enthält alle Zeilen vom Anfang der Partition bis zum letzten Peer der aktuellen Zeile.

Alle Aggregatfunktionen können als Fensterfunktionen verwendet werden, indem die OVER Klausel hinzugefügt wird. Die Aggregatfunktion wird für jede Zeile über die Zeilen innerhalb des Fensterrahmens der aktuellen Zeile berechnet. Zusätzlich zu den Aggregatfunktionen LiveAnalytics unterstützt Timestream for die folgenden Rangfolge- und Wertfunktionen.

Funktion	Ausgabedatentyp	Beschreibung
cume_dist ()	bigint	Gibt die kumulative Verteilung eines Werts in einer Gruppe von Werten zurück. Das Ergebnis ist die Anzahl der Zeilen, die der Zeile in der Fensterreihenfolge der Fensterpartition vorausgehen oder ihr gleichrangig sind, geteilt durch die Gesamtzahl der Zeilen in der Fensterpartition. Somit ergeben alle Gleichheitswerte in der Reihenfolge denselben Verteilungswert.
dense_rank ()	bigint	Gibt den Rang eines Wertes in einer Gruppe von Werten zurück. Dies ist mit rank () vergleichbar, mit dem Unterschied, dass Gleichheitswerte keine Lücken in der Sequenz erzeugen.
antil (n)	bigint	Unterteilt die Zeilen für jede Fensterpartition in n Buckets im Bereich von 1 bis höchstens n. Die Bucket-

Funktion	Ausgabedatentyp	Beschreibung
		Werte unterscheiden sich höchstens um 1. Wenn sich die Anzahl der Zeilen in der Partition nicht gleichmäßig in die Anzahl der Buckets aufteilt, werden die restliche n Werte einzeln pro Bucket verteilt, beginnend mit dem ersten Bucket.
percent_rank ()	double	Gibt die prozentuale Rangfolge eines Werts in einer Gruppe von Werten zurück. Das Ergebnis ist (r - 1)/(n - 1), wobei r der Rang () der Zeile und n die Gesamtzahl der Zeilen in der Fensterpartition ist.
Rang ()	bigint	Gibt den Rang eines Werts in einer Gruppe von Werten zurück. Der Rang ist eins plus der Anzahl der Zeilen vor der Zeile, die der Zeile nicht ebenbürtig sind. Somit führen Gleichheitswerte in der Reihenfolge zu Lücken in der Reihenfolge. Die Rangfolge wird für jede Fensterpartition durchgeführt.

Funktion	Ausgabedatentyp	Beschreibung
Zeilennummer ()	bigint	Gibt eine eindeutige, sequentielle Zahl für jede Zeile zurück, beginnend mit eins, entsprechend der Reihenfol ge der Zeilen innerhalb der Fensterpartition.
erster_Wert (x)	[dasselbe wie Eingabe]	Gibt den ersten Wert des Fensters zurück. Diese Funktion ist auf den Fensterra hmen beschränkt. Die Funktion verwendet einen Ausdruck oder ein Ziel als Parameter.
last_value (x)	[dasselbe wie Eingabe]	Gibt den letzten Wert des Fensters zurück. Diese Funktion ist auf den Fensterra hmen beschränkt. Die Funktion verwendet einen Ausdruck oder ein Ziel als Parameter.

Funktion	Ausgabedatentyp	Beschreibung
nth_value (x, Offset)	[wie Eingabe]	Gibt den Wert am angegebenen Offset vom Anfang des Fensters zurück. Offsets beginnen bei 1. Der Offset kann ein beliebiger skalarer Ausdruck sein. Wenn der Offset Null oder größer als die Anzahl der Werte im Fenster ist, wird Null zurückgegeben. Es ist ein Fehler, wenn der Offset Null oder negativ ist. Die Funktion verwendet einen Ausdruck oder ein Ziel als ersten Parameter.
lead (x [, offset [, default_value]])	[dasselbe wie Eingabe]	Gibt den Wert an den versetzten Zeilen hinter der aktuellen Zeile im Fenster zurück. Offsets beginnen bei 0, was der aktuellen Zeile entspricht. Der Offset kann ein beliebiger skalarer Ausdruck sein. Der Standard-Offset ist 1. Wenn der Offset Null oder größer als das Fenster ist, wird der Standardwert zurückgegeben, oder wenn er nicht angegeben ist, wird Null zurückgegeben. Die Funktion verwendet einen Ausdruck oder ein Ziel als ersten Parameter.

Funktion	Ausgabedatentyp	Beschreibung
lag (x [, Offset [, Standardwert]])	[wie Eingabe]	Gibt den Wert zurück, der vor der aktuellen Zeile im Fenster liegt. Offsets beginnen bei 0, was der aktuellen Zeile entspricht. Der Offset kann ein beliebiger skalarer Ausdruck sein. Der Standard-Offset ist 1. Wenn der Offset Null oder größer als das Fenster ist, wird der Standardwert zurückgegeben, oder wenn er nicht angegeben ist, wird Null zurückgegeben. Die Funktion verwendet einen Ausdruck oder ein Ziel als ersten Parameter.

Beispielabfragen

Dieser Abschnitt enthält Anwendungsbeispiele für die Abfragesprache von Timestream for LiveAnalytics.

Themen

- Einfache Abfragen
- Abfragen mit Zeitreihenfunktionen
- Abfragen mit Aggregatfunktionen

Einfache Abfragen

Im Folgenden werden die 10 zuletzt hinzugefügten Datenpunkte für eine Tabelle abgerufen.

```
SELECT * FROM <database_name>.<table_name>
ORDER BY time DESC
LIMIT 10
```

Im Folgenden werden die 5 ältesten Datenpunkte für eine bestimmte Kennzahl abgerufen.

```
SELECT * FROM <database_name>.<table_name>
WHERE measure_name = '<measure_name>'
ORDER BY time ASC
LIMIT 5
```

Das Folgende funktioniert mit Zeitstempeln mit einer Granularität von Nanosekunden.

```
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
```

Messwerte für Datensätze mit mehreren Kennzahlen werden anhand des Spaltennamens identifiziert. Messwerte für Datensätze mit einer Kennzahl werden anhand von`measure_value::<data_type>`, wo einer von,, oder `<data_type>` steht `double``bigint`, oder `varchar` wie unter beschrieben`boolean`, identifiziert. Unterstützte Datentypen Weitere Informationen zur Modellierung von Kennzahlwerten finden Sie unter Einzelne Tabelle im Vergleich zu mehreren Tabellen.

Im Folgenden werden Werte für eine Kennzahl abgerufen, die `speed` aus Datensätzen mit mehreren Kennzahlen mit dem Wert von aufgerufen werden. `measure_name` `IoTMulti-stats`

```
SELECT speed FROM <database_name>.<table_name> where measure_name = 'IoTMulti-stats'
```

Im Folgenden werden `double` Werte aus Datensätzen mit einer Kennzahl von abgerufen. `measure_name` `load`

```
SELECT measure_value::double FROM <database_name>.<table_name> WHERE measure_name =
 'load'
```

Abfragen mit Zeitreihenfunktionen

Themen

- Beispieldatensatz und Abfragen

Beispieldatensatz und Abfragen

Sie können Timestream verwenden LiveAnalytics , um die Leistung und Verfügbarkeit Ihrer Dienste und Anwendungen zu verstehen und zu verbessern. Im Folgenden finden Sie eine Beispieltabelle und Beispielabfragen, die auf dieser Tabelle ausgeführt werden.

In der Tabelle `ec2_metrics` werden Telemetriedaten wie CPU Auslastung und andere Messwerte von EC2 Instances gespeichert. Sie können die Tabelle unten einsehen.

Zeit	Region	az	Hostname	measure_name	Messwert::doppelt	Messwert::bigint
2019-12-04 19:00:00.000000000	us-east-1	us-ost-1a	Frontend01	CPU-Auslastung	35,1	Null
2019-12-04 19:00:00.000000000	us-east-1	us-ost-1a	Frontend01	memory_utilization	55,3	Null
2019-12-04 19:00:00.000000000	us-east-1	us-ost-1a	Frontend01	Netzwerk-Bytes-Eingang	Null	1.500
2019-12-04 19:00:00.000000000	us-east-1	us-ost-1a	Frontend01	netzwerk_bytes_out	Null	6.700
2019-12-04 19:00:00.000000000	us-east-1	us-ost-1b	Frontend02	CPU-Auslastung	38,5	Null
2019-12-04 19:00:00.	us-east-1	us-ost-1b	Frontend02	memory_utilization	58,4	Null

Zeit	Region	az	Hostname	measure_name	Messwert::doppelt	Messwert::bigint
000 000000						
2019-12-04 19:00:00.000 000000	us-east-1	us-ost-1b	Frontend 02	Netzwerk-Bytes-Eingang	Null	23.000
2019-12-04 19:00:00.000 000000	us-east-1	us-ost-1b	Frontend 02	netzwerk_bytes_out	Null	12.000
2019-12-04 19:00:00.000 000000	us-east-1	us-ost-1c	Frontend 03	CPU-Auslastung	45.0	Null
2019-12-04 19:00:00.000 000000	us-east-1	us-ost-1c	Frontend 03	memory_utilization	65,8	Null
2019-12-04 19:00:00.000 000000	us-east-1	us-ost-1c	Frontend 03	netzwerk_bytes_in	Null	15 000
2019-12-04 19:00:00.000 000000	us-east-1	us-ost-1c	Frontend 03	netzwerk_bytes_out	Null	836.000

Zeit	Region	az	Hostname	measure_name	Messwert::doppelt	Messwert::bigint
2019-12-04 19:00:05.000000000	us-east-1	us-ost-1a	Frontend01	CPU-Auslastung	55,2	Null
2019-12-04 19:00:05.000000000	us-east-1	us-ost-1a	Frontend01	memory_utilization	75.0	Null
2019-12-04 19:00:05.000000000	us-east-1	us-ost-1a	Frontend01	Netzwerk-Bytes-Eingang	Null	1.245
2019-12-04 19:00:05.000000000	us-east-1	us-ost-1a	Frontend01	netzwerk_bytes_out	Null	68.432
2019-12-04 19:00:08.000000000	us-east-1	us-ost-1b	Frontend02	CPU-Auslastung	65,6	Null
2019-12-04 19:00:08.000000000	us-east-1	us-ost-1b	Frontend02	memory_utilization	85,3	Null
2019-12-04 19:00:08.000000000	us-east-1	us-ost-1b	Frontend02	Netzwerk-Bytes-Eingang	Null	1.245

Zeit	Region	az	Hostname	measure_name	Messwert::doppelt	Messwert::bigint
2019-12-04 19:00:08.000 000000	us-east-1	us-ost-1b	Frontend 02	netzwerk_bytes_out	Null	68.432
2019-12-04 19:00:20.000 000000	us-east-1	us-ost-1c	Frontend 03	CPU-Auslastung	12.1	Null
2019-12-04 19:00:20.000 000000	us-east-1	us-ost-1c	Frontend 03	memory_utilization	32,0	Null
2019-12-04 19:00:20.000 000000	us-east-1	us-ost-1c	Frontend 03	netzwerk_bytes_in	Null	1.400
2019-12-04 19:00:20.000 000000	us-east-1	us-ost-1c	Frontend 03	netzwerk_bytes_out	Null	345
2019-12-04 19:00:10.000 000000	us-east-1	us-ost-1a	Frontend 01	CPU-Auslastung	15.3	Null
2019-12-04 19:00:10.000 000000	us-east-1	us-ost-1a	Frontend 01	memory_utilization	35,4	Null

Zeit	Region	az	Hostname	measure_name	Messwert::doppelt	Messwert::bigint
2019-12-04 19:00:10.000000000	us-east-1	us-ost-1a	Frontend01	Netzwerk-Bytes-Eingang	Null	23
2019-12-04 19:00:10.000000000	us-east-1	us-ost-1a	Frontend01	netzwerk_bytes_out	Null	0
2019-12-04 19:00:16.000000000	us-east-1	us-ost-1b	Frontend02	CPU-Auslastung	44.0	Null
2019-12-04 19:00:16.000000000	us-east-1	us-ost-1b	Frontend02	memory_utilization	64,2	Null
2019-12-04 19:00:16.000000000	us-east-1	us-ost-1b	Frontend02	Netzwerk-Bytes-Eingang	Null	1.450
2019-12-04 19:00:16.000000000	us-east-1	us-ost-1b	Frontend02	netzwerk_bytes_out	Null	200
2019-12-04 19:00:40.000000000	us-east-1	us-ost-1c	Frontend03	CPU-Auslastung	66,4	Null

Zeit	Region	az	Hostname	measure_name	Messwert::doppelt	Messwert::bigint
2019-12-04 19:00:40.000 000000	us-east-1	us-ost-1c	Frontend 03	memory_utilization	86,3	Null
2019-12-04 19:00:40.000 000000	us-east-1	us-ost-1c	Frontend 03	netzwerk_bytes_in	Null	300
2019-12-04 19:00:40.000 000000	us-east-1	us-ost-1c	Frontend 03	netzwerk_bytes_out	Null	423

Ermitteln Sie die durchschnittliche CPU Auslastung von p90, p95 und p99 für einen bestimmten EC2 Host in den letzten 2 Stunden:

```
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
```

Identifizieren Sie EC2 Hosts, deren CPU Auslastung um 10% oder mehr höher ist als die durchschnittliche CPU Auslastung der gesamten Flotte in den letzten 2 Stunden:

```
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
```

Ermitteln Sie die durchschnittliche CPU Auslastung in Intervallen von 30 Sekunden für einen bestimmten EC2 Host in den letzten 2 Stunden:

```
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
```

Ermitteln Sie die durchschnittliche CPU Auslastung in Intervallen von 30 Sekunden für einen bestimmten EC2 Host in den letzten 2 Stunden und füllen Sie die fehlenden Werte mit linearer Interpolation aus:

```
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)
```

Ermitteln Sie die durchschnittliche CPU Auslastung in Intervallen von 30 Sekunden für einen bestimmten EC2 Host in den letzten 2 Stunden und füllen Sie die fehlenden Werte durch Interpolation auf der Grundlage der letzten übertragenen Beobachtung aus:

```
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)
```

Abfragen mit Aggregatfunktionen

Im Folgenden finden Sie einen Beispieldatensatz für ein IoT-Szenario zur Veranschaulichung von Abfragen mit Aggregatfunktionen.

Themen

- Beispiel für Daten

- Beispielabfragen

Beispiel für Daten

Mit Timestream können Sie IoT-Sensordaten wie Standort, Kraftstoffverbrauch, Geschwindigkeit und Ladekapazität einer oder mehrerer LKW-Flotten speichern und analysieren, um ein effektives Flottenmanagement zu ermöglichen. Im Folgenden finden Sie das Schema und einige Daten einer Tabelle iot_trucks, in der Telemetriedaten wie Standort, Kraftstoffverbrauch, Geschwindigkeit und Ladekapazität von Lkw gespeichert werden.

Zeit	truck_id	Marke	Modell	Flotte	Treibstoffkapazität	Ladekapität	measure_name	Messwert:doppelt	Messwert::varchar
2019-12-4 19:00:00.000000000	1234567	GMC	Astro	Alpha (Alpha)	100	500	Treibstoff auslesen	65,2	Null
2019-12-4 19:00:00.000000000	1234567	GMC	Astro	Alpha (Alpha)	100	500	load	400,0	Null
2019-12-4 19:00:00.000000000	1234567	GMC	Astro	Alpha (Alpha)	100	500	speed	90,2	Null
2019-12-4 19:00:00.000000000	1234567	GMC	Astro	Alpha (Alpha)	100	500	location	Null	47,6062 N, 122,3321 W

Zeit	truck_id	Marke	Modell	Flotte	Treibstoffkapazität	Ladekapität	measure_name	Messwert:doppelt	Messwert:varchar
2019-12-4 19:00:00.000000000	1234567	Kenworth	W 900	Alpha (Alpha)	150	1000	Ausleser von Treibstoff	10.1	Null
2019-12-4 19:00:00.000000000	1234567	Kenworth	W 900	Alpha (Alpha)	150	1000	load	950,3	Null
2019-12-4 19:00:00.000000000	1234567	Kenworth	W 900	Alpha (Alpha)	150	1000	speed	50,8	Null
2019-12-4 19:00:00.000000000	1234567	Kenworth	W 900	Alpha (Alpha)	150	1000	location	Null	40,7128 Grad N, 74,0060 Grad W

Beispielabfragen

Rufen Sie eine Liste aller Sensorattribute und -werte ab, die für jeden Lkw in der Flotte überwacht werden.

```
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
```

Rufen Sie die neuesten Kraftstoffwerte für jeden Lkw in der Flotte in den letzten 24 Stunden ab.

```
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
```

Identifizieren Sie Lkw, die in den letzten 48 Stunden mit wenig Kraftstoff (weniger als 10%) gefahren sind:

```
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
)
```

Finden Sie die Durchschnittslast und die Höchstgeschwindigkeit für jeden Lkw in der letzten Woche heraus:

```
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
```

Ermitteln Sie die Ladeeffizienz für jeden Lkw in der letzten Woche:

```
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
```

APIReferenz

Dieser Abschnitt enthält die API Referenzdokumentation für Amazon Timestream.

Timestream hat zwei FunktionenAPIs: Query und Write.

- APIMit Write können Sie Operationen wie das Erstellen von Tabellen, das Markieren von Ressourcen und das Schreiben von Datensätzen in Timestream ausführen.
- Mit der Abfrage API können Sie Abfrageoperationen ausführen.

> **Note**
>
> Beide APIs beinhalten die DescribeEndpoints Aktion. Sowohl für Query als auch für Write sind die DescribeEndpoints Aktionen identisch.

APIIm Folgenden finden Sie weitere Informationen zu den einzelnen Datentypen, häufigen Fehlern und Parametern.

> **Note**
>
> Fehlercodes, die allen AWS Diensten gemeinsam sind, finden Sie im Abschnitt AWS Support.

Themen

Aktionen

Die folgenden Aktionen werden von Amazon Timestream Write unterstützt:

- ListDatabases
- ListTables
- ListTagsForResource
- ResumeBatchLoadTask
- TagResource
- UntagResource
- UpdateDatabase
- UpdateTable
- WriteRecords

Die folgenden Aktionen werden von Amazon Timestream Query unterstützt:

- CancelQuery
- CreateScheduledQuery
- DeleteScheduledQuery
- DescribeAccountSettings
- DescribeEndpoints
- DescribeScheduledQuery
- ExecuteScheduledQuery
- ListScheduledQueries
- ListTagsForResource
- PrepareQuery
- Query
- TagResource
- UntagResource
- UpdateAccountSettings
- UpdateScheduledQuery

Amazon Timestream Write

Die folgenden Aktionen werden von Amazon Timestream Write unterstützt:

- CreateBatchLoadTask
- CreateDatabase
- CreateTable
- DeleteDatabase
- DeleteTable
- DescribeBatchLoadTask
- DescribeDatabase
- DescribeEndpoints
- DescribeTable
- ListBatchLoadTasks
- ListDatabases
- ListTables
- ListTagsForResource
- ResumeBatchLoadTask
- TagResource
- UntagResource
- UpdateDatabase
- UpdateTable
- WriteRecords

CreateBatchLoadTask

Bedienung: Amazon Timestream Write

Erstellt eine neue Timestream-Aufgabe zum Laden von Batches. Eine Batch-Load-Task verarbeitet Daten aus einer CSV Quelle an einem S3-Standort und schreibt in eine Timestream-Tabelle. Eine Zuordnung von der Quelle zum Ziel wird in einer Batch-Load-Task definiert. Fehler und Ereignisse werden in einen Bericht an einem S3-Standort geschrieben. Wenn der AWS KMS Schlüssel für den Bericht nicht angegeben ist, wird der Bericht mit einem verwalteten S3-Schlüssel verschlüsselt, sofern dies möglich SSE_S3 ist. Andernfalls wird ein Fehler ausgegeben. Weitere Informationen finden Sie unter Von AWS verwaltete Schlüssel. Hier gelten Servicekontingente. Einzelheiten finden Sie im Codebeispiel.

Anforderungssyntax

```
{
   "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"
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

ClientToken

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Erforderlich: Nein

DataModelConfiguration

Typ: DataModelConfiguration Objekt

Erforderlich: Nein

DataSourceConfiguration

Definiert Konfigurationsdetails zur Datenquelle für eine Batch-Load-Task.

Typ: DataSourceConfiguration Objekt

Erforderlich: Ja

RecordVersion

Type: Long

Erforderlich: Nein

ReportConfiguration

Berichtskonfiguration für eine Batch-Load-Aufgabe. Dies enthält Details darüber, wo Fehlerberichte gespeichert werden.

Typ: ReportConfiguration Objekt

Erforderlich: Ja

TargetDatabaseName

Timestream-Zieldatenbank für eine Batch-Load-Aufgabe.

Typ: Zeichenfolge

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

TargetTableName

Ziel-Timestream-Tabelle für eine Batch-Load-Aufgabe.

Typ: Zeichenfolge

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

Antwortsyntax

```
{
   "TaskId": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Dienst im JSON Format zurückgegeben.

TaskId

Die ID der Batch-Load-Task.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 32 Zeichen.

Pattern: `[A-Z0-9]+`

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion auszuführen.

HTTPStatuscode: 400

ConflictException

Timestream konnte diese Anfrage nicht verarbeiten, da sie eine Ressource enthält, die bereits vorhanden ist.

HTTPStatuscode: 400

InternalServerException

Timestream konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 500

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ResourceNotFoundException

Der Vorgang hat versucht, auf eine nicht existierende Ressource zuzugreifen. Die Ressource wurde möglicherweise nicht richtig angegeben, oder ihr Status ist nicht ACTIVE korrekt.

HTTPStatuscode: 400

ServiceQuotaExceededException

Das Instanzkontingent der Ressource für dieses Konto wurde überschritten.

HTTPStatuscode: 400

ThrottlingException

Ein Benutzer hat zu viele Anfragen gestellt und sie haben die Servicekontingenten überschritten. Die Anforderung wurde gedrosselt.

HTTPStatuscode: 400

ValidationException

Eine ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

CreateDatabase

Bedienung: Amazon Timestream Write

Erstellt eine neue Timestream-Datenbank. Wenn der AWS KMS Schlüssel nicht angegeben ist, wird die Datenbank mit einem von Timestream verwalteten AWS KMS Schlüssel verschlüsselt, der sich in Ihrem Konto befindet. Weitere Informationen finden Sie unter Von AWS verwaltete Schlüssel. Hier gelten Servicekontingente. Einzelheiten finden Sie im Codebeispiel.

Anforderungssyntax

```
{
   "DatabaseName": "string",
   "KmsKeyId": "string",
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

DatabaseName

Der Name der Timestream-Datenbank.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 256 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

KmsKeyId

Der AWS KMS Schlüssel für die Datenbank. Wenn der AWS KMS Schlüssel nicht angegeben ist, wird die Datenbank mit einem von Timestream verwalteten AWS KMS Schlüssel verschlüsselt,

der sich in Ihrem Konto befindet. Weitere Informationen finden Sie unter Von AWS verwaltete Schlüssel.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Nein

Tags

Eine Liste von Schlüssel-Wert-Paaren zur Bezeichnung der Tabelle.

Typ: Array von Tag-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 200 Elemente.

Erforderlich: Nein

Antwortsyntax

```
{
   "Database": {
      "Arn": "string",
      "CreationTime": number,
      "DatabaseName": "string",
      "KmsKeyId": "string",
      "LastUpdatedTime": number,
      "TableCount": number
   }
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Dienst im JSON Format zurückgegeben.

Database

Die neu erstellte Timestream-Datenbank.

Typ: Database Objekt

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion auszuführen.

HTTPStatuscode: 400

ConflictException

Timestream konnte diese Anfrage nicht verarbeiten, da sie eine Ressource enthält, die bereits vorhanden ist.

HTTPStatuscode: 400

InternalServerException

Timestream konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 500

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ServiceQuotaExceededException

Das Instanzkontingent der Ressource für dieses Konto wurde überschritten.

HTTPStatuscode: 400

ThrottlingException

Ein Benutzer hat zu viele Anfragen gestellt und sie haben die Servicekontingenten überschritten. Die Anforderung wurde gedrosselt.

HTTPStatuscode: 400

ValidationException

Eine ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

CreateTable

Bedienung: Amazon Timestream Write

Fügt einer vorhandenen Datenbank in Ihrem Konto eine neue Tabelle hinzu. In einer AWS-Konto müssen Tabellennamen innerhalb jeder Region mindestens eindeutig sein, wenn sie sich in derselben Datenbank befinden. Möglicherweise haben Sie identische Tabellennamen in derselben Region, wenn sich die Tabellen in separaten Datenbanken befinden. Beim Erstellen der Tabelle müssen Sie den Namen der Tabelle und der Datenbank sowie die Aufbewahrungseigenschaften angeben. Hier gelten Servicekontingente. Einzelheiten finden Sie unter Codebeispiel.

Anforderungssyntax

```
{
   "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"
```

```
        }
    ]
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

DatabaseName

Der Name der Timestream-Datenbank.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 256 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

MagneticStoreWriteProperties

Enthält Eigenschaften, die beim Aktivieren von Magnetspeicher-Schreibvorgängen in der Tabelle festgelegt werden können.

Typ: MagneticStoreWriteProperties Objekt

Erforderlich: Nein

RetentionProperties

Die Dauer, für die Ihre Zeitreihendaten im Speicher und im Magnetspeicher gespeichert werden müssen.

Typ: RetentionProperties Objekt

Erforderlich: Nein

Schema

Das Schema der Tabelle.

Typ: Schema Objekt

Erforderlich: Nein

TableName

Der Name der Timestream-Tabelle.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 256 Zeichen.

Pattern: `[a-zA-Z0-9_.-]+`

Erforderlich: Ja

Tags

Eine Liste von Schlüssel-Wert-Paaren zur Bezeichnung der Tabelle.

Typ: Array von Tag-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 200 Elemente.

Erforderlich: Nein

Antwortsyntax

```
{
   "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"
   }
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Dienst im JSON Format zurückgegeben.

Table

Die neu erstellte Timestream-Tabelle.

Typ: Table Objekt

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion auszuführen.

HTTPStatuscode: 400

ConflictException

Timestream konnte diese Anfrage nicht verarbeiten, da sie eine Ressource enthält, die bereits vorhanden ist.

HTTPStatuscode: 400

InternalServerException

Timestream konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 500

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ResourceNotFoundException

Der Vorgang hat versucht, auf eine nicht vorhandene Ressource zuzugreifen. Die Ressource wurde möglicherweise nicht richtig angegeben, oder ihr Status ist nicht ACTIVE korrekt.

HTTPStatuscode: 400

ServiceQuotaExceededException

Das Instanzkontingent der Ressource für dieses Konto wurde überschritten.

HTTPStatuscode: 400

ThrottlingException

Ein Benutzer hat zu viele Anfragen gestellt und die Servicekontingente überschritten. Die Anforderung wurde gedrosselt.

HTTPStatuscode: 400

ValidationException

Eine ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

DeleteDatabase
Bedienung: Amazon Timestream Write

Löscht eine angegebene Timestream-Datenbank. Dieser Vorgang kann nicht rückgängig gemacht werden. Nach dem Löschen einer Datenbank können die Zeitreihendaten aus ihren Tabellen nicht wiederhergestellt werden.

> **Note**
>
> Alle Tabellen in der Datenbank müssen zuerst gelöscht werden, andernfalls wird ein ValidationException Fehler ausgegeben.
> Aufgrund der Beschaffenheit verteilter Wiederholungsversuche kann der Vorgang entweder erfolgreich oder fehlerhaft sein. ResourceNotFoundException Kunden sollten sie als gleichwertig betrachten.

Einzelheiten finden Sie unter Codebeispiel.

Anforderungssyntax

```
{
   "DatabaseName": "string"
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

DatabaseName

Der Name der Timestream-Datenbank, die gelöscht werden soll.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort mit einem leeren HTTP Text zurück.

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion auszuführen.

HTTPStatuscode: 400

InternalServerException

Timestream konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 500

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ResourceNotFoundException

Der Vorgang hat versucht, auf eine nicht existierende Ressource zuzugreifen. Die Ressource wurde möglicherweise nicht richtig angegeben, oder ihr Status ist nicht ACTIVE korrekt.

HTTPStatuscode: 400

ThrottlingException

Ein Benutzer hat zu viele Anfragen gestellt und sie haben die Servicekontingenten überschritten. Die Anforderung wurde gedrosselt.

HTTPStatuscode: 400

ValidationException

Eine ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

DeleteTable
Bedienung: Amazon Timestream Write

Löscht eine angegebene Timestream-Tabelle. Dieser Vorgang kann nicht rückgängig gemacht werden. Nach dem Löschen einer Timestream-Datenbanktabelle können die in der Tabelle gespeicherten Zeitreihendaten nicht wiederhergestellt werden.

> **Note**
>
> Aufgrund der Beschaffenheit verteilter Wiederholungsversuche kann der Vorgang entweder einen Erfolg oder einen Fehler zurückgeben. ResourceNotFoundException Kunden sollten sie als gleichwertig betrachten.

Einzelheiten finden Sie unter Codebeispiel.

Anforderungssyntax

```
{
   "DatabaseName": "string",
   "TableName": "string"
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

DatabaseName

Der Name der Datenbank, in der die Timestream-Datenbank gelöscht werden soll.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Ja

TableName

Der Name der Timestream-Tabelle, die gelöscht werden soll.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort mit einem leeren HTTP Text zurück.

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion auszuführen.

HTTPStatuscode: 400

InternalServerException

Timestream konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 500

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ResourceNotFoundException

Der Vorgang hat versucht, auf eine nicht existierende Ressource zuzugreifen. Die Ressource wurde möglicherweise nicht richtig angegeben, oder ihr Status ist nicht ACTIVE korrekt.

HTTPStatuscode: 400

ThrottlingException

Ein Benutzer hat zu viele Anfragen gestellt und sie haben die Servicekontingenten überschritten. Die Anforderung wurde gedrosselt.

HTTPStatuscode: 400

ValidationException

Eine ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

DescribeBatchLoadTask
Bedienung: Amazon Timestream Write

Gibt Informationen über die Batch-Load-Aufgabe zurück, einschließlich Konfigurationen, Zuordnungen, Fortschritt und anderer Details. Hier gelten Servicekontingente. Einzelheiten finden Sie unter Codebeispiel.

Anforderungssyntax

```
{
   "TaskId": "string"
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anforderung akzeptiert die folgenden Daten im JSON Format.

TaskId

Die ID der Batch-Load-Task.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 32 Zeichen.

Pattern: `[A-Z0-9]+`

Erforderlich: Ja

Antwortsyntax

```
{
   "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"
   }
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Dienst im JSON Format zurückgegeben.

BatchLoadTaskDescription

Beschreibung der Batch-Load-Aufgabe.

Typ: BatchLoadTaskDescription Objekt

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion auszuführen.

HTTPStatuscode: 400

InternalServerException

Timestream konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 500

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ResourceNotFoundException

Der Vorgang hat versucht, auf eine nicht existierende Ressource zuzugreifen. Die Ressource wurde möglicherweise nicht richtig angegeben, oder ihr Status ist nicht ACTIVE korrekt.

HTTPStatuscode: 400

ThrottlingException

Ein Benutzer hat zu viele Anfragen gestellt und sie haben die Servicekontingenten überschritten. Die Anforderung wurde gedrosselt.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle

- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

DescribeDatabase

Bedienung: Amazon Timestream Write

Gibt Informationen über die Datenbank zurück, einschließlich des Datenbanknamens, der Uhrzeit, zu der die Datenbank erstellt wurde, und der Gesamtzahl der in der Datenbank gefundenen Tabellen. Hier gelten Servicekontingente. Einzelheiten finden Sie unter Codebeispiel.

Anforderungssyntax

```
{
   "DatabaseName": "string"
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

DatabaseName

Der Name der Timestream-Datenbank.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Ja

Antwortsyntax

```
{
   "Database": {
      "Arn": "string",
      "CreationTime": number,
      "DatabaseName": "string",
      "KmsKeyId": "string",
      "LastUpdatedTime": number,
      "TableCount": number
   }
```

```
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Dienst im JSON Format zurückgegeben.

Database

Der Name der Timestream-Tabelle.

Typ: Database Objekt

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion auszuführen.

HTTPStatuscode: 400

InternalServerException

Timestream konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 500

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ResourceNotFoundException

Der Vorgang hat versucht, auf eine nicht vorhandene Ressource zuzugreifen. Die Ressource wurde möglicherweise nicht richtig angegeben, oder ihr Status ist nicht ACTIVE korrekt.

HTTPStatuscode: 400

ThrottlingException

Ein Benutzer hat zu viele Anfragen gestellt und sie haben die Servicekontingenten überschritten. Die Anforderung wurde gedrosselt.

HTTPStatuscode: 400

ValidationException

Eine ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

DescribeEndpoints
Bedienung: Amazon Timestream Write

Gibt eine Liste der verfügbaren Endpunkte zurück, für die API Timestream-Aufrufe ausgeführt werden können. Dieser API Vorgang ist sowohl über Write als auch über Query verfügbar. APIs

Da Timestream so konzipiert SDKs sind, dass sie transparent mit der Architektur des Dienstes zusammenarbeiten, einschließlich der Verwaltung und Zuordnung der Dienstendpunkte, empfehlen wir Ihnen, diesen API Vorgang nur zu verwenden, wenn:

- Sie verwenden VPCEndpoints () mit Timestream AWS PrivateLink
- Ihre Anwendung verwendet eine Programmiersprache, die noch nicht unterstützt wird SDK
- Sie benötigen eine bessere Kontrolle über die clientseitige Implementierung

Ausführliche Informationen darüber, wie und wann Sie es verwenden und implementieren sollten DescribeEndpoints, finden Sie unter The Endpoint Discovery Pattern.

Antwortsyntax

```
{
   "Endpoints": [
      {
         "Address": "string",
         "CachePeriodInMinutes": number
      }
   ]
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Dienst im JSON Format zurückgegeben.

Endpoints

Ein `Endpoints` Objekt wird zurückgegeben, wenn eine `DescribeEndpoints` Anfrage gestellt wird.

Typ: Array von Endpoint-Objekten

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

InternalServerException

Timestream konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 500

ThrottlingException

Ein Benutzer hat zu viele Anfragen gestellt und die Servicekontingenten überschritten. Die Anforderung wurde gedrosselt.

HTTPStatuscode: 400

ValidationException

Eine ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

DescribeTable

Bedienung: Amazon Timestream Write

Gibt Informationen über die Tabelle zurück, einschließlich des Tabellennamens, des Datenbanknamens, der Aufbewahrungsdauer des Speicherspeichers und des Magnetspeichers. Hier gelten Servicekontingente. Einzelheiten finden Sie unter Codebeispiel.

Anforderungssyntax

```
{
   "DatabaseName": "string",
   "TableName": "string"
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anforderung akzeptiert die folgenden Daten im JSON Format.

DatabaseName

Der Name der Timestream-Datenbank.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Ja

TableName

Der Name der Timestream-Tabelle.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Ja

Antwortsyntax

```
{
```

```
   "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"
   }
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Dienst im JSON Format zurückgegeben.

Table

Die Timestream-Tabelle.

Typ: Table Objekt

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion auszuführen.

HTTPStatuscode: 400

InternalServerException

Timestream konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 500

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ResourceNotFoundException

Der Vorgang hat versucht, auf eine nicht vorhandene Ressource zuzugreifen. Die Ressource wurde möglicherweise nicht richtig angegeben, oder ihr Status ist nicht ACTIVE korrekt.

HTTPStatuscode: 400

ThrottlingException

Ein Benutzer hat zu viele Anfragen gestellt und sie haben die Servicekontingenten überschritten. Die Anforderung wurde gedrosselt.

HTTPStatuscode: 400

ValidationException

Eine ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

ListBatchLoadTasks
Bedienung: Amazon Timestream Write

Stellt eine Liste der Batch-Load-Aufgaben zusammen mit dem Namen, dem Status, wann die Aufgabe fortgesetzt werden kann, und anderen Details bereit. Einzelheiten finden Sie unter Codebeispiel.

Anforderungssyntax

```
{
   "MaxResults": number,
   "NextToken": "string",
   "TaskStatus": "string"
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anforderung akzeptiert die folgenden Daten im JSON Format.

MaxResults

Die Gesamtzahl der Elemente, die in der Ausgabe zurückgegeben werden sollen. Wenn die Gesamtzahl der verfügbaren Elemente den angegebenen Wert übersteigt, NextToken wird in der Ausgabe ein Wert bereitgestellt. Um die Paginierung fortzusetzen, geben Sie den NextToken Wert als Argument für einen nachfolgenden API Aufruf an.

Typ: Ganzzahl

Gültiger Bereich: Mindestwert 1. Maximalwert 100.

Erforderlich: Nein

NextToken

Ein Token für den Beginn der Seitennummerierung. Dies ist die Antwort NextToken aus einer zuvor gekürzten Antwort.

Typ: Zeichenfolge

Erforderlich: Nein

TaskStatus

Status der Batch-Load-Aufgabe.

Typ: Zeichenfolge

Zulässige Werte: `CREATED | IN_PROGRESS | FAILED | SUCCEEDED | PROGRESS_STOPPED | PENDING_RESUME`

Erforderlich: Nein

Antwortsyntax

```
{
   "BatchLoadTasks": [
      {
         "CreationTime": number,
         "DatabaseName": "string",
         "LastUpdatedTime": number,
         "ResumableUntil": number,
         "TableName": "string",
         "TaskId": "string",
         "TaskStatus": "string"
      }
   ],
   "NextToken": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Dienst im JSON Format zurückgegeben.

BatchLoadTasks

Eine Liste mit Details zu Batch-Load-Aufgaben.

Typ: Array von BatchLoadTask-Objekten

NextToken

Ein Token für den Beginn der Seitennummerierung. Geben Sie den nächsten an ListBatchLoadTasksRequest.

Typ: Zeichenfolge

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion auszuführen.

HTTPStatuscode: 400

InternalServerException

Timestream konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 500

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ThrottlingException

Ein Benutzer hat zu viele Anfragen gestellt und sie haben die Servicekontingenten überschritten. Die Anforderung wurde gedrosselt.

HTTPStatuscode: 400

ValidationException

Eine ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

ListDatabases
Bedienung: Amazon Timestream Write

Gibt eine Liste Ihrer Timestream-Datenbanken zurück. Hier gelten Servicekontingente. Einzelheiten finden Sie unter Codebeispiel.

Anforderungssyntax

```
{
   "MaxResults": number,
   "NextToken": "string"
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

MaxResults

Die Gesamtzahl der Elemente, die in der Ausgabe zurückgegeben werden sollen. Wenn die Gesamtzahl der verfügbaren Elemente den angegebenen Wert übersteigt, NextToken wird in der Ausgabe ein Wert bereitgestellt. Um die Paginierung fortzusetzen, geben Sie den NextToken Wert als Argument für einen nachfolgenden API Aufruf an.

Typ: Ganzzahl

Gültiger Bereich: Mindestwert 1. Der Maximalwert ist 20.

Erforderlich: Nein

NextToken

Das Paginierungstoken. Um die Paginierung fortzusetzen, geben Sie den NextToken Wert als Argument für einen nachfolgenden API Aufruf an.

Typ: Zeichenfolge

Erforderlich: Nein

Antwortsyntax

```
{
   "Databases": [
      {
         "Arn": "string",
         "CreationTime": number,
         "DatabaseName": "string",
         "KmsKeyId": "string",
         "LastUpdatedTime": number,
         "TableCount": number
      }
   ],
   "NextToken": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Dienst im JSON Format zurückgegeben.

Databases

Eine Liste von Datenbanknamen.

Typ: Array von Database-Objekten

NextToken

Das Paginierungstoken. Dieser Parameter wird zurückgegeben, wenn die Antwort gekürzt wird.

Typ: Zeichenfolge

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion auszuführen.

HTTPStatuscode: 400

InternalServerException

Timestream konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 500

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ThrottlingException

Ein Benutzer hat zu viele Anfragen gestellt und sie haben die Servicekontingenten überschritten. Die Anforderung wurde gedrosselt.

HTTPStatuscode: 400

ValidationException

Eine ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python

- AWS SDKfür Ruby V3

ListTables

Bedienung: Amazon Timestream Write

Stellt eine Liste von Tabellen zusammen mit dem Namen, dem Status und den Aufbewahrungseigenschaften jeder Tabelle bereit. Einzelheiten finden Sie unter Codebeispiel.

Anforderungssyntax

```
{
   "DatabaseName": "string",
   "MaxResults": number,
   "NextToken": "string"
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anforderung akzeptiert die folgenden Daten im JSON Format.

DatabaseName

Der Name der Timestream-Datenbank.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Nein

MaxResults

Die Gesamtzahl der Elemente, die in der Ausgabe zurückgegeben werden sollen. Wenn die Gesamtzahl der verfügbaren Elemente den angegebenen Wert übersteigt, NextToken wird in der Ausgabe ein Wert bereitgestellt. Um die Paginierung fortzusetzen, geben Sie den NextToken Wert als Argument für einen nachfolgenden API Aufruf an.

Typ: Ganzzahl

Gültiger Bereich: Mindestwert 1. Der Maximalwert ist 20.

Erforderlich: Nein

NextToken

Das Paginierungstoken. Um die Paginierung fortzusetzen, geben Sie den NextToken Wert als Argument für einen nachfolgenden API Aufruf an.

Typ: Zeichenfolge

Erforderlich: Nein

Antwortsyntax

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

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Dienst im JSON Format zurückgegeben.

NextToken

Ein Token für den Beginn der Seitennummerierung. Dies ist die Antwort NextToken aus einer zuvor gekürzten Antwort.

Typ: Zeichenfolge

Tables

Eine Liste von Tabellen.

Typ: Array von Table-Objekten

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion auszuführen.

HTTPStatuscode: 400

InternalServerException

Timestream konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 500

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ResourceNotFoundException

Der Vorgang hat versucht, auf eine nicht existierende Ressource zuzugreifen. Die Ressource wurde möglicherweise nicht richtig angegeben, oder ihr Status ist nicht ACTIVE korrekt.

HTTPStatuscode: 400

ThrottlingException

Ein Benutzer hat zu viele Anfragen gestellt und sie haben die Servicekontingenten überschritten. Die Anforderung wurde gedrosselt.

HTTPStatuscode: 400

ValidationException

Eine ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

ListTagsForResource
Bedienung: Amazon Timestream Write

Listet alle Tags auf einer Timestream-Ressource auf.

Anforderungssyntax

```
{
   "ResourceARN": "string"
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

ResourceARN

Die Timestream-Ressource mit den Tags, die aufgelistet werden sollen. Dieser Wert ist ein Amazon-Ressourcenname (ARN).

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Die maximale Länge beträgt 1011.

Erforderlich: Ja

Antwortsyntax

```
{
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Dienst im JSON Format zurückgegeben.

Tags

Die aktuell mit der Timestream-Ressource verknüpften Tags.

Typ: Array von Tag-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 200 Elemente.

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ResourceNotFoundException

Der Vorgang hat versucht, auf eine nicht existierende Ressource zuzugreifen. Die Ressource wurde möglicherweise nicht richtig angegeben, oder ihr Status ist nicht ACTIVE korrekt.

HTTPStatuscode: 400

ThrottlingException

Ein Benutzer hat zu viele Anfragen gestellt und sie haben die Servicekontingenten überschritten. Die Anforderung wurde gedrosselt.

HTTPStatuscode: 400

ValidationException

Eine ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

ResumeBatchLoadTask
Bedienung: Amazon Timestream Write

Anforderungssyntax

```
{
   "TaskId": "string"
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

TaskId

Die ID der Batch-Ladeaufgabe, die fortgesetzt werden soll.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 32 Zeichen.

Pattern: `[A-Z0-9]+`

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort mit einem leeren HTTP Text zurück.

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion auszuführen.

HTTPStatuscode: 400

InternalServerException

Timestream konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 500

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ResourceNotFoundException

Der Vorgang hat versucht, auf eine nicht existierende Ressource zuzugreifen. Die Ressource wurde möglicherweise nicht richtig angegeben, oder ihr Status ist nicht ACTIVE korrekt.

HTTPStatuscode: 400

ThrottlingException

Ein Benutzer hat zu viele Anfragen gestellt und sie haben die Servicekontingenten überschritten. Die Anforderung wurde gedrosselt.

HTTPStatuscode: 400

ValidationException

Eine ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2

- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

TagResource
Bedienung: Amazon Timestream Write

Ordnet einer Timestream-Ressource eine Reihe von Tags zu. Anschließend können Sie diese benutzerdefinierten Tags aktivieren, sodass sie in der Billing and Cost Management-Konsole zur Nachverfolgung der Kostenzuweisung angezeigt werden.

Anforderungssyntax

```
{
   "ResourceARN": "string",
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

ResourceARN

Identifiziert die Timestream-Ressource, zu der Tags hinzugefügt werden sollen. Dieser Wert ist ein Amazon-Ressourcenname (ARN).

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Die maximale Länge beträgt 1011.

Erforderlich: Ja

Tags

Die Tags, die der Timestream-Ressource zugewiesen werden sollen.

Typ: Array von Tag-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 200 Elemente.

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort mit einem leeren HTTP Text zurück.

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ResourceNotFoundException

Der Vorgang hat versucht, auf eine nicht existierende Ressource zuzugreifen. Die Ressource wurde möglicherweise nicht richtig angegeben, oder ihr Status ist nicht ACTIVE korrekt.

HTTPStatuscode: 400

ServiceQuotaExceededException

Das Instanzkontingent der Ressource für dieses Konto wurde überschritten.

HTTPStatuscode: 400

ThrottlingException

Ein Benutzer hat zu viele Anfragen gestellt und sie haben die Servicekontingenten überschritten. Die Anforderung wurde gedrosselt.

HTTPStatuscode: 400

ValidationException

Eine ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

UntagResource
Bedienung: Amazon Timestream Write

Entfernt die Zuordnung von Tags aus einer Timestream-Ressource.

Anforderungssyntax

```
{
   "ResourceARN": "string",
   "TagKeys": [ "string" ]
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

ResourceARN

Die Timestream-Ressource, aus der die Tags entfernt werden. Dieser Wert ist ein Amazon-Ressourcenname (ARN).

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Die maximale Länge beträgt 1011.

Erforderlich: Ja

TagKeys

Eine Liste von Tag-Schlüsseln. Bestehende Tags der Ressource, deren Schlüssel Mitglieder dieser Liste sind, werden aus der Timestream-Ressource entfernt.

Typ: Zeichenfolge-Array

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 200 Elemente.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort mit einem leeren HTTP Text zurück.

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ResourceNotFoundException

Der Vorgang hat versucht, auf eine nicht existierende Ressource zuzugreifen. Die Ressource wurde möglicherweise nicht richtig angegeben, oder ihr Status ist nicht ACTIVE korrekt.

HTTPStatuscode: 400

ServiceQuotaExceededException

Das Instanzkontingent der Ressource für dieses Konto wurde überschritten.

HTTPStatuscode: 400

ThrottlingException

Ein Benutzer hat zu viele Anfragen gestellt und sie haben die Servicekontingenten überschritten. Die Anforderung wurde gedrosselt.

HTTPStatuscode: 400

ValidationException

Eine ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

UpdateDatabase
Bedienung: Amazon Timestream Write

Ändert den AWS KMS Schlüssel für eine bestehende Datenbank. Beim Aktualisieren der Datenbank müssen Sie den Datenbanknamen und die Kennung des neuen AWS KMS Schlüssels angeben, der verwendet werden soll (`KmsKeyId`). Bei gleichzeitigen `UpdateDatabase` Anfragen gewinnt der erste Writer.

Einzelheiten finden Sie unter Codebeispiel.

Anforderungssyntax

```
{
   "DatabaseName": "string",
   "KmsKeyId": "string"
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

DatabaseName

Name der Datenbank.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Ja

KmsKeyId

Die Kennung des neuen AWS KMS Schlüssels (`KmsKeyId`), der zur Verschlüsselung der in der Datenbank gespeicherten Daten verwendet werden soll. Wenn der `KmsKeyId` aktuell in der Datenbank registrierte Wert mit dem `KmsKeyId` in der Anfrage übereinstimmt, erfolgt keine Aktualisierung.

Sie können das `KmsKeyId` mit einer der folgenden Optionen angeben:

- Schlüssel-ID: `1234abcd-12ab-34cd-56ef-1234567890ab`
- SchlüsselARN: `arn:aws:kms:us-east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Alias-Name: `alias/ExampleAlias`
- AliasARN: `arn:aws:kms:us-east-1:111122223333:alias/ExampleAlias`

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Ja

Antwortsyntax

```
{
   "Database": {
      "Arn": "string",
      "CreationTime": number,
      "DatabaseName": "string",
      "KmsKeyId": "string",
      "LastUpdatedTime": number,
      "TableCount": number
   }
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Dienst im JSON Format zurückgegeben.

Database

Ein Container der obersten Ebene für eine Tabelle. Datenbanken und Tabellen sind die grundlegenden Verwaltungskonzepte in Amazon Timestream. Alle Tabellen in einer Datenbank sind mit demselben AWS KMS Schlüssel verschlüsselt.

Typ: Database Objekt

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion auszuführen.

HTTPStatuscode: 400

InternalServerException

Timestream konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 500

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ResourceNotFoundException

Der Vorgang hat versucht, auf eine nicht vorhandene Ressource zuzugreifen. Die Ressource wurde möglicherweise nicht richtig angegeben, oder ihr Status ist nicht ACTIVE korrekt.

HTTPStatuscode: 400

ServiceQuotaExceededException

Das Instanzkontingent der Ressource für dieses Konto wurde überschritten.

HTTPStatuscode: 400

ThrottlingException

Ein Benutzer hat zu viele Anfragen gestellt und sie haben die Servicekontingenten überschritten. Die Anforderung wurde gedrosselt.

HTTPStatuscode: 400

ValidationException

Eine ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

UpdateTable

Bedienung: Amazon Timestream Write

Ändert die Aufbewahrungsdauer des Speicherspeichers und des Magnetspeichers für Ihre Timestream-Tabelle. Beachten Sie, dass die Änderung der Aufbewahrungsdauer sofort wirksam wird. Wenn beispielsweise die Aufbewahrungsdauer des Speicherspeichers ursprünglich auf 2 Stunden festgelegt und dann auf 24 Stunden geändert wurde, kann der Speicherspeicher Daten für 24 Stunden speichern, wird aber 22 Stunden nach dieser Änderung mit Daten für 24 Stunden gefüllt. Timestream ruft keine Daten aus dem Magnetspeicher ab, um den Speicherspeicher zu füllen.

Einzelheiten finden Sie unter Codebeispiel.

Anforderungssyntax

```
{
   "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"
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anforderung akzeptiert die folgenden Daten im JSON Format.

DatabaseName

Der Name der Timestream-Datenbank.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Ja

MagneticStoreWriteProperties

Enthält Eigenschaften, die beim Aktivieren von Magnetspeicher-Schreibvorgängen in der Tabelle festgelegt werden können.

Typ: MagneticStoreWriteProperties Objekt

Erforderlich: Nein

RetentionProperties

Die Aufbewahrungsdauer des Speicherspeichers und des Magnetspeichers.

Typ: RetentionProperties Objekt

Erforderlich: Nein

Schema

Das Schema der Tabelle.

Typ: Schema Objekt

Erforderlich: Nein

TableName

Der Name der Timestream-Tabelle.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Ja

Antwortsyntax

```
{
   "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"
   }
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Dienst im JSON Format zurückgegeben.

Table

Die aktualisierte Timestream-Tabelle.

Typ: Table Objekt

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion auszuführen.

HTTPStatuscode: 400

InternalServerException

Timestream konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 500

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ResourceNotFoundException

Der Vorgang hat versucht, auf eine nicht vorhandene Ressource zuzugreifen. Die Ressource wurde möglicherweise nicht richtig angegeben, oder ihr Status ist nicht ACTIVE korrekt.

HTTPStatuscode: 400

ThrottlingException

Ein Benutzer hat zu viele Anfragen gestellt und sie haben die Servicekontingenten überschritten. Die Anforderung wurde gedrosselt.

HTTPStatuscode: 400

ValidationException

Eine ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

WriteRecords
Bedienung: Amazon Timestream Write

Ermöglicht es Ihnen, Ihre Zeitreihendaten in Timestream zu schreiben. Sie können einen einzelnen Datenpunkt oder einen Stapel von Datenpunkten angeben, die in das System eingefügt werden sollen. Timestream bietet Ihnen ein flexibles Schema, das die Spaltennamen und Datentypen für Ihre Timestream-Tabellen auto anhand der Dimensionsnamen und Datentypen der Datenpunkte erkennt, die Sie beim Aufrufen von Schreibvorgängen in die Datenbank angeben.

Timestream unterstützt die Lesesemantik mit eventueller Konsistenz. Das bedeutet, dass, wenn Sie Daten unmittelbar nach dem Schreiben eines Datenstapels in Timestream abfragen, die Abfrageergebnisse möglicherweise nicht die Ergebnisse eines kürzlich abgeschlossenen Schreibvorgangs widerspiegeln. Die Ergebnisse können auch einige veraltete Daten enthalten. Wenn Sie die Abfrageanforderung nach kurzer Zeit wiederholen, sollten die Ergebnisse die neuesten Daten zurückgeben. Hier gelten Servicekontingente.

Einzelheiten finden Sie unter Codebeispiel.

Ärgernisse

Sie können den `Version` Parameter in einer `WriteRecords` Anfrage verwenden, um Datenpunkte zu aktualisieren. Timestream verfolgt mit jedem Datensatz eine Versionsnummer. `Version`wird standardmäßig verwendet, `1` wenn sie nicht für den Datensatz in der Anfrage angegeben ist. Timestream aktualisiert den Messwert eines vorhandenen Datensatzes zusammen mit dem Wert, `Version` wenn er eine Schreibanforderung erhält, mit einer höheren `Version` Zahl für diesen Datensatz. Wenn es eine Aktualisierungsanfrage erhält, bei der der Messwert mit dem des vorhandenen Datensatzes identisch ist, aktualisiert Timestream trotzdem`Version`, wenn er größer als der bestehende Wert von ist. `Version` Sie können einen Datenpunkt beliebig oft aktualisieren, solange der Wert von `Version` kontinuierlich steigt.

Nehmen wir beispielsweise an, Sie schreiben einen neuen Datensatz, ohne dies `Version` in der Anfrage anzugeben. Timestream speichert diesen Datensatz und ist `Version` auf `1` eingestellt. Nehmen wir nun an, Sie versuchen, diesen Datensatz mit einer `WriteRecords` Anforderung für denselben Datensatz mit einem anderen Messwert zu aktualisieren, geben `Version` ihn aber wie zuvor nicht an. In diesem Fall lehnt Timestream diese Aktualisierung mit einem ab, `RejectedRecordsException` da die Version des aktualisierten Datensatzes nicht höher ist als der bestehende Wert von Version.

Wenn Sie die Aktualisierungsanforderung jedoch erneut mit der `Version` Einstellung auf senden würden2, würde Timestream den Wert des Datensatzes erfolgreich aktualisieren, und der `Version`

wäre auf gesetzt. 2 Nehmen wir als Nächstes an, Sie haben eine `WriteRecords` Anfrage mit demselben Datensatz und einem identischen Messwert gesendet, aber mit der `Version` Einstellung auf. 3 In diesem Fall würde Timestream nur `Version` auf 3 aktualisieren. Bei allen weiteren Updates müsste eine Versionsnummer gesendet werden, die größer als ist3, oder die Aktualisierungsanforderungen müssten eine `RejectedRecordsException` erhalten.

Anforderungssyntax

```
{
   "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"
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

CommonAttributes

Ein Datensatz, der die gemeinsamen Maß-, Dimensions-, Zeit- und Versionsattribute enthält, die von allen Datensätzen in der Anfrage gemeinsam genutzt werden. Die angegebenen Maß- und Dimensionsattribute werden mit den Kennzahl- und Dimensionsattributen im Datensatzobjekt zusammengeführt, wenn die Daten in Timestream geschrieben werden. Dimensionen dürfen sich nicht überlappen, sonst `ValidationException` wird eine ausgelöst. Mit anderen Worten, ein Datensatz muss Dimensionen mit eindeutigen Namen enthalten.

Typ: Record Objekt

Erforderlich: Nein

DatabaseName

Der Name der Timestream-Datenbank.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Ja

Records

Eine Reihe von Datensätzen, die die eindeutigen Maß-, Dimension-, Zeit- und Versionsattribute für jeden Zeitreihendatenpunkt enthalten.

Typ: Array von Record-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 1 Element. Die maximale Anzahl beträgt 100 Elemente.

Erforderlich: Ja

TableName

Der Name der Timestream-Tabelle.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Ja

Antwortsyntax

```
{
   "RecordsIngested": {
      "MagneticStore": number,
      "MemoryStore": number,
      "Total": number
   }
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Dienst im JSON Format zurückgegeben.

RecordsIngested

Informationen zu den Datensätzen, die im Rahmen dieser Anfrage aufgenommen wurden.

Typ: RecordsIngested Objekt

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion durchzuführen.

HTTPStatuscode: 400

InternalServerException

Timestream konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 500

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

RejectedRecordsException

WriteRecords würde diese Ausnahme in den folgenden Fällen auslösen:

- Datensätze mit doppelten Daten, bei denen es mehrere Datensätze mit denselben Dimensionen, Zeitstempeln und Kennzahlnamen gibt, aber:
  - Die Kennzahlwerte sind unterschiedlich
  - Version ist in der Anfrage nicht vorhanden, oder der Wert der Version im neuen Datensatz ist gleich oder niedriger als der vorhandene Wert

  Wenn Timestream in diesem Fall Daten ablehnt, gibt das `ExistingVersion` Feld in der `RejectedRecords` Antwort die Version des aktuellen Datensatzes an. Um eine Aktualisierung zu erzwingen, können Sie die Anfrage erneut senden, wobei eine Version für den Datensatz auf einen Wert gesetzt ist, der größer als der ist. `ExistingVersion`
- Datensätze mit Zeitstempeln, die außerhalb der Aufbewahrungsdauer des Speicherspeichers liegen.

- Datensätze mit Dimensionen oder Kennzahlen, die die von Timestream definierten Grenzwerte überschreiten.

Weitere Informationen finden Sie unter Kontingente im Amazon Timestream Developer Guide.

HTTPStatuscode: 400

ResourceNotFoundException

Der Vorgang hat versucht, auf eine nicht vorhandene Ressource zuzugreifen. Die Ressource wurde möglicherweise nicht richtig angegeben, oder ihr Status ist nicht ACTIVE korrekt.

HTTPStatuscode: 400

ThrottlingException

Ein Benutzer hat zu viele Anfragen gestellt und die Servicekontingente überschritten. Die Anforderung wurde gedrosselt.

HTTPStatuscode: 400

ValidationException

Eine ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

Amazon Timestream Timestream-Abfrage

Die folgenden Aktionen werden von Amazon Timestream Query unterstützt:

- CancelQuery
- CreateScheduledQuery
- DeleteScheduledQuery
- DescribeAccountSettings
- DescribeEndpoints
- DescribeScheduledQuery
- ExecuteScheduledQuery
- ListScheduledQueries
- ListTagsForResource
- PrepareQuery
- Query
- TagResource
- UntagResource
- UpdateAccountSettings
- UpdateScheduledQuery

CancelQuery
Bedienung: Amazon Timestream Query

Bricht eine Anfrage ab, die gestellt wurde. Eine Stornierung ist nur möglich, wenn die Abfrage nicht vollständig ausgeführt wurde, bevor die Stornierungsanforderung gestellt wurde. Da es sich bei der Stornierung um eine idempotente Operation handelt, geben nachfolgende Stornierungsanforderungen a zurück`CancellationMessage`, was darauf hinweist, dass die Abfrage bereits abgebrochen wurde. Einzelheiten finden Sie unter Codebeispiel.

Anforderungssyntax

```
{
   "QueryId": "string"
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

QueryId

Die ID der Abfrage, die storniert werden muss. `QueryID`wird als Teil des Abfrageergebnisses zurückgegeben.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `[a-zA-Z0-9]+`

Erforderlich: Ja

Antwortsyntax

```
{
   "CancellationMessage": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Dienst im JSON Format zurückgegeben.

CancellationMessage

A `CancellationMessage` wird zurückgegeben, wenn bereits eine `CancelQuery` Anforderung für die von `QueryId` angegebene Abfrage gestellt wurde.

Typ: Zeichenfolge

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion auszuführen.

HTTPStatuscode: 400

InternalServerException

Der Dienst konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 400

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ThrottlingException

Die Anforderung wurde aufgrund der Drosselung von Anforderungen abgelehnt.

HTTPStatuscode: 400

ValidationException

Ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

CreateScheduledQuery

Bedienung: Amazon Timestream Query

Erstellen Sie eine geplante Abfrage, die in Ihrem Namen nach dem konfigurierten Zeitplan ausgeführt wird. Timestream übernimmt die Ausführungsrolle, die als Teil des `ScheduledQueryExecutionRoleArn`-Parameters bereitgestellt wird, um die Abfrage auszuführen. Sie können den `NotificationConfiguration`-Parameter verwenden, um die Benachrichtigung für Ihre geplanten Abfragevorgänge zu konfigurieren.

Anforderungssyntax

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

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

ClientToken

Die Verwendung von a ClientToken macht den Aufruf CreateScheduledQuery idempotent, mit anderen Worten, wenn dieselbe Anfrage wiederholt gestellt wird, führt dies zu demselben Ergebnis. Das Stellen mehrerer identischer CreateScheduledQuery Anfragen hat den gleichen Effekt wie das Stellen einer einzelnen Anfrage.

- Wenn ohne a aufgerufen CreateScheduledQuery wird`ClientToken`, SDK generiert die Query in `ClientToken` Ihrem Namen eine.
- Nach 8 Stunden wird jede Anforderung mit dem gleichen `ClientToken` als neue Anforderung behandelt.

Typ: Zeichenfolge

Längenbeschränkungen: Mindestlänge von 32. Maximale Länge beträgt 128 Zeichen.

Erforderlich: Nein

ErrorReportConfiguration

Konfiguration für die Fehler-Berichterstellung. Fehlerberichte werden generiert, wenn beim Schreiben der Abfrage-Ergebnisse ein Problem auftritt.

Typ: ErrorReportConfiguration Objekt

Erforderlich: Ja

KmsKeyId

Der KMS Amazon-Schlüssel, der verwendet wird, um die geplante Abfrageressource im Ruhezustand zu verschlüsseln. Wenn der KMS Amazon-Schlüssel nicht angegeben ist, wird die geplante Abfrageressource mit einem Timestream-eigenen KMS Amazon-Schlüssel verschlüsselt. Um einen KMS Schlüssel anzugeben, verwenden Sie die Schlüssel-ID, den SchlüsselARN, den Aliasnamen oder den AliasARN. Wenn Sie einen Aliasnamen verwenden, stellen Sie dem Namen alias/ voran.

Wenn `SSE_KMS` als Verschlüsselungstyp ErrorReportConfiguration verwendet wird, KmsKeyId wird derselbe verwendet, um den Fehlerbericht im Ruhezustand zu verschlüsseln.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Nein

Name

Name der geplanten Abfrage.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Erforderlich: Ja

NotificationConfiguration

Benachrichtigungs-Konfiguration für eine geplante Abfrage. Timestream sendet eine Benachrichtigung, wenn ein Abfragelauf abgeschlossen ist, wenn der Status aktualisiert wird oder wenn Sie ihn löschen.

Typ: NotificationConfiguration Objekt

Erforderlich: Ja

QueryString

Die auszuführende Abfragezeichenfolge. Parameternamen können in der Abfragezeichenfolge @ Zeichen gefolgt von einer Kennung angegeben werden. Der benannte Parameter `@scheduled_runtime` ist reserviert und kann in der Abfrage verwendet werden, um den Zeitpunkt abzurufen, zu dem die Abfrage planmäßig ausgeführt werden soll.

Der anhand des ScheduleConfiguration Parameters berechnete Zeitstempel entspricht dem Wert von `@scheduled_runtime` paramater für jeden Abfragelauf. Stellen Sie sich zum Beispiel eine Instance einer geplanten Abfrage vor, die am 01.12.2021 00:00:00 ausgeführt wird. In dieser Instance wird der `@scheduled_runtime` Parameter beim Aufrufen der Abfrage mit dem Zeitstempel 2021-12-01 00:00:00 initialisiert.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge von 262144.

Erforderlich: Ja

ScheduleConfiguration

Die Zeitplankonfiguration für die Abfrage.

Typ: ScheduleConfiguration Objekt

Erforderlich: Ja

ScheduledQueryExecutionRoleArn

Die ARN für die IAM Rolle, die Timestream bei der Ausführung der geplanten Abfrage übernimmt.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Ja

Tags

Eine Liste von Schlüssel-Wert-Paaren zur Benennung der geplanten Abfrage.

Typ: Array von Tag-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 200 Elemente.

Erforderlich: Nein

TargetConfiguration

Konfiguration, die zum Schreiben des Ergebnisses einer Abfrage verwendet wird.

Typ: TargetConfiguration Objekt

Erforderlich: Nein

Antwortsyntax

```
{
   "Arn": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Dienst im JSON Format zurückgegeben.

Arn

ARNfür die erstellte geplante Abfrage.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion auszuführen.

HTTPStatuscode: 400

ConflictException

Die Ergebnisse für eine stornierte Abfrage konnten nicht abgefragt werden.

HTTPStatuscode: 400

InternalServerException

Der Dienst konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 400

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ServiceQuotaExceededException

Sie haben das Servicekontingent überschritten.

HTTPStatuscode: 400

ThrottlingException

Die Anforderung wurde aufgrund der Drosselung von Anforderungen abgelehnt.

HTTPStatuscode: 400

ValidationException

Ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

DeleteScheduledQuery
Bedienung: Amazon Timestream Query

Löscht eine angegebene geplante Abfrage. Dieser Vorgang kann nicht rückgängig gemacht werden.

Anforderungssyntax

```
{
   "ScheduledQueryArn": "string"
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

ScheduledQueryArn

Die ARN der geplanten Abfrage.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort mit einem leeren HTTP Text zurück.

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion auszuführen.

HTTPStatuscode: 400

InternalServerException

Der Dienst konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 400

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ResourceNotFoundException

Die angeforderte Ressource wurde nicht gefunden.

HTTPStatuscode: 400

ThrottlingException

Die Anforderung wurde aufgrund der Drosselung von Anforderungen abgelehnt.

HTTPStatuscode: 400

ValidationException

Ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3

- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

DescribeAccountSettings
Bedienung: Amazon Timestream Query

Beschreibt die Einstellungen für Ihr Konto, zu denen das Preismodell für Abfragen und der konfigurierte Höchstwert gehören, TCUs den der Dienst für Ihre Abfrage-Workloads verwenden kann.

Ihnen wird nur die Dauer der Recheneinheiten in Rechnung gestellt, die für Ihre Workloads verwendet werden.

Antwortsyntax

```
{
   "MaxQueryTCU": number,
   "QueryPricingModel": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine Antwort von HTTP 200 zurück.

Die folgenden Daten werden vom Dienst im JSON Format zurückgegeben.

MaxQueryTCU

Die maximale Anzahl von Timestream-Recheneinheiten (TCUs), die der Service zu einem beliebigen Zeitpunkt für die Bearbeitung Ihrer Abfragen verwendet.

Typ: Ganzzahl

QueryPricingModel

Das Preismodell für Abfragen in Ihrem Konto.

Typ: Zeichenfolge

Zulässige Werte: `BYTES_SCANNED | COMPUTE_UNITS`

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion durchzuführen.

HTTPStatuscode: 400

InternalServerException

Der Dienst konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 400

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ThrottlingException

Die Anforderung wurde aufgrund der Drosselung von Anforderungen abgelehnt.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

DescribeEndpoints
Bedienung: Amazon Timestream Query

DescribeEndpoints gibt eine Liste verfügbarer Endpunkte zurück, für die API Timestream-Aufrufe ausgeführt werden können. Dies API ist sowohl über Write als auch über Query verfügbar.

Da Timestream so konzipiert SDKs sind, dass sie auf transparente Weise mit der Architektur des Dienstes zusammenarbeiten, einschließlich der Verwaltung und Zuordnung der Dienstendpunkte, sollten Sie diese Option nur verwenden, wenn: API

- Sie verwenden VPCEndpoints () mit Timestream AWS PrivateLink
- Ihre Anwendung verwendet eine Programmiersprache, die noch nicht unterstützt wird SDK
- Sie benötigen eine bessere Kontrolle über die clientseitige Implementierung

Ausführliche Informationen darüber, wie und wann Sie es verwenden und implementieren sollten DescribeEndpoints, finden Sie unter The Endpoint Discovery Pattern.

Antwortsyntax

```
{
   "Endpoints": [
      {
         "Address": "string",
         "CachePeriodInMinutes": number
      }
   ]
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Dienst im JSON Format zurückgegeben.

Endpoints

Ein `Endpoints` Objekt wird zurückgegeben, wenn eine `DescribeEndpoints` Anfrage gestellt wird.

Typ: Array von Endpoint-Objekten

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

InternalServerException

Der Dienst konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 400

ThrottlingException

Die Anforderung wurde aufgrund der Drosselung von Anforderungen abgelehnt.

HTTPStatuscode: 400

ValidationException

Ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

DescribeScheduledQuery
Bedienung: Amazon Timestream Query

Stellt detaillierte Informationen zu einer geplanten Abfrage bereit.

Anforderungssyntax

```
{
   "ScheduledQueryArn": "string"
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

ScheduledQueryArn

Die ARN der geplanten Abfrage.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Ja

Antwortsyntax

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

```
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Dienst im JSON Format zurückgegeben.

ScheduledQuery

Die geplante Abfrage.

Typ: ScheduledQueryDescription Objekt

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion auszuführen.

HTTPStatuscode: 400

InternalServerException

Der Dienst konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 400

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ResourceNotFoundException

Die angeforderte Ressource wurde nicht gefunden.

HTTPStatuscode: 400

ThrottlingException

Die Anforderung wurde aufgrund der Drosselung von Anforderungen abgelehnt.

HTTPStatuscode: 400

ValidationException

Ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

ExecuteScheduledQuery
Bedienung: Amazon Timestream Query

Sie können dies verwendenAPI, um eine geplante Abfrage manuell auszuführen.

Wenn Sie diese Option aktiviert haben`QueryInsights`, werden API auch Einblicke und Metriken zu der Abfrage zurückgegeben, die Sie als Teil einer SNS Amazon-Benachrichtigung ausgeführt haben. `QueryInsights`shilft bei der Leistungsoptimierung Ihrer Abfrage. Weitere Informationen `QueryInsights` dazu finden Sie unter Verwenden von Abfrageerkenntnissen zur Optimierung von Abfragen in Amazon Timestream.

Anforderungssyntax

```
{
   "ClientToken": "string",
   "InvocationTime": number,
   "QueryInsights": {
      "Mode": "string"
   },
   "ScheduledQueryArn": "string"
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

ClientToken

Nicht verwendet

Typ: Zeichenfolge

Längenbeschränkungen: Mindestlänge von 32. Maximale Länge beträgt 128 Zeichen.

Erforderlich: Nein

InvocationTime

Der Zeitstempel inUTC. Die Abfrage wird so ausgeführt, als ob sie zu diesem Zeitstempel aufgerufen worden wäre.

Typ: Zeitstempel

Erforderlich: Ja

QueryInsights

Kapselt Einstellungen für die Aktivierung. `QueryInsights`

Wenn Sie Einblicke und Kennzahlen zu `QueryInsights` Rücksendungen als Teil der SNS Amazon-Benachrichtigung für die von Ihnen ausgeführte Abfrage aktivieren. Sie können `QueryInsights` es verwenden, um die Leistung und die Kosten Ihrer Abfrage zu optimieren.

Typ: ScheduledQueryInsights Objekt

Erforderlich: Nein

ScheduledQueryArn

ARNder geplanten Abfrage.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort mit einem leeren HTTP Text zurück.

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion auszuführen.

HTTPStatuscode: 400

InternalServerException

Der Dienst konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 400

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ResourceNotFoundException

Die angeforderte Ressource wurde nicht gefunden.

HTTPStatuscode: 400

ThrottlingException

Die Anforderung wurde aufgrund der Drosselung von Anforderungen abgelehnt.

HTTPStatuscode: 400

ValidationException

Ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Beispiele

Geplante Abfragebenachrichtigung für den CONTROL Modus ENABLED WITH _ RATE _ _

Das folgende Beispiel zeigt eine erfolgreiche Benachrichtigung über eine geplante Abfrage für den `ENABLED_WITH_RATE_CONTROL` Modus des `QueryInsights` Parameters.

```
"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
        }
    }
}
```

Benachrichtigung über eine geplante Abfrage für den DISABLED Modus

Das folgende Beispiel zeigt eine erfolgreiche Benachrichtigung über eine geplante Abfrage für den `DISABLED` Modus des `QueryInsights` Parameters.

```
"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
        }
    }
}
```

Fehlerbenachrichtigung für den CONTROL Modus ENABLED WITH _ RATE _ _

Das folgende Beispiel zeigt eine Benachrichtigung über eine fehlgeschlagene geplante Abfrage für den ENABLED_WITH_RATE_CONTROL Modus des `QueryInsights` Parameters.

```
"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"
    }
}
```

Meldung zur Benachrichtigung über einen Fehler für den DISABLED Modus

Das folgende Beispiel zeigt eine Benachrichtigung über eine fehlgeschlagene geplante Abfrage für den `DISABLED` Modus des `QueryInsights` Parameters.

```
"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"
    }
}
```

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieses Parameters API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

ListScheduledQueries
Bedienung: Amazon Timestream Query

Ruft eine Liste aller geplanten Abfragen im Amazon-Konto und in der Region des Anrufers ab. `ListScheduledQueries`ist letztendlich konsistent.

Anforderungssyntax

```
{
   "MaxResults": number,
   "NextToken": "string"
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

MaxResults

Die maximale Anzahl von Elementen, die in der Ausgabe zurückgegeben werden sollen. Wenn die Gesamtzahl der verfügbaren Elemente den angegebenen Wert überschreitet, `NextToken` wird in der Ausgabe ein Wert bereitgestellt. Um die Paginierung fortzusetzen, geben Sie den `NextToken` Wert als Argument für den nachfolgenden Aufruf von an an an an`ListScheduledQueriesRequest`.

Typ: Ganzzahl

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 1 000.

Erforderlich: Nein

NextToken

Ein Paginierungstoken, um die Paginierung fortzusetzen.

Typ: Zeichenfolge

Erforderlich: Nein

Antwortsyntax

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

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Dienst im JSON Format zurückgegeben.

NextToken

Ein Token für den Beginn der Seitennummerierung. Dies ist die Antwort NextToken aus einer zuvor gekürzten Antwort.

Typ: Zeichenfolge

ScheduledQueries

Eine Liste von geplanten Abfragen.

Typ: Array von ScheduledQuery-Objekten

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion auszuführen.

HTTPStatuscode: 400

InternalServerException

Der Dienst konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 400

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ThrottlingException

Die Anforderung wurde aufgrund der Drosselung von Anforderungen abgelehnt.

HTTPStatuscode: 400

ValidationException

Ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

ListTagsForResource
Bedienung: Amazon Timestream Query

Listet alle Tags auf einer Timestream-Abfrageressource auf.

Anforderungssyntax

```
{
   "MaxResults": number,
   "NextToken": "string",
   "ResourceARN": "string"
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

MaxResults

Die maximale Anzahl von Tags, die zurückgegeben werden sollen.

Typ: Ganzzahl

Gültiger Bereich: Mindestwert 1. Maximalwert 200.

Erforderlich: Nein

NextToken

Ein Paginierungstoken zur Wiederaufnahme der Paginierung.

Typ: Zeichenfolge

Erforderlich: Nein

ResourceARN

Die Timestream-Ressource mit den Tags, die aufgelistet werden sollen. Dieser Wert ist ein Amazon-Ressourcenname (ARN).

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Ja

Antwortsyntax

```
{
   "NextToken": "string",
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Service eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Dienst im JSON Format zurückgegeben.

NextToken

Ein Paginierungstoken, um die Paginierung mit einem nachfolgenden Aufruf von fortzusetzen. `ListTagsForResourceResponse`

Typ: Zeichenfolge

Tags

Die aktuell mit der Timestream-Ressource verknüpften Tags.

Typ: Array von Tag-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 200 Elemente.

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ResourceNotFoundException

Die angeforderte Ressource wurde nicht gefunden.

HTTPStatuscode: 400

ThrottlingException

Die Anforderung wurde aufgrund der Drosselung von Anforderungen abgelehnt.

HTTPStatuscode: 400

ValidationException

Ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

PrepareQuery
Bedienung: Amazon Timestream Query

Ein synchroner Vorgang, mit dem Sie eine Abfrage mit Parametern senden können, die von Timestream für die spätere Ausführung gespeichert werden sollen. Timestream unterstützt nur die Verwendung dieses Vorgangs mit `ValidateOnly` der Einstellung auf. `true`

Anforderungssyntax

```
{
   "QueryString": "string",
   "ValidateOnly": boolean
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

QueryString

Die Timestream-Abfragezeichenfolge, die Sie als vorbereitete Anweisung verwenden möchten. Parameternamen können in der Abfragezeichenfolge @ Zeichen gefolgt von einer Kennung angegeben werden.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge von 262144.

Erforderlich: Ja

ValidateOnly

Wenn Sie diesen Wert auf setzen`true`, überprüft Timestream nur, ob es sich bei der Abfragezeichenfolge um eine gültige Timestream-Abfrage handelt, und speichert die vorbereitete Abfrage nicht für die spätere Verwendung.

Typ: Boolesch

Erforderlich: Nein

Antwortsyntax

```
{
   "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"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Dienst im JSON Format zurückgegeben.

Columns

Eine Liste von SELECT Klauselspalten der übermittelten Abfragezeichenfolge.

Typ: Array von SelectColumn-Objekten

Parameters

Eine Liste von Parametern, die in der übermittelten Abfragezeichenfolge verwendet wurden.

Typ: Array von ParameterMapping-Objekten

QueryString

Die Abfragezeichenfolge, die Sie vorbereiten möchten.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge von 262144.

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion auszuführen.

HTTPStatuscode: 400

InternalServerException

Der Dienst konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 400

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ThrottlingException

Die Anforderung wurde aufgrund der Drosselung von Anforderungen abgelehnt.

HTTPStatuscode: 400

ValidationException

Ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

Query
Bedienung: Amazon Timestream Query

`Query`ist ein synchroner Vorgang, mit dem Sie eine Abfrage für Ihre Amazon Timestream Timestream-Daten ausführen können.

Wenn Sie diese Option aktiviert haben`QueryInsights`, werden API auch Erkenntnisse und Metriken zurückgegeben, die sich auf die von Ihnen ausgeführte Abfrage beziehen. `QueryInsights`shilft bei der Leistungsoptimierung Ihrer Abfrage. Weitere Informationen `QueryInsights` dazu finden Sie unter Verwenden von Abfrageerkenntnissen zur Optimierung von Abfragen in Amazon Timestream.

> **Note**
>
> Die maximale Anzahl von `Query` API Anfragen, die Sie bei `QueryInsights` aktivierter Option stellen dürfen, beträgt 1 Abfrage pro Sekunde (QPS). Wenn Sie diese Abfragerate überschreiten, kann dies zu einer Drosselung führen.

`Query`wird nach 60 Sekunden ein Timeout ausgelöst. Sie müssen das Standard-Timeout in der aktualisierenSDK, um ein Timeout von 60 Sekunden zu unterstützen. Einzelheiten finden Sie im Codebeispiel.

Ihre Abfrageanfrage schlägt in den folgenden Fällen fehl:

- Wenn Sie eine `Query` Anfrage mit demselben Client-Token außerhalb des 5-minütigen Idempotenzfensters einreichen.
- Wenn Sie innerhalb des 5-minütigen Idempotenzfensters eine `Query` Anfrage mit demselben Client-Token einreichen, aber andere Parameter ändern.
- Wenn die Größe der Zeile (einschließlich der Abfrage-Metadaten) 1 MB überschreitet, schlägt die Abfrage fehl und die folgende Fehlermeldung wird angezeigt:

  ```
  Query aborted as max page response size has been exceeded by the output result row
  ```
- Wenn der IAM Prinzipal des Abfrageinitiators und des Ergebnislesers nicht identisch sind und/oder der Abfrageinitiator und der Ergebnisleser nicht dieselbe Abfragezeichenfolge in den Abfrageanforderungen haben, schlägt die Abfrage mit einem `Invalid pagination token` Fehler fehl.

Anforderungssyntax

```
{
   "ClientToken": "string",
   "MaxRows": number,
   "NextToken": "string",
   "QueryInsights": {
      "Mode": "string"
   },
   "QueryString": "string"
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

ClientToken

Eindeutige, bei der Groß- und Kleinschreibung berücksichtigende Zeichenfolge mit bis zu 64 ASCII Zeichen, die bei der `Query` Anfrage angegeben wird. Die Angabe von a `ClientToken` macht den Aufruf `Query` idempotent. Das bedeutet, dass das wiederholte Ausführen derselben Abfrage zum gleichen Ergebnis führt. Mit anderen Worten, das Stellen mehrerer identischer `Query` Anfragen hat den gleichen Effekt wie das Stellen einer einzelnen Anfrage. Beachten Sie `ClientToken` bei der Verwendung in einer Abfrage Folgendes:

- Wenn die Abfrage ohne a instanziiert API wird`ClientToken`, SDK generiert die Abfrage in `ClientToken` Ihrem Namen eine.
- Wenn der `Query` Aufruf nur das, `ClientToken` aber kein A enthält, wird davon ausgegangen`NextToken`, dass es sich bei diesem Aufruf von um einen neuen Abfragelauf `Query` handelt.
- Wenn der Aufruf enthält`NextToken`, wird davon ausgegangen, dass es sich bei diesem speziellen Aufruf um einen nachfolgenden Aufruf eines vorherigen Abfrageaufrufs handeltAPI, und es wird eine Ergebnismenge zurückgegeben.
- Nach 4 Stunden `ClientToken` wird jede Anfrage mit demselben Inhalt wie eine neue Anfrage behandelt.

Typ: Zeichenfolge

Längenbeschränkungen: Mindestlänge von 32. Maximale Länge beträgt 128 Zeichen.

Erforderlich: Nein

MaxRows

Die Gesamtzahl der Zeilen, die in der `Query` Ausgabe zurückgegeben werden sollen. Bei der ersten Ausführung von `Query` mit einem angegebenen `MaxRows` Wert wird in zwei Fällen die Ergebnismenge der Abfrage zurückgegeben:

- Die Größe des Ergebnisses ist kleiner als`1MB`.
- Die Anzahl der Zeilen in der Ergebnismenge ist geringer als der Wert von`maxRows`.

Andernfalls gibt der erste Aufruf von `Query` only a zurück`NextToken`, das dann in nachfolgenden Aufrufen zum Abrufen der Ergebnismenge verwendet werden kann. Um die Paginierung fortzusetzen, geben Sie den `NextToken` Wert im nachfolgenden Befehl ein.

Wenn die Zeilengröße groß ist (z. B. eine Zeile hat viele Spalten), gibt Timestream möglicherweise weniger Zeilen zurück, um zu verhindern, dass die Antwortgröße die Grenze von 1 MB überschreitet. Wenn nicht `MaxRows` angegeben, sendet Timestream die erforderliche Anzahl von Zeilen, um das Limit von 1 MB einzuhalten.

Typ: Ganzzahl

Gültiger Bereich: Mindestwert 1. Maximaler Wert von 1 000.

Erforderlich: Nein

NextToken

Ein Paginierungstoken, das verwendet wird, um eine Reihe von Ergebnissen zurückzugeben. Wenn das mit aufgerufen `Query` API wird, wird davon ausgegangen`NextToken`, dass es sich bei diesem bestimmten Aufruf um einen nachfolgenden Aufruf eines vorherigen Aufrufs von handelt`Query`, und es wird eine Ergebnismenge zurückgegeben. Wenn der `Query` Aufruf jedoch nur den enthält, `Query` wird davon ausgegangen`ClientToken`, dass es sich bei diesem Aufruf von um einen neuen Abfragelauf handelt.

Beachten Sie bei der Verwendung NextToken in einer Abfrage Folgendes:

- Ein Paginierungstoken kann für bis zu fünf `Query` Aufrufe ODER für eine Dauer von bis zu 1 Stunde verwendet werden — je nachdem, was zuerst eintritt.

- Wenn Sie dasselbe verwenden, `NextToken` wird derselbe Satz von Datensätzen zurückgegeben. Um die Ergebnismenge weiterhin paginieren zu können, müssen Sie die neueste Version verwenden. `nextToken`
- Angenommen, ein `Query` Aufruf gibt zwei `NextToken` Werte zurück, und. `TokenA` `TokenB` Wenn in einem nachfolgenden `Query` Aufruf verwendet `TokenB` wird, `TokenA` ist es ungültig und kann nicht wiederverwendet werden.
- Um nach Beginn der Paginierung eine vorherige Ergebnismenge aus einer Abfrage anzufordern, müssen Sie die Abfrage erneut aufrufen. API
- Die neueste Version `NextToken` sollte verwendet werden, um zu paginieren, bis sie zurückgegeben `null` wird. Ab diesem Zeitpunkt sollte eine neue `NextToken` Version verwendet werden.
- Wenn der IAM Prinzipal des Abfrageinitiators und des Ergebnislesers nicht identisch sind und/oder der Abfrageinitiator und der Ergebnisleser nicht dieselbe Abfragezeichenfolge in den Abfrageanforderungen haben, schlägt die Abfrage mit einem Fehler fehl. `Invalid pagination token`

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Nein

QueryInsights

Kapselt Einstellungen für die Aktivierung. `QueryInsights`

Durch die Aktivierung werden zusätzlich zu den Abfrageergebnissen für die von Ihnen ausgeführte Abfrage auch Erkenntnisse und Metriken `QueryInsights` zurückgegeben. Sie können es verwenden`QueryInsights`, um Ihre Abfrageleistung zu optimieren.

Typ: QueryInsights Objekt

Erforderlich: Nein

QueryString

Die Abfrage, die von Timestream ausgeführt werden soll.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge von 262144.

Erforderlich: Ja

Antwortsyntax

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

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Dienst im JSON Format zurückgegeben.

ColumnInfo

Die Spaltendatentypen der zurückgegebenen Ergebnismenge.

Typ: Array von ColumnInfo-Objekten

NextToken

Ein Paginierungstoken, das bei einem `Query` Aufruf erneut verwendet werden kann, um die nächsten Ergebnisse abzurufen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

QueryId

Eine eindeutige ID für die angegebene Abfrage.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `[a-zA-Z0-9]+`

QueryInsightsResponse

`QueryInsights`Enthält Erkenntnisse und Metriken zu der von Ihnen ausgeführten Abfrage.

Typ: QueryInsightsResponse Objekt

QueryStatus

Informationen zum Status der Abfrage, einschließlich des Fortschritts und der gescannten Byte.

Typ: QueryStatus Objekt

Rows

Die von der Abfrage zurückgegebenen Ergebnismengenzeilen.

Typ: Array von Row-Objekten

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion auszuführen.

HTTPStatuscode: 400

ConflictException

Die Ergebnisse für eine stornierte Abfrage konnten nicht abgefragt werden.

HTTPStatuscode: 400

InternalServerException

Der Dienst konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 400

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

QueryExecutionException

Timestream konnte die Abfrage nicht erfolgreich ausführen.

HTTPStatuscode: 400

ThrottlingException

Die Anforderung wurde aufgrund der Drosselung von Anforderungen abgelehnt.

HTTPStatuscode: 400

ValidationException

Ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3

- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

TagResource
Bedienung: Amazon Timestream Query

Ordnen Sie einer Timestream-Ressource eine Reihe von Tags zu. Anschließend können Sie diese benutzerdefinierten Tags aktivieren, sodass sie in der Billing and Cost Management-Konsole zur Nachverfolgung der Kostenzuweisung angezeigt werden.

Anforderungssyntax

```
{
   "ResourceARN": "string",
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

ResourceARN

Identifiziert die Timestream-Ressource, zu der Tags hinzugefügt werden sollen. Dieser Wert ist ein Amazon-Ressourcenname (ARN).

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Ja

Tags

Die Tags, die der Timestream-Ressource zugewiesen werden sollen.

Typ: Array von Tag-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 200 Elemente.

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort mit einem leeren HTTP Text zurück.

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ResourceNotFoundException

Die angeforderte Ressource wurde nicht gefunden.

HTTPStatuscode: 400

ServiceQuotaExceededException

Sie haben das Servicekontingent überschritten.

HTTPStatuscode: 400

ThrottlingException

Die Anforderung wurde aufgrund der Drosselung von Anforderungen abgelehnt.

HTTPStatuscode: 400

ValidationException

Ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

UntagResource
Bedienung: Amazon Timestream Query

Entfernt die Zuordnung von Tags aus einer Timestream-Abfrageressource.

Anforderungssyntax

```
{
   "ResourceARN": "string",
   "TagKeys": [ "string" ]
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

ResourceARN

Die Timestream-Ressource, aus der die Tags entfernt werden. Dieser Wert ist ein Amazon-Ressourcenname (ARN).

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Ja

TagKeys

Eine Liste von Tags-Schlüsseln. Bestehende Tags der Ressource, deren Schlüssel Mitglieder dieser Liste sind, werden aus der Timestream-Ressource entfernt.

Typ: Zeichenfolge-Array

Array-Mitglieder: Die Mindestanzahl beträgt 0 Elemente. Die maximale Anzahl beträgt 200 Elemente.

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort mit einem leeren HTTP Text zurück.

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ResourceNotFoundException

Die angeforderte Ressource wurde nicht gefunden.

HTTPStatuscode: 400

ThrottlingException

Die Anforderung wurde aufgrund der Drosselung von Anforderungen abgelehnt.

HTTPStatuscode: 400

ValidationException

Ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET

- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

UpdateAccountSettings
Bedienung: Amazon Timestream Query

Stellt Ihr Konto auf die Verwendung TCUs für die Preisgestaltung von Abfragen um und ändert die maximale Anzahl an Recheneinheiten für Abfragen, die Sie konfiguriert haben. Wenn Sie den Wert von `MaxQueryTCU` auf eine gewünschte Konfiguration reduzieren, kann es bis zu 24 Stunden dauern, bis der neue Wert wirksam wird.

> **Note**
>
> Nachdem Sie Ihr Konto auf die Verwendung TCUs für Preisabfragen umgestellt haben, können Sie nicht mehr zur Verwendung von gescannten Bytes für die Preisabfrage übergehen.

Anforderungssyntax

```
{
   "MaxQueryTCU": number,
   "QueryPricingModel": "string"
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

MaxQueryTCU

Die maximale Anzahl von Recheneinheiten, die der Service zu einem beliebigen Zeitpunkt für die Bearbeitung Ihrer Abfragen verwendet. Um Abfragen ausführen zu können, müssen Sie eine Mindestkapazität von 4 festlegenTCU. Sie können die maximale Anzahl von TCU in Vielfachen von 4 festlegen, z. B. 4, 8, 16, 32 usw.

Der unterstützte Höchstwert `MaxQueryTCU` ist 1000. Um eine Erhöhung dieses Soft-Limits zu beantragen, wenden Sie sich an den AWS Support. Informationen zum Standardkontingent für finden Sie maxQuery TCU unter Standardkontingente.

Typ: Ganzzahl

Erforderlich: Nein

QueryPricingModel

Das Preismodell für Abfragen in einem Konto.

> **Note**
>
> Der `QueryPricingModel` Parameter wird von mehreren Timestream-Vorgängen verwendet. Der `UpdateAccountSettings` API Vorgang erkennt jedoch keine anderen Werte als`COMPUTE_UNITS`.

Typ: Zeichenfolge

Zulässige Werte: `BYTES_SCANNED | COMPUTE_UNITS`

Erforderlich: Nein

Antwortsyntax

```
{
   "MaxQueryTCU": number,
   "QueryPricingModel": "string"
}
```

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort zurück.

Die folgenden Daten werden vom Dienst im JSON Format zurückgegeben.

MaxQueryTCU

Die konfigurierte maximale Anzahl von Recheneinheiten, die der Service zu einem beliebigen Zeitpunkt für die Bearbeitung Ihrer Abfragen verwendet.

Typ: Ganzzahl

QueryPricingModel

Das Preismodell für ein Konto.

Typ: Zeichenfolge

Zulässige Werte: `BYTES_SCANNED | COMPUTE_UNITS`

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion auszuführen.

HTTPStatuscode: 400

InternalServerException

Der Dienst konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 400

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ThrottlingException

Die Anforderung wurde aufgrund der Drosselung von Anforderungen abgelehnt.

HTTPStatuscode: 400

ValidationException

Ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

UpdateScheduledQuery
Bedienung: Amazon Timestream Query

Aktualisieren Sie eine geplante Abfrage.

Anforderungssyntax

```
{
   "ScheduledQueryArn": "string",
   "State": "string"
}
```

Anforderungsparameter

Informationen zu den Parametern, die alle Aktionen gemeinsam haben, finden Sie unter Allgemeine Parameter.

Die Anfrage akzeptiert die folgenden Daten im JSON Format.

ScheduledQueryArn

ARNder geplanten Abfrage.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Ja

State

Status der geplanten Abfrage.

Typ: Zeichenfolge

Zulässige Werte: `ENABLED | DISABLED`

Erforderlich: Ja

Antwortelemente

Wenn die Aktion erfolgreich ist, sendet der Dienst eine HTTP 200-Antwort mit einem leeren HTTP Text zurück.

Fehler

Weitere Informationen zu den allgemeinen Fehlern, die bei allen Aktionen zurückgegeben werden, finden Sie unter Häufige Fehler.

AccessDeniedException

Sie sind nicht berechtigt, diese Aktion auszuführen.

HTTPStatuscode: 400

InternalServerException

Der Dienst konnte diese Anfrage aufgrund eines internen Serverfehlers nicht vollständig verarbeiten.

HTTPStatuscode: 400

InvalidEndpointException

Der angeforderte Endpunkt war nicht gültig.

HTTPStatuscode: 400

ResourceNotFoundException

Die angeforderte Ressource wurde nicht gefunden.

HTTPStatuscode: 400

ThrottlingException

Die Anforderung wurde aufgrund der Drosselung von Anforderungen abgelehnt.

HTTPStatuscode: 400

ValidationException

Ungültige oder falsch formatierte Anfrage.

HTTPStatuscode: 400

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS -Befehlszeilenschnittstelle
- AWS SDKfür. NET
- AWS SDKfür C++
- AWS SDKfür Go v2
- AWS SDKfür Java V2
- AWS SDKfür JavaScript V3
- AWS SDKfür PHP V3
- AWS SDKfür Python
- AWS SDKfür Ruby V3

Datentypen

Die folgenden Datentypen werden von Amazon Timestream Write unterstützt:

- 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

Die folgenden Datentypen werden von Amazon Timestream Query unterstützt:

- 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 Write

Die folgenden Datentypen werden von Amazon Timestream Write unterstützt:

- 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
Bedienung: Amazon Timestream Write

Details zum Fortschritt einer Batch-Load-Aufgabe.

Inhalt

BytesMetered

Type: Long

Erforderlich: Nein

FileFailures

Type: Long

Erforderlich: Nein

ParseFailures

Type: Long

Erforderlich: Nein

RecordIngestionFailures

Type: Long

Erforderlich: Nein

RecordsIngested

Type: Long

Erforderlich: Nein

RecordsProcessed

Type: Long

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

BatchLoadTask

Bedienung: Amazon Timestream Write

Details zu einer Batch-Load-Aufgabe.

Inhalt

CreationTime

Der Zeitpunkt, zu dem die Timestream-Aufgabe zum Batch-Laden erstellt wurde.

Typ: Zeitstempel

Erforderlich: Nein

DatabaseName

Datenbankname für die Datenbank, in die eine Batch-Load-Task Daten lädt.

Typ: Zeichenfolge

Erforderlich: Nein

LastUpdatedTime

Der Zeitpunkt, zu dem die Timestream-Batch-Load-Task zuletzt aktualisiert wurde.

Typ: Zeitstempel

Erforderlich: Nein

ResumableUntil

Typ: Zeitstempel

Erforderlich: Nein

TableName

Tabellenname für die Tabelle, in die eine Batch-Load-Task Daten lädt.

Typ: Zeichenfolge

Erforderlich: Nein

TaskId

Die ID der Batch-Load-Task.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 32 Zeichen.

Pattern: `[A-Z0-9]+`

Erforderlich: Nein

TaskStatus

Status der Batch-Load-Aufgabe.

Typ: Zeichenfolge

Zulässige Werte: `CREATED | IN_PROGRESS | FAILED | SUCCEEDED | PROGRESS_STOPPED | PENDING_RESUME`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

BatchLoadTaskDescription
Bedienung: Amazon Timestream Write

Details zu einer Batch-Load-Aufgabe.

Inhalt

CreationTime

Der Zeitpunkt, zu dem die Timestream-Aufgabe zum Batch-Laden erstellt wurde.

Typ: Zeitstempel

Erforderlich: Nein

DataModelConfiguration

Datenmodellkonfiguration für eine Batch-Load-Task. Dies enthält Details darüber, wo ein Datenmodell für eine Batch-Load-Task gespeichert ist.

Typ: DataModelConfiguration Objekt

Erforderlich: Nein

DataSourceConfiguration

Konfigurationsdetails zur Datenquelle für eine Batch-Load-Task.

Typ: DataSourceConfiguration Objekt

Erforderlich: Nein

ErrorMessage

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Nein

LastUpdatedTime

Der Zeitpunkt, zu dem die Timestream-Batch-Load-Task zuletzt aktualisiert wurde.

Typ: Zeitstempel

Erforderlich: Nein

ProgressReport

Typ: BatchLoadProgressReport Objekt

Erforderlich: Nein

RecordVersion

Type: Long

Erforderlich: Nein

ReportConfiguration

Berichtskonfiguration für eine Batch-Load-Task. Dies enthält Details darüber, wo Fehlerberichte gespeichert werden.

Typ: ReportConfiguration Objekt

Erforderlich: Nein

ResumableUntil

Typ: Zeitstempel

Erforderlich: Nein

TargetDatabaseName

Typ: Zeichenfolge

Erforderlich: Nein

TargetTableName

Typ: Zeichenfolge

Erforderlich: Nein

TaskId

Die ID der Batch-Load-Aufgabe.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 32 Zeichen.

Pattern: `[A-Z0-9]+`

Erforderlich: Nein

TaskStatus

Status der Batch-Load-Aufgabe.

Typ: Zeichenfolge

Zulässige Werte: `CREATED | IN_PROGRESS | FAILED | SUCCEEDED | PROGRESS_STOPPED | PENDING_RESUME`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

CsvConfiguration
Bedienung: Amazon Timestream Write

Ein durch Trennzeichen getrenntes Datenformat, bei dem das Spaltentrennzeichen ein Komma und das Datensatztrennzeichen ein Zeilenumbruchzeichen sein kann.

Inhalt

ColumnSeparator

Das Spaltentrennzeichen kann aus Komma (','), Pipe ('|), Semikolon (';'), Tab ('/t') oder Leerzeichen (') bestehen.

Typ: Zeichenfolge

Längenbeschränkungen: Feste Länge von 1.

Erforderlich: Nein

EscapeChar

Das Escape-Zeichen kann eines von sein

Typ: Zeichenfolge

Längenbeschränkungen: Feste Länge von 1.

Erforderlich: Nein

NullValue

Kann ein Leerzeichen (') sein.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Nein

QuoteChar

Kann ein einfaches Anführungszeichen (') oder ein doppeltes Anführungszeichen („) sein.

Typ: Zeichenfolge

Längenbeschränkungen: Feste Länge von 1.

Erforderlich: Nein

TrimWhiteSpace

Legt fest, dass der Leerraum am Anfang und am Ende gekürzt werden soll.

Typ: Boolesch

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen dazu, wie Sie dies API in einer der sprachspezifischen Sprachen verwenden können AWS SDKs, finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

Database

Bedienung: Amazon Timestream Write

Ein Container auf oberster Ebene für eine Tabelle. Datenbanken und Tabellen sind die grundlegenden Verwaltungskonzepte in Amazon Timestream. Alle Tabellen in einer Datenbank sind mit demselben AWS KMS Schlüssel verschlüsselt.

Inhalt

Arn

Der Amazon-Ressourcenname, der diese Datenbank eindeutig identifiziert.

Typ: Zeichenfolge

Erforderlich: Nein

CreationTime

Der Zeitpunkt, zu dem die Datenbank erstellt wurde, berechnet anhand der Zeit der Unix-Epoche.

Typ: Zeitstempel

Erforderlich: Nein

DatabaseName

Der Name der Timestream-Datenbank.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Nein

KmsKeyId

Die Kennung des AWS KMS Schlüssels, der zur Verschlüsselung der in der Datenbank gespeicherten Daten verwendet wird.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Nein

LastUpdatedTime

Das letzte Mal, dass diese Datenbank aktualisiert wurde.

Typ: Zeitstempel

Erforderlich: Nein

TableCount

Die Gesamtzahl der in einer Timestream-Datenbank gefundenen Tabellen.

Type: Long

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

DataModel
Bedienung: Amazon Timestream Write

Datenmodell für eine Batch-Load-Aufgabe.

Inhalt

DimensionMappings

Zuordnungen von Quelle zu Ziel für Dimensionen.

Typ: Array von DimensionMapping-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 1 Element.

Erforderlich: Ja

MeasureNameColumn

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Nein

MixedMeasureMappings

Zuordnungen von Quelle zu Ziel für Kennzahlen.

Typ: Array von MixedMeasureMapping-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 1 Element.

Erforderlich: Nein

MultiMeasureMappings

Quell-Ziel-Zuordnungen für Datensätze mit mehreren Kennzahlen.

Typ: MultiMeasureMappings Objekt

Erforderlich: Nein

TimeColumn

Quellspalte, die der Zeit zugeordnet werden soll.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Nein

TimeUnit

Die Granularität der Zeitstempeleinheit. Sie gibt an, ob der Zeitwert in Sekunden, Millisekunden, Nanosekunden oder anderen unterstützten Werten angegeben ist. Der Standardwert ist `MILLISECONDS`.

Typ: Zeichenfolge

Zulässige Werte: `MILLISECONDS | SECONDS | MICROSECONDS | NANOSECONDS`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

DataModelConfiguration
Bedienung: Amazon Timestream Write

Inhalt

DataModel

Typ: DataModel Objekt

Erforderlich: Nein

DataModelS3Configuration

Typ: DataModelS3Configuration Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

DataModelS3Configuration
Bedienung: Amazon Timestream Write

Inhalt

BucketName

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 63 Zeichen.

Pattern: `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Erforderlich: Nein

ObjectKey

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 1024 Zeichen.

Pattern: `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen finden Sie im Folgenden AWS SDKs:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

DataSourceConfiguration
Bedienung: Amazon Timestream Write

Definiert Konfigurationsdetails zur Datenquelle.

Inhalt

DataFormat

Das ist derzeitCSV.

Typ: Zeichenfolge

Zulässige Werte: `CSV`

Erforderlich: Ja

DataSourceS3Configuration

Konfiguration eines S3-Speicherorts für eine Datei, die zu ladende Daten enthält.

Typ: DataSourceS3Configuration Objekt

Erforderlich: Ja

CsvConfiguration

Ein durch Trennzeichen getrenntes Datenformat, bei dem das Spaltentrennzeichen ein Komma und das Datensatztrennzeichen ein Zeilenumbruchzeichen sein kann.

Typ: CsvConfiguration Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen dazu, wie Sie dies API in einer der sprachspezifischen Sprachen AWS SDKs verwenden können, finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

DataSourceS3Configuration
Bedienung: Amazon Timestream Write

Inhalt

BucketName

Der Bucket-Name des S3-Buckets des Kunden.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 63 Zeichen.

Pattern: `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Erforderlich: Ja

ObjectKeyPrefix

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 1. Maximale Länge beträgt 1024 Zeichen.

Pattern: `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen finden Sie im Folgenden AWS SDKs:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

Dimension
Bedienung: Amazon Timestream Write

Stellt die Metadatenattribute der Zeitreihe dar. Beispielsweise sind der Name und die Availability Zone einer EC2 Instanz oder der Name des Herstellers einer Windturbine Dimensionen.

Inhalt

Name

Dimension stellt die Metadatenattribute der Zeitreihe dar. Beispielsweise sind der Name und die Availability Zone einer EC2 Instanz oder der Name des Herstellers einer Windkraftanlage Dimensionen.

Einschränkungen für Dimensionsnamen finden Sie unter Einschränkungen bei der Benennung von Dimensionen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Die maximale Länge beträgt 60.

Erforderlich: Ja

Value

Der Wert der Dimension.

Typ: Zeichenfolge

Erforderlich: Ja

DimensionValueType

Der Datentyp der Dimension für den Zeitreihen-Datenpunkt.

Typ: Zeichenfolge

Zulässige Werte: `VARCHAR`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

DimensionMapping
Bedienung: Amazon Timestream Write

Inhalt

DestinationColumn

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen.

Erforderlich: Nein

SourceColumn

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen.

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

Endpoint
Bedienung: Amazon Timestream Write

Stellt einen verfügbaren Endpunkt dar, gegen den API Anrufe getätigt werden sollen, sowie den TTL für diesen Endpunkt.

Inhalt

Address

Eine Endpunktadresse.

Typ: Zeichenfolge

Erforderlich: Ja

CachePeriodInMinutes

Die TTL für den Endpunkt, in Minuten.

Type: Long

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

MagneticStoreRejectedDataLocation
Bedienung: Amazon Timestream Write

Der Ort, an dem Fehlerberichte für Datensätze geschrieben werden, die beim Schreiben von Magnetspeichern asynchron abgelehnt wurden.

Inhalt

S3Configuration

Konfiguration eines S3-Speicherorts an dem Fehlerberichte für Datensätze geschrieben werden, die beim Schreiben von Magnetspeichern asynchron abgelehnt wurden.

Typ: S3Configuration Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

MagneticStoreWriteProperties
Bedienung: Amazon Timestream Write

Der Satz von Eigenschaften in einer Tabelle für die Konfiguration magnetischer Speicherschreibvorgänge.

Inhalt

EnableMagneticStoreWrites

Ein Flag, um Magnetspeicher-Schreibvorgänge zu aktivieren.

Typ: Boolesch

Erforderlich: Ja

MagneticStoreRejectedDataLocation

Der Ort, an dem Fehlerberichte für Datensätze geschrieben werden, die beim Schreiben von Magnetspeichern asynchron abgelehnt wurden.

Typ: MagneticStoreRejectedDataLocation Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

MeasureValue
Bedienung: Amazon Timestream Write

Stellt das Datenattribut der Zeitreihe dar. Beispielsweise handelt es sich bei der CPU Auslastung einer EC2 Instanz oder RPM der einer Windkraftanlage um Messwerte. MeasureValue hat sowohl einen Namen als auch einen Wert.

MeasureValue ist nur für den Typ zulässig`MULTI`. Mithilfe von `MULTI` type können Sie mehrere Datenattribute, die derselben Zeitreihe zugeordnet sind, in einem einzigen Datensatz übergeben

Inhalt

Name

Der Name des MeasureValue.

Einschränkungen für MeasureValue Namen finden Sie unter Benennungsbeschränkungen im Amazon Timestream Developer Guide.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen.

Erforderlich: Ja

Type

Enthält den Datentyp des MeasureValue Datenpunkts für die Zeitreihe.

Typ: Zeichenfolge

Zulässige Werte: `DOUBLE | BIGINT | VARCHAR | BOOLEAN | TIMESTAMP | MULTI`

Erforderlich: Ja

Value

Der Wert für. MeasureValue Weitere Informationen finden Sie unter Datentypen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

MixedMeasureMapping
Bedienung: Amazon Timestream Write

Inhalt

MeasureValueType

Typ: Zeichenfolge

Zulässige Werte: `DOUBLE | BIGINT | VARCHAR | BOOLEAN | TIMESTAMP | MULTI`

Erforderlich: Ja

MeasureName

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen.

Erforderlich: Nein

MultiMeasureAttributeMappings

Typ: Array von MultiMeasureAttributeMapping-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 1 Element.

Erforderlich: Nein

SourceColumn

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen.

Erforderlich: Nein

TargetMeasureName

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen.

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

MultiMeasureAttributeMapping
Bedienung: Amazon Timestream Write

Inhalt

SourceColumn

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen.

Erforderlich: Ja

MeasureValueType

Typ: Zeichenfolge

Zulässige Werte: `DOUBLE | BIGINT | BOOLEAN | VARCHAR | TIMESTAMP`

Erforderlich: Nein

TargetMultiMeasureAttributeName

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen.

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

MultiMeasureMappings
Bedienung: Amazon Timestream Write

Inhalt

MultiMeasureAttributeMappings

Typ: Array von MultiMeasureAttributeMapping-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 1 Element.

Erforderlich: Ja

TargetMultiMeasureName

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen.

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

PartitionKey
Bedienung: Amazon Timestream Write

Ein Attribut, das bei der Partitionierung von Daten in einer Tabelle verwendet wird. Ein Dimensionsschlüssel partitioniert Daten mithilfe der Werte der Dimension, die durch den Dimensionsnamen als Partitionsschlüssel angegeben sind, während ein Kennzahlschlüssel Daten mithilfe von Kennzahlnamen (Werte der Spalte 'measure_name') partitioniert.

Inhalt

Type

Der Typ des Partitionsschlüssels. Die Optionen sind DIMENSION (Dimensionsschlüssel) und MEASURE (Maßschlüssel).

Typ: Zeichenfolge

Zulässige Werte: `DIMENSION | MEASURE`

Erforderlich: Ja

EnforcementInRecord

Der Grad der Durchsetzung bei der Angabe eines Dimensionsschlüssels in aufgenommenen Datensätzen. Die Optionen sind REQUIRED (Dimensionsschlüssel muss angegeben werden) und OPTIONAL (Dimensionsschlüssel muss nicht angegeben werden).

Typ: Zeichenfolge

Zulässige Werte: `REQUIRED | OPTIONAL`

Erforderlich: Nein

Name

Der Name des Attributs, das für einen Dimensionsschlüssel verwendet wird.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen.

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

Record
Bedienung: Amazon Timestream Write

Stellt einen Zeitreihen-Datenpunkt dar, der in Timestream geschrieben wird. Jeder Datensatz enthält eine Reihe von Dimensionen. Dimensionen stellen die Metadatenattribute eines Zeitreihen-Datenpunkts dar, z. B. den Instanznamen oder die Availability Zone einer EC2 Instanz. Ein Datensatz enthält auch den Kennzahlnamen, d. h. den Namen der Messgröße, die erfasst wird (z. B. die CPU Auslastung einer EC2 Instanz). Darüber hinaus enthält ein Datensatz den Messwert und den Werttyp, der den Datentyp des Messwerts darstellt. Außerdem enthält der Datensatz den Zeitstempel, zu dem die Kennzahl erfasst wurde, und die Zeitstempeleinheit, die die Granularität des Zeitstempels darstellt.

Datensätze haben ein `Version` Feld, bei dem es sich um ein 64-Bit-Feld handelt`long`, das Sie zum Aktualisieren von Datenpunkten verwenden können. Schreibvorgänge an einem doppelten Datensatz mit derselben Dimension, demselben Zeitstempel und demselben Kennzahlnamen, aber unterschiedlichem Kennzahlwert sind nur erfolgreich, wenn das `Version` Attribut des Datensatzes in der Schreibanforderung höher ist als das des vorhandenen Datensatzes. Timestream ist bei Datensätzen ohne das Feld standardmäßig auf „`Version`of`1`“ eingestellt. `Version`

Inhalt

Dimensions

Enthält die Liste der Dimensionen für Zeitreihen-Datenpunkte.

Typ: Array von Dimension-Objekten

Array-Mitglieder: Maximale Anzahl von 128 Elementen.

Erforderlich: Nein

MeasureName

Measure stellt das Datenattribut der Zeitreihe dar. Beispielsweise handelt es sich bei der CPU Auslastung einer EC2 Instanz oder RPM der einer Windenergieanlage um Messwerte.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Nein

MeasureValue

Enthält den Messwert für den Zeitreihen-Datenpunkt.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Nein

MeasureValues

Enthält die Liste der vier MeasureValue Zeitreihen-Datenpunkte.

Dies ist nur für den Typ `MULTI` zulässig. Verwenden Sie für Skalarwerte direkt `MeasureValue` das Attribut des Datensatzes.

Typ: Array von MeasureValue-Objekten

Erforderlich: Nein

MeasureValueType

Enthält den Datentyp des Messwerts für den Zeitreihendatenpunkt. Der Standardtyp ist`DOUBLE`. Weitere Informationen finden Sie unter Datentypen.

Typ: Zeichenfolge

Zulässige Werte: `DOUBLE | BIGINT | VARCHAR | BOOLEAN | TIMESTAMP | MULTI`

Erforderlich: Nein

Time

Enthält den Zeitpunkt, zu dem der Messwert für den Datenpunkt erfasst wurde. Der Zeitwert plus die Einheit gibt die Zeit an, die seit der Epoche vergangen ist. Wenn der Zeitwert beispielsweise ist `12345` und die Einheit ist`ms`, dann ist seit der `12345 ms` Epoche vergangen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Nein

TimeUnit

Die Granularität der Zeitstempeleinheit. Sie gibt an, ob der Zeitwert in Sekunden, Millisekunden, Nanosekunden oder anderen unterstützten Werten angegeben ist. Der Standardwert ist `MILLISECONDS`.

Typ: Zeichenfolge

Zulässige Werte: `MILLISECONDS | SECONDS | MICROSECONDS | NANOSECONDS`

Erforderlich: Nein

Version

64-Bit-Attribut, das für Datensatzaktualisierungen verwendet wird. Schreibanforderungen für doppelte Daten mit einer höheren Versionsnummer aktualisieren den vorhandenen Messwert und die Version. In Fällen, in denen der Messwert derselbe ist, `Version` wird er trotzdem aktualisiert. Der Standardwert ist 1.

> **Note**
>
> `Version`muss 1 oder höher sein, sonst erhalten Sie eine `ValidationException` Fehlermeldung.

Type: Long

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

RecordsIngested
Bedienung: Amazon Timestream Write

Informationen zu den Aufzeichnungen, die im Rahmen dieser Anfrage aufgenommen wurden.

Inhalt

MagneticStore

Anzahl der Datensätze, die in den Magnetspeicher aufgenommen wurden.

Typ: Ganzzahl

Erforderlich: Nein

MemoryStore

Anzahl der in den Speicherspeicher aufgenommenen Datensätze.

Typ: Ganzzahl

Erforderlich: Nein

Total

Gesamtzahl der erfolgreich aufgenommenen Datensätze.

Typ: Ganzzahl

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

RejectedRecord
Bedienung: Amazon Timestream Write

Stellt Datensätze dar, die aufgrund von Datenvalidierungsproblemen, die vor dem erneuten Einfügen von Zeitreihendaten in das System behoben werden müssen, nicht erfolgreich in Timestream eingefügt wurden.

Inhalt

ExistingVersion

Die bestehende Version des Datensatzes. Dieser Wert wird in Szenarien aufgefüllt, in denen ein identischer Datensatz mit einer höheren Version als der Version in der Schreibanforderung vorhanden ist.

Type: Long

Erforderlich: Nein

Reason

Der Grund, warum ein Datensatz nicht erfolgreich in Timestream eingefügt wurde. Zu den möglichen Fehlerursachen gehören:

- Datensätze mit doppelten Daten, bei denen es mehrere Datensätze mit denselben Dimensionen, Zeitstempeln und Kennzahlnamen gibt, aber:
  - Die Kennzahlwerte sind unterschiedlich
  - Version ist in der Anfrage nicht vorhanden, oder der Wert der Version im neuen Datensatz ist gleich oder niedriger als der vorhandene Wert

  Wenn Timestream Daten für diesen Fall ablehnt, gibt das `ExistingVersion` Feld in der `RejectedRecords` Antwort die Version des aktuellen Datensatzes an. Um eine Aktualisierung zu erzwingen, können Sie die Anfrage erneut senden, wobei eine Version für den Datensatz auf einen Wert gesetzt ist, der größer als der ist. `ExistingVersion`
- Datensätze mit Zeitstempeln, die außerhalb der Aufbewahrungsdauer des Speicherspeichers liegen.

  > **Note**
  >
  > Wenn das Aufbewahrungsfenster aktualisiert wird, erhalten Sie eine `RejectedRecords` Ausnahme, wenn Sie sofort versuchen, Daten innerhalb des neuen

Fensters aufzunehmen. Um eine `RejectedRecords` Ausnahme zu vermeiden, warten Sie mit der Aufnahme neuer Daten, bis die Dauer des neuen Fensters abgelaufen ist. Weitere Informationen finden Sie unter Bewährte Methoden für die Konfiguration von Timestream und in der Erläuterung der Funktionsweise von Speicher in Timestream.

- Datensätze mit Dimensionen oder Kennzahlen, die die von Timestream definierten Grenzwerte überschreiten.

Weitere Informationen finden Sie unter Zugriffsverwaltung im Timestream-Entwicklerhandbuch.

Typ: Zeichenfolge

Erforderlich: Nein

RecordIndex

Der Index des Datensatzes in der Eingabeanforderung für WriteRecords. Indizes beginnen mit 0.

Typ: Ganzzahl

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

ReportConfiguration
Bedienung: Amazon Timestream Write

Berichtskonfiguration für eine Batch-Load-Aufgabe. Dies enthält Details darüber, wo Fehlerberichte gespeichert werden.

Inhalt

ReportS3Configuration

Konfiguration eines S3-Speicherorts zum Schreiben von Fehlerberichten und Ereignissen für ein Batch-Load.

Typ: ReportS3Configuration Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

ReportS3Configuration
Bedienung: Amazon Timestream Write

Inhalt

BucketName

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 63 Zeichen.

Pattern: `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Erforderlich: Ja

EncryptionOption

Typ: Zeichenfolge

Zulässige Werte: `SSE_S3 | SSE_KMS`

Erforderlich: Nein

KmsKeyId

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Nein

ObjectKeyPrefix

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge von 928.

Pattern: `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

RetentionProperties
Bedienung: Amazon Timestream Write

Die Aufbewahrungseigenschaften beinhalten die Dauer, für die Ihre Zeitreihendaten im Magnetspeicher und im Arbeitsspeicher gespeichert werden müssen.

Inhalt

MagneticStoreRetentionPeriodInDays

Die Dauer, für die Daten im Magnetspeicher gespeichert werden müssen.

Type: Long

Gültiger Bereich: Mindestwert 1. Höchstwert von 73000.

Erforderlich: Ja

MemoryStoreRetentionPeriodInHours

Die Dauer, für die Daten im Arbeitsspeicher gespeichert werden müssen.

Type: Long

Gültiger Bereich: Mindestwert 1. Maximalwert von 8766.

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

S3Configuration

Bedienung: Amazon Timestream Write

Die Konfiguration, die einen S3-Speicherort angibt.

Inhalt

BucketName

Der Bucket-Name des S3-Buckets des Kunden.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 63 Zeichen.

Pattern: `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Erforderlich: Nein

EncryptionOption

Die Verschlüsselungsoption für den S3-Speicherort des Kunden. Die Optionen sind serverseitige S3-Verschlüsselung mit einem verwalteten S3-Schlüssel oder einem AWS verwalteten Schlüssel.

Typ: Zeichenfolge

Zulässige Werte: `SSE_S3 | SSE_KMS`

Erforderlich: Nein

KmsKeyId

Die AWS KMS Schlüssel-ID für den S3-Standort des Kunden bei der Verschlüsselung mit einem AWS verwalteten Schlüssel.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Nein

ObjectKeyPrefix

Die Objektschlüsselvorschau für den S3-Standort des Kunden.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Die maximale Länge beträgt 928.

Pattern: `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

Schema
Bedienung: Amazon Timestream Write

Ein Schema spezifiziert das erwartete Datenmodell der Tabelle.

Inhalt

CompositePartitionKey

Eine nicht leere Liste von Partitionsschlüsseln, die die Attribute definieren, die zur Partitionierung der Tabellendaten verwendet werden. Die Reihenfolge der Liste bestimmt die Partitionshierarchie. Der Name und der Typ der einzelnen Partitionsschlüssel sowie die Reihenfolge der Partitionsschlüssel können nach der Erstellung der Tabelle nicht geändert werden. Die Durchsetzungsstufe der einzelnen Partitionsschlüssel kann jedoch geändert werden.

Typ: Array von PartitionKey-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 1 Element.

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen dazu, wie Sie dies API in einer der sprachspezifischen Sprachen verwenden können AWS SDKs, finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

Table
Bedienung: Amazon Timestream Write

Stellt eine Datenbanktabelle in Timestream dar. Tabellen enthalten eine oder mehrere verwandte Zeitreihen. Sie können die Aufbewahrungsdauer des Speicherspeichers und des Magnetspeichers für eine Tabelle ändern.

Inhalt

Arn

Der Amazon-Ressourcenname, der diese Tabelle eindeutig identifiziert.

Typ: Zeichenfolge

Erforderlich: Nein

CreationTime

Der Zeitpunkt, zu dem die Timestream-Tabelle erstellt wurde.

Typ: Zeitstempel

Erforderlich: Nein

DatabaseName

Der Name der Timestream-Datenbank, die diese Tabelle enthält.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Nein

LastUpdatedTime

Die Uhrzeit, zu der die Timestream-Tabelle zuletzt aktualisiert wurde.

Typ: Zeitstempel

Erforderlich: Nein

MagneticStoreWriteProperties

Enthält Eigenschaften, die beim Aktivieren von Magnetspeicher-Schreibvorgängen in der Tabelle festgelegt werden können.

Typ: MagneticStoreWriteProperties Objekt

Erforderlich: Nein

RetentionProperties

Die Aufbewahrungsdauer für den In-Memory-Speicher und den Magnetspeicher.

Typ: RetentionProperties Objekt

Erforderlich: Nein

Schema

Das Schema der Tabelle.

Typ: Schema Objekt

Erforderlich: Nein

TableName

Der Name der Timestream-Tabelle.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Nein

TableStatus

Der aktuelle Status der Tabelle:

- `DELETING`- Die Tabelle wird gelöscht.
- `ACTIVE`- Die Tabelle ist einsatzbereit.

Typ: Zeichenfolge

Zulässige Werte: `ACTIVE | DELETING | RESTORING`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

Tag
Bedienung: Amazon Timestream Write

Ein Tag ist eine Bezeichnung, die Sie einer 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 Timestream-Datenbanktabelle zuweisen, z. B. nach Zweck, Besitzer oder Umgebung.

Inhalt

Key

Der Schlüssel des Tags. Bei Tag-Schlüsseln wird zwischen Groß- und Kleinschreibung unterschieden.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Erforderlich: Ja

Value

Der Wert des Tags. Bei Tag-Werten wird zwischen Groß- und Kleinschreibung unterschieden und sie können Null sein.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen dazu, wie Sie dies API in einer der sprachspezifischen Sprachen verwenden können AWS SDKs, finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

Amazon Timestream Timestream-Abfrage

Die folgenden Datentypen werden von Amazon Timestream Query unterstützt:

- 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
Bedienung: Amazon Timestream Query

Enthält die Metadaten für Abfrageergebnisse wie Spaltennamen, Datentypen und andere Attribute.

Inhalt

Type

Der Datentyp der Ergebnismengenspalte. Der Datentyp kann skalar oder komplex sein. Skalare Datentypen sind Ganzzahlen, Zeichenketten, Doppelzahlen, Boolesche Werte und andere. Komplexe Datentypen sind Typen wie Arrays, Zeilen und andere.

Typ: Type Objekt

Erforderlich: Ja

Name

Der Name der Ergebnismengenspalte. Der Name der Ergebnismenge ist für Spalten aller Datentypen außer für Arrays verfügbar.

Typ: Zeichenfolge

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

Datum

Bedienung: Amazon Timestream Query

Datum steht für einen einzelnen Datenpunkt in einem Abfrageergebnis.

Inhalt

ArrayValue

Gibt an, ob der Datenpunkt ein Array ist.

Typ: Array von Datum-Objekten

Erforderlich: Nein

NullValue

Gibt an, ob der Datenpunkt Null ist.

Typ: Boolesch

Erforderlich: Nein

RowValue

Gibt an, ob der Datenpunkt eine Zeile ist.

Typ: Row Objekt

Erforderlich: Nein

ScalarValue

Gibt an, ob es sich bei dem Datenpunkt um einen Skalarwert wie Integer, String, Double oder Boolean handelt.

Typ: Zeichenfolge

Erforderlich: Nein

TimeSeriesValue

Gibt an, ob es sich bei dem Datenpunkt um einen Zeitreihendatentyp handelt.

Typ: Array von TimeSeriesDataPoint-Objekten

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

DimensionMapping
Bedienung: Amazon Timestream Query

Dieser Typ wird verwendet, um Spalten aus dem Abfrageergebnis einer Dimension in der Zieltabelle zuzuordnen.

Inhalt

DimensionValueType

Der Typ für die Dimension.

Typ: Zeichenfolge

Zulässige Werte: `VARCHAR`

Erforderlich: Ja

Name

Spaltenname aus dem Abfrageergebnis.

Typ: Zeichenfolge

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

Endpoint
Bedienung: Amazon Timestream Query

Stellt einen verfügbaren Endpunkt dar, gegen den API Anrufe getätigt werden sollen, sowie den TTL für diesen Endpunkt.

Inhalt

Address

Eine Endpunktadresse.

Typ: Zeichenfolge

Erforderlich: Ja

CachePeriodInMinutes

Die TTL für den Endpunkt, in Minuten.

Type: Long

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

ErrorReportConfiguration
Bedienung: Amazon Timestream Query

Konfiguration für die Fehler-Berichterstattung erforderlich.

Inhalt

S3Configuration

Die S3-Konfiguration für die Fehlerberichte.

Typ: S3Configuration Objekt

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

ErrorReportLocation
Bedienung: Amazon Timestream Query

Dies enthält den Speicherort des Fehlerberichts für einen einzelnen geplanten Abfrageaufruf.

Inhalt

S3ReportLocation

Der S3-Speicherort, an dem Fehlerberichte geschrieben werden.

Typ: S3ReportLocation Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

ExecutionStats
Bedienung: Amazon Timestream Query

Statistiken für einen einzelnen geplanten Abfragelauf.

Inhalt

BytesMetered

Byte, die für einen einzelnen geplanten Abfragelauf gemessen wurden.

Type: Long

Erforderlich: Nein

CumulativeBytesScanned

Für einen einzelnen geplanten Abfragelauf gescannte Byte.

Type: Long

Erforderlich: Nein

DataWrites

Die Anzahl der Schreibvorgänge wurde für Datensätze gemessen, die in einem einzigen geplanten Abfragelauf aufgenommen wurden.

Type: Long

Erforderlich: Nein

ExecutionTimeInMillis

Gesamtzeit, gemessen in Millisekunden, die für den Abschluss des geplanten Abfragelaufs benötigt wurde.

Type: Long

Erforderlich: Nein

QueryResultRows

Anzahl der Zeilen, die in der Ausgabe der Ausführung einer Abfrage vor der Aufnahme in die Zieldatenquelle vorhanden sind.

Type: Long

Erforderlich: Nein

RecordsIngested

Die Anzahl der Datensätze, die für einen einzelnen geplanten Abfragelauf aufgenommen wurden.

Type: Long

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

MixedMeasureMapping
Bedienung: Amazon Timestream Query

MixedMeasureMappings sind Zuordnungen, die verwendet werden können, um Daten in eine Mischung aus engen Kennzahlen und mehreren Kennzahlen in der abgeleiteten Tabelle aufzunehmen.

Inhalt

MeasureValueType

Typ des Werts, aus dem gelesen werden soll. sourceColumn Wenn das Mapping für istMULTI, verwenden Sie MeasureValueType. MULTI.

Typ: Zeichenfolge

Zulässige Werte: `BIGINT | BOOLEAN | DOUBLE | VARCHAR | MULTI`

Erforderlich: Ja

MeasureName

Bezieht sich auf den Wert von Measure_Name in einer Ergebniszeile. Dieses Feld ist erforderlich, wenn MeasureNameColumn es angegeben wird.

Typ: Zeichenfolge

Erforderlich: Nein

MultiMeasureAttributeMappings

Erforderlich wann measureValueType istMULTI. Attributzuordnungen für MULTI Wertkennzahlen.

Typ: Array von MultiMeasureAttributeMapping-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 1 Element.

Erforderlich: Nein

SourceColumn

Dieses Feld bezieht sich auf die Quellspalte, aus der der Messwert für die Materialisierung der Ergebnisse gelesen werden soll.

Typ: Zeichenfolge

Erforderlich: Nein

TargetMeasureName

Name der Ziel-Kennzahl, die verwendet werden soll. Falls nicht angegeben, lautet der Name der Zielkennzahl standardmäßig Kennzahlname, falls angegeben, oder auf andere Weise. sourceColumn

Typ: Zeichenfolge

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen finden Sie im Folgenden AWS SDKs:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

MultiMeasureAttributeMapping
Bedienung: Amazon Timestream Query

Zuordnung von Attributen für MULTI Wertkennzahlen.

Inhalt

MeasureValueType

Typ des Attributs, das aus der Quellspalte gelesen werden soll.

Typ: Zeichenfolge

Zulässige Werte: `BIGINT | BOOLEAN | DOUBLE | VARCHAR | TIMESTAMP`

Erforderlich: Ja

SourceColumn

Quellspalte, aus der der Attributwert gelesen werden soll.

Typ: Zeichenfolge

Erforderlich: Ja

TargetMultiMeasureAttributeName

Benutzerdefinierter Name für den Attributnamen in abgeleiteter Tabelle. Falls nicht angegeben, würde der Name der Quellspalte verwendet werden.

Typ: Zeichenfolge

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

MultiMeasureMappings
Bedienung: Amazon Timestream Query

Es muss nur einer von MixedMeasureMappings oder MultiMeasureMappings bereitgestellt werden. MultiMeasureMappings kann verwendet werden, um Daten als Mehrfachkennzahlen in die abgeleitete Tabelle aufzunehmen.

Inhalt

MultiMeasureAttributeMappings

Erforderlich Attribut-Zuordnungen, die zum Zuweisen von Abfrage-Ergebnissen zur Erfassung von Daten für Mehrfachkennzahl-Attribute verwendet werden.

Typ: Array von MultiMeasureAttributeMapping-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 1 Element.

Erforderlich: Ja

TargetMultiMeasureName

Der Name der Ziel-Mehrfachkennzahl in der abgeleiteten Tabelle. Diese Eingabe ist erforderlich, wenn sie nicht bereitgestellt measureNameColumn wird. Wenn angegeben, MeasureNameColumn wird der Wert aus dieser Spalte als Name für mehrere Kennzahlen verwendet.

Typ: Zeichenfolge

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen dazu, wie Sie dies API in einer der sprachspezifischen Sprachen verwenden können AWS SDKs, finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

NotificationConfiguration
Bedienung: Amazon Timestream Query

Benachrichtigungs-Konfiguration für eine geplante Abfrage. Timestream sendet eine Benachrichtigung, wenn eine geplante Abfrage erstellt, ihr Status aktualisiert oder gelöscht wird.

Inhalt

SnsConfiguration

Einzelheiten zur SNS Konfiguration.

Typ: SnsConfiguration Objekt

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

ParameterMapping
Bedienung: Amazon Timestream Query

Zuordnung für benannte Parameter.

Inhalt

Name

Name des Parameters.

Typ: Zeichenfolge

Erforderlich: Ja

Type

Enthält den Datentyp einer Spalte in einem Abfrageergebnissatz. Der Datentyp kann skalar oder komplex sein. Die unterstützten skalaren Datentypen sind Integer, Boolean, String, Double, Timestamp, Date, Time und Intervalle. Die unterstützten komplexen Datentypen sind Arrays, Zeilen und Zeitreihen.

Typ: Type Objekt

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen finden Sie im Folgenden AWS SDKs:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

QueryInsights
Bedienung: Amazon Timestream Query

`QueryInsights`ist eine Funktion zur Leistungsoptimierung, mit der Sie Ihre Abfragen optimieren, Kosten senken und die Leistung verbessern können. Damit `QueryInsights` können Sie die Effizienz der Bereinigung Ihrer Abfragen beurteilen und Bereiche identifizieren, in denen Verbesserungspotenzial besteht, um die Abfrageleistung zu verbessern. Mit `QueryInsights` können Sie auch die Effektivität Ihrer Abfragen im Hinblick auf die zeitliche und räumliche Kürzung analysieren und Möglichkeiten zur Leistungsverbesserung identifizieren. Insbesondere können Sie auswerten, wie gut Ihre Abfragen zeitbasierte und partitionsschlüsselbasierte Indizierungsstrategien verwenden, um den Datenabruf zu optimieren. Um die Abfrageleistung zu optimieren, ist es wichtig, dass Sie sowohl die zeitlichen als auch die räumlichen Parameter, die die Abfrageausführung steuern, fein abstimmen.

Die wichtigsten von bereitgestellten Metriken `QueryInsights` sind `QuerySpatialCoverage` und`QueryTemporalRange`. `QuerySpatialCoverage`gibt an, wie viel von der räumlichen Achse die Abfrage scannt, wobei niedrigere Werte effizienter sind. `QueryTemporalRange`zeigt den gescannten Zeitraum an, wobei engere Bereiche leistungsfähiger sind.

Vorteile von QueryInsights

Im Folgenden sind die wichtigsten Vorteile der Verwendung von`QueryInsights`:

- Identifizierung ineffizienter Abfragen — `QueryInsights` stellt Informationen zur zeit- und attributbasierten Bereinigung der Tabellen bereit, auf die mit der Abfrage zugegriffen wird. Anhand dieser Informationen können Sie die Tabellen identifizieren, auf die nicht optimal zugegriffen wird.
- Optimierung Ihres Datenmodells und Partitionierung — Sie können die `QueryInsights` Informationen verwenden, um auf Ihr Datenmodell und Ihre Partitionierungsstrategie zuzugreifen und diese zu optimieren.
- Optimieren von Abfragen — `QueryInsights` zeigt Möglichkeiten auf, Indizes effektiver zu nutzen.

> **Note**
>
> Die maximale Anzahl von `Query` API Anfragen, die Sie bei `QueryInsights` aktivierter Option stellen dürfen, beträgt 1 Abfrage pro Sekunde (QPS). Wenn Sie diese Abfragerate überschreiten, kann dies zu einer Drosselung führen.

Inhalt

Mode

Stellt die folgenden Aktivierungsmodi zur Verfügung: `QueryInsights`

- `ENABLED_WITH_RATE_CONTROL`— Aktiviert `QueryInsights` die Abfragen, die gerade verarbeitet werden. Dieser Modus beinhaltet auch einen Ratensteuerungsmechanismus, der die `QueryInsights` Funktion auf 1 Abfrage pro Sekunde beschränkt (QPS).
- `DISABLED`— Deaktiviert. `QueryInsights`

Typ: Zeichenfolge

Zulässige Werte: `ENABLED_WITH_RATE_CONTROL | DISABLED`

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

QueryInsightsResponse
Bedienung: Amazon Timestream Query

Bietet verschiedene Einblicke und Metriken in Bezug auf die von Ihnen ausgeführte Abfrage.

Inhalt

OutputBytes

Gibt die Größe des Abfrageergebnissatzes in Byte an. Sie können diese Daten verwenden, um zu überprüfen, ob sich die Ergebnismenge im Rahmen der Abfrageoptimierung geändert hat.

Type: Long

Erforderlich: Nein

OutputRows

Gibt die Gesamtzahl der Zeilen an, die als Teil des Abfrageergebnissatzes zurückgegeben wurden. Sie können diese Daten verwenden, um zu überprüfen, ob sich die Anzahl der Zeilen in der Ergebnismenge im Rahmen der Abfrageoptimierung geändert hat.

Type: Long

Erforderlich: Nein

QuerySpatialCoverage

Bietet Einblicke in die räumliche Abdeckung der Abfrage, einschließlich der Tabelle mit suboptimaler (maximaler) räumlicher Bereinigung. Diese Informationen können Ihnen dabei helfen, Bereiche zu identifizieren, in denen Ihre Partitionierungsstrategie verbessert werden muss, um die räumliche Kürzung zu verbessern.

Typ: QuerySpatialCoverage Objekt

Erforderlich: Nein

QueryTableCount

Gibt die Anzahl der Tabellen in der Abfrage an.

Type: Long

Erforderlich: Nein

QueryTemporalRange

Bietet Einblicke in den zeitlichen Bereich der Abfrage, einschließlich der Tabelle mit dem größten (maximalen) Zeitbereich. Im Folgenden sind einige der möglichen Optionen zur Optimierung der zeitbasierten Bereinigung aufgeführt:

- Fehlende Zeitprädikate hinzufügen.
- Entferne Funktionen rund um die Zeitprädikate.
- Fügen Sie allen Unterabfragen Zeitprädikate hinzu.

Typ: QueryTemporalRange Objekt

Erforderlich: Nein

UnloadPartitionCount

Gibt die Partitionen an, die durch den Vorgang erstellt wurden. `Unload`

Type: Long

Erforderlich: Nein

UnloadWrittenBytes

Gibt die Größe in Byte an, die durch den `Unload` Vorgang geschrieben wurde.

Type: Long

Erforderlich: Nein

UnloadWrittenRows

Gibt die Zeilen an, die von der `Unload` Abfrage geschrieben wurden.

Type: Long

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen dazu, wie Sie dies API in einer der sprachspezifischen Sprachen verwenden können AWS SDKs, finden Sie im Folgenden:

- AWS SDKfür C++

- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

QuerySpatialCoverage
Bedienung: Amazon Timestream Query

Bietet Einblicke in die räumliche Abdeckung der Abfrage, einschließlich der Tabelle mit suboptimaler (maximaler) räumlicher Bereinigung. Diese Informationen können Ihnen dabei helfen, Bereiche zu identifizieren, in denen Ihre Partitionierungsstrategie verbessert werden muss, um die räumliche Kürzung zu verbessern

Sie können die `QuerySpatialCoverage` Informationen beispielsweise wie folgt verwenden:

- Fügen Sie measure_name hinzu oder verwenden Sie kundenspezifische Partitionsschlüssel () CDPK -Prädikate.
- Wenn Sie die vorhergehende Aktion bereits ausgeführt haben, entfernen Sie Funktionen oder Klauseln, wie z. `LIKE`

Inhalt

Max

Bietet Einblicke in die räumliche Abdeckung der ausgeführten Abfrage und in die Tabelle mit der ineffizientesten räumlichen Bereinigung.

- `Value`— Das maximale Verhältnis der räumlichen Abdeckung.
- `TableArn`— Der Amazon-Ressourcenname (ARN) der Tabelle mit suboptimaler räumlicher Bereinigung.
- `PartitionKey`— Der für die Partitionierung verwendete Partitionsschlüssel. Dabei kann es sich um einen Standard `measure_name` - oder einen Partitionsschlüssel handeln. CDPK

Typ: QuerySpatialCoverageMax Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2

- AWS SDKfür Ruby V3

QuerySpatialCoverageMax
Bedienung: Amazon Timestream Query

Bietet Einblicke in die Tabelle mit dem suboptimalsten räumlichen Bereich, der von Ihrer Abfrage gescannt wurde.

Inhalt

PartitionKey

Der für die Partitionierung verwendete Partitionsschlüssel. Dabei kann es sich um einen standardmäßigen `measure_name` oder einen benutzerdefinierten Partitionsschlüssel handeln.

Typ: Zeichenfolgen-Array

Erforderlich: Nein

TableArn

Der Amazon-Ressourcenname (ARN) der Tabelle mit der suboptimalsten räumlichen Bereinigung.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Nein

Value

Das maximale Verhältnis der räumlichen Abdeckung.

Type: Double

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2

- AWS SDKfür Ruby V3

QueryStatus
Bedienung: Amazon Timestream Query

Informationen über den Status der Abfrage, einschließlich des Fortschritts und der gescannten Byte.

Inhalt

CumulativeBytesMetered

Die von der Abfrage gescannte Datenmenge in Byte, für die Ihnen eine Gebühr berechnet wird. Dies ist eine kumulative Summe und stellt die gesamte Datenmenge dar, die Ihnen seit dem Start der Abfrage in Rechnung gestellt wurde. Die Gebühr wird nur einmal erhoben, und zwar entweder, wenn die Abfrage abgeschlossen ist, oder wenn die Abfrage abgebrochen wird.

Type: Long

Erforderlich: Nein

CumulativeBytesScanned

Die von der Abfrage gescannte Datenmenge in Byte. Dies ist eine kumulative Summe und stellt die Gesamtmenge der seit dem Start der Abfrage gescannten Byte dar.

Type: Long

Erforderlich: Nein

ProgressPercentage

Der Fortschritt der Abfrage, ausgedrückt als Prozentsatz.

Type: Double

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser Funktion API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

QueryTemporalRange
Bedienung: Amazon Timestream Query

Bietet Einblicke in den zeitlichen Bereich der Abfrage, einschließlich der Tabelle mit dem größten (maximalen) Zeitraum.

Inhalt

Max

Beinhaltet die folgenden Eigenschaften, die Einblicke in die Tabelle mit der schlechtesten Leistung auf der Zeitachse bieten:

- `Value`— Die maximale Dauer in Nanosekunden zwischen dem Start und dem Ende der Abfrage.
- `TableArn`— Der Amazon-Ressourcenname (ARN) der Tabelle, die mit dem größten Zeitraum abgefragt wird.

Typ: QueryTemporalRangeMax Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen finden Sie im Folgenden AWS SDKs:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

QueryTemporalRangeMax
Bedienung: Amazon Timestream Query

Bietet Einblicke in die Tabelle mit der suboptimalsten zeitlichen Bereinigung, die im Rahmen Ihrer Abfrage gescannt wurde.

Inhalt

TableArn

Der Amazon-Ressourcenname (ARN) der Tabelle, die mit dem größten Zeitraum abgefragt wird.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Nein

Value

Die maximale Dauer in Nanosekunden zwischen dem Start und dem Ende der Abfrage.

Type: Long

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

Row
Bedienung: Amazon Timestream Query

Stellt eine einzelne Zeile in den Abfrageergebnissen dar.

Inhalt

Data

Liste der Datenpunkte in einer einzelnen Zeile der Ergebnismenge.

Typ: Array von Datum-Objekten

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

S3Configuration
Bedienung: Amazon Timestream Query

Details zum S3-Speicherort für Fehlerberichte, die sich aus der Ausführung einer Abfrage ergeben.

Inhalt

BucketName

Den Namen des S3-Buckets, unter dem Fehlerberichte erstellt werden.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 63 Zeichen.

Pattern: `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Erforderlich: Ja

EncryptionOption

Optionen für die Verschlüsselung gespeicherter Daten der Fehlerberichte. Wenn keine Verschlüsselungsoption angegeben ist, wählt Timestream SSE _S3 als Standard.

Typ: Zeichenfolge

Zulässige Werte: `SSE_S3 | SSE_KMS`

Erforderlich: Nein

ObjectKeyPrefix

Präfix für den Fehlerberichts-Schlüssel. Timestream fügt dem Fehlerberichts-Pfad standardmäßig das folgende Präfix hinzu.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Die maximale Länge beträgt 896.

Pattern: `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

S3ReportLocation
Bedienung: Amazon Timestream Query

Speicherort des S3-Berichts für den geplanten Abfragelauf.

Inhalt

BucketName

Name des S3 Buckets.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 3. Maximale Länge beträgt 63 Zeichen.

Pattern: `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Erforderlich: Nein

ObjectKey

S3-Schlüssel.

Typ: Zeichenfolge

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

ScheduleConfiguration
Bedienung: Amazon Timestream Query

Konfiguration des Zeitplans der Abfrage.

Inhalt

ScheduleExpression

Ein Ausdruck, der angibt, wann der geplante Abfragelauf ausgelöst werden soll. Dies kann ein Cron-Ausdruck oder ein Ratenausdruck sein.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

ScheduledQuery
Bedienung: Amazon Timestream Query

Geplante Abfrage

Inhalt

Arn

Der Amazon-Ressourcenname.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Ja

Name

Der Name der geplanten Abfrage.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Erforderlich: Ja

State

Status der geplanten Abfrage.

Typ: Zeichenfolge

Zulässige Werte: `ENABLED | DISABLED`

Erforderlich: Ja

CreationTime

Die Erstellungszeit der geplanten Abfrage.

Typ: Zeitstempel

Erforderlich: Nein

ErrorReportConfiguration

Konfiguration für die Berichterstattung über geplante Abfragefehler.

Typ: ErrorReportConfiguration Objekt

Erforderlich: Nein

LastRunStatus

Status des letzten geplanten Abfragelaufs.

Typ: Zeichenfolge

Zulässige Werte: `AUTO_TRIGGER_SUCCESS | AUTO_TRIGGER_FAILURE | MANUAL_TRIGGER_SUCCESS | MANUAL_TRIGGER_FAILURE`

Erforderlich: Nein

NextInvocationTime

Das nächste Mal, wenn die geplante Abfrage ausgeführt werden soll.

Typ: Zeitstempel

Erforderlich: Nein

PreviousInvocationTime

Das letzte Mal, als die geplante Abfrage ausgeführt wurde.

Typ: Zeitstempel

Erforderlich: Nein

TargetDestination

Zieldatenquelle, in die das endgültige geplante Abfrageergebnis geschrieben wird.

Typ: TargetDestination Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

ScheduledQueryDescription
Bedienung: Amazon Timestream Query

Struktur, die eine geplante Abfrage beschreibt.

Inhalt

Arn

Geplante AbfrageARN.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Ja

Name

Name der geplanten Abfrage.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 64 Zeichen.

Pattern: `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Erforderlich: Ja

NotificationConfiguration

Konfiguration der Benachrichtigung.

Typ: NotificationConfiguration Objekt

Erforderlich: Ja

QueryString

Die auszuführende Abfrage.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge von 262144.

Erforderlich: Ja

ScheduleConfiguration

Konfiguration des Zeitplans der Abfrage.

Typ: ScheduleConfiguration Objekt

Erforderlich: Ja

State

Status der geplanten Abfrage.

Typ: Zeichenfolge

Zulässige Werte: `ENABLED | DISABLED`

Erforderlich: Ja

CreationTime

Erstellungszeit der geplanten Abfrage.

Typ: Zeitstempel

Erforderlich: Nein

ErrorReportConfiguration

Konfiguration zur Fehlerberichterstattung für die geplante Abfrage.

Typ: ErrorReportConfiguration Objekt

Erforderlich: Nein

KmsKeyId

Ein Kunde hat den KMS Schlüssel zur Verschlüsselung der geplanten Abfrageressource bereitgestellt.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Nein

LastRunSummary

Laufzeitübersicht für den letzten geplanten Abfragelauf.

Typ: ScheduledQueryRunSummary Objekt

Erforderlich: Nein

NextInvocationTime

Das nächste Mal, wenn die geplante Ausführung der geplanten Abfrage geplant ist.

Typ: Zeitstempel

Erforderlich: Nein

PreviousInvocationTime

Das letzte Mal, als die Abfrage ausgeführt wurde.

Typ: Zeitstempel

Erforderlich: Nein

RecentlyFailedRuns

Laufzeitübersicht für die letzten fünf fehlgeschlagenen geplanten Abfrageläufe.

Typ: Array von ScheduledQueryRunSummary-Objekten

Erforderlich: Nein

ScheduledQueryExecutionRoleArn

IAMRolle, die Timestream verwendet, um die Zeitplanabfrage auszuführen.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Nein

TargetConfiguration

Geplante Abfrage-Ziel-Speicherkonfiguration.

Typ: TargetConfiguration Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser Funktion API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

ScheduledQueryInsights
Bedienung: Amazon Timestream Query

Kapselt Einstellungen für die Aktivierung `QueryInsights` auf einem. `ExecuteScheduledQueryRequest`

Inhalt

Mode

Stellt die folgenden Aktivierungsmodi zur Verfügung: `ScheduledQueryInsights`

- `ENABLED_WITH_RATE_CONTROL`— Aktiviert `ScheduledQueryInsights` die Abfragen, die gerade verarbeitet werden. Dieser Modus beinhaltet auch einen Mechanismus zur Ratensteuerung, der die `QueryInsights` Funktion auf eine Abfrage pro Sekunde beschränkt (QPS).
- `DISABLED`— Deaktiviert. `ScheduledQueryInsights`

Typ: Zeichenfolge

Zulässige Werte: `ENABLED_WITH_RATE_CONTROL | DISABLED`

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

ScheduledQueryInsightsResponse
Bedienung: Amazon Timestream Query

Bietet verschiedene Einblicke und Metriken in Bezug auf `ExecuteScheduledQueryRequest` das, was ausgeführt wurde.

Inhalt

OutputBytes

Gibt die Größe des Abfrageergebnissatzes in Byte an. Sie können diese Daten verwenden, um zu überprüfen, ob sich die Ergebnismenge im Rahmen der Abfrageoptimierung geändert hat.

Type: Long

Erforderlich: Nein

OutputRows

Gibt die Gesamtzahl der Zeilen an, die als Teil des Abfrageergebnissatzes zurückgegeben wurden. Sie können diese Daten verwenden, um zu überprüfen, ob sich die Anzahl der Zeilen in der Ergebnismenge im Rahmen der Abfrageoptimierung geändert hat.

Type: Long

Erforderlich: Nein

QuerySpatialCoverage

Bietet Einblicke in die räumliche Abdeckung der Abfrage, einschließlich der Tabelle mit suboptimaler (maximaler) räumlicher Bereinigung. Diese Informationen können Ihnen dabei helfen, Bereiche zu identifizieren, in denen Ihre Partitionierungsstrategie verbessert werden muss, um die räumliche Kürzung zu verbessern.

Typ: QuerySpatialCoverage Objekt

Erforderlich: Nein

QueryTableCount

Gibt die Anzahl der Tabellen in der Abfrage an.

Type: Long

Erforderlich: Nein

QueryTemporalRange

Bietet Einblicke in den zeitlichen Bereich der Abfrage, einschließlich der Tabelle mit dem größten (maximalen) Zeitbereich. Im Folgenden sind einige der möglichen Optionen zur Optimierung der zeitbasierten Bereinigung aufgeführt:

- Fehlende Zeitprädikate hinzufügen.
- Entferne Funktionen rund um die Zeitprädikate.
- Fügen Sie allen Unterabfragen Zeitprädikate hinzu.

Typ: QueryTemporalRange Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen dazu, wie Sie dies API in einer der sprachspezifischen Sprachen AWS SDKs verwenden können, finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

ScheduledQueryRunSummary
Bedienung: Amazon Timestream Query

Zusammenfassung für die geplante Abfrage ausführen

Inhalt

ErrorReportLocation

S3-Speicherort für den Fehlerbericht.

Typ: ErrorReportLocation Objekt

Erforderlich: Nein

ExecutionStats

Laufzeitstatistiken für einen geplanten Lauf.

Typ: ExecutionStats Objekt

Erforderlich: Nein

FailureReason

Fehlermeldung für die geplante Abfrage im Falle eines Fehlers. Möglicherweise müssen Sie sich den Fehlerbericht ansehen, um detailliertere Fehlergründe zu erhalten.

Typ: Zeichenfolge

Erforderlich: Nein

InvocationTime

InvocationTime für diesen Lauf. Dies ist der Zeitpunkt, zu dem die Ausführung der Abfrage geplant ist. Der Parameter `@scheduled_runtime` kann in der Abfrage verwendet werden, um den Wert abzurufen.

Typ: Zeitstempel

Erforderlich: Nein

QueryInsightsResponse

Bietet verschiedene Einblicke und Metriken im Zusammenhang mit der Zusammenfassung der Ausführung der geplanten Abfrage.

Typ: ScheduledQueryInsightsResponse Objekt

Erforderlich: Nein

RunStatus

Der Status eines geplanten Abfragelaufs.

Typ: Zeichenfolge

Zulässige Werte: `AUTO_TRIGGER_SUCCESS | AUTO_TRIGGER_FAILURE | MANUAL_TRIGGER_SUCCESS | MANUAL_TRIGGER_FAILURE`

Erforderlich: Nein

TriggerTime

Die tatsächliche Uhrzeit, zu der die Abfrage ausgeführt wurde.

Typ: Zeitstempel

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung dieser Funktion API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

SelectColumn
Bedienung: Amazon Timestream Query

Details der Spalte, die von der Abfrage zurückgegeben wird.

Inhalt

Aliased

Stimmt, wenn der Spaltenname von der Abfrage mit einem Alias versehen wurde. Andernfalls falsch.

Typ: Boolesch

Erforderlich: Nein

DatabaseName

Datenbank, die diese Spalte enthält.

Typ: Zeichenfolge

Erforderlich: Nein

Name

Name der Spalte.

Typ: Zeichenfolge

Erforderlich: Nein

TableName

Tabelle in der Datenbank, die diese Spalte enthält.

Typ: Zeichenfolge

Erforderlich: Nein

Type

Enthält den Datentyp einer Spalte in einer Abfrageergebnismenge. Der Datentyp kann skalar oder komplex sein. Die unterstützten skalaren Datentypen sind Integer, Boolean, String, Double, Timestamp, Date, Time und Intervalle. Die unterstützten komplexen Datentypen sind Arrays, Zeilen und Zeitreihen.

Typ: Type Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen finden Sie im Folgenden AWS SDKs:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

SnsConfiguration
Bedienung: Amazon Timestream Query

Einzelheiten SNS dazu sind erforderlich, um die Benachrichtigung zu senden.

Inhalt

TopicArn

SNSThemaARN, an das die Benachrichtigungen zum geplanten Abfragestatus gesendet werden.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 2048 Zeichen.

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

Tag
Bedienung: Amazon Timestream Query

Ein Tag ist eine Bezeichnung, die Sie einer 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 Timestream-Datenbanktabelle zuweisen, z. B. nach Zweck, Besitzer oder Umgebung.

Inhalt

Key

Der Schlüssel des Tags. Bei Tag-Schlüsseln wird zwischen Groß- und Kleinschreibung unterschieden.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge beträgt 1 Zeichen. Maximale Länge beträgt 128 Zeichen.

Erforderlich: Ja

Value

Der Wert des Tags. Bei Tag-Werten wird zwischen Groß- und Kleinschreibung unterschieden und sie können Null sein.

Typ: Zeichenfolge

Längenbeschränkungen: Minimale Länge von 0. Maximale Länge beträgt 256 Zeichen.

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

TargetConfiguration
Bedienung: Amazon Timestream Query

Konfiguration, die zum Schreiben der Ausgabe einer Abfrage verwendet wird.

Inhalt

TimestreamConfiguration

Konfiguration zum Schreiben von Daten in die Timestream-Datenbank und -Tabelle.

Typ: TimestreamConfiguration Objekt

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

TargetDestination
Bedienung: Amazon Timestream Query

Zieldetails zum Schreiben von Daten für eine Zieldatenquelle. Die aktuell unterstützte Datenquelle ist Timestream.

Inhalt

TimestreamDestination

Fragen Sie die Ergebnisdetails für die Timestream-Datenquelle ab.

Typ: TimestreamDestination Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

TimeSeriesDataPoint
Bedienung: Amazon Timestream Query

Der Zeitreihen-Datentyp stellt die Werte einer Kennzahl im Zeitverlauf dar. Eine Zeitreihe ist eine Reihe von Zeilen mit Zeitstempeln und Kennzahlwerten, wobei die Zeilen in aufsteigender Reihenfolge sortiert sind. A TimeSeriesDataPoint ist ein einzelner Datenpunkt in der Zeitreihe. Es stellt ein Tupel von (Zeit, Messwert) in einer Zeitreihe dar.

Inhalt

Time

Der Zeitstempel, zu dem der Messwert erfasst wurde.

Typ: Zeichenfolge

Erforderlich: Ja

Value

Der Messwert für den Datenpunkt.

Typ: Datum Objekt

Erforderlich: Ja

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

TimestreamConfiguration
Bedienung: Amazon Timestream Query

Konfiguration zum Schreiben von Daten in die Timestream-Datenbank und -Tabelle. Diese Konfiguration ermöglicht es dem Benutzer, die Auswahl-Spalten des Abfrage-Ergebnisses den Spalten der Zieltabelle zuzuordnen.

Inhalt

DatabaseName

Name der Timestream-Datenbank, in die das Abfrage-Ergebnis geschrieben wird.

Typ: Zeichenfolge

Erforderlich: Ja

DimensionMappings

Dies ermöglicht die Zuweisung von Spalten aus dem Abfrage-Ergebnis zur Dimension in der Ziel-Tabelle.

Typ: Array von DimensionMapping-Objekten

Erforderlich: Ja

TableName

Name der Timestream-Tabelle, in die das Abfrage-Ergebnis geschrieben wird. Die Tabelle sollte sich in derselben Datenbank befinden, die in der Timestream-Konfiguration bereitgestellt wird.

Typ: Zeichenfolge

Erforderlich: Ja

TimeColumn

Spalte aus dem Abfrage-Ergebnis, die als Zeit-Spalte in der Zieltabelle verwendet werden sollte. Der Spaltentyp dafür sollte seinTIMESTAMP.

Typ: Zeichenfolge

Erforderlich: Ja

MeasureNameColumn

Name der Messwert-Spalte.

Typ: Zeichenfolge

Erforderlich: Nein

MixedMeasureMappings

Gibt an, wie Kennzahlen an Datensätze mit mehrfachen Messwerten zugeordnet werden.

Typ: Array von MixedMeasureMapping-Objekten

Array-Mitglieder: Die Mindestanzahl beträgt 1 Element.

Erforderlich: Nein

MultiMeasureMappings

Zuweisungen mit mehreren Messwerten.

Typ: MultiMeasureMappings Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen AWS SDKs finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

TimestreamDestination
Bedienung: Amazon Timestream Query

Ziel für die geplante Abfrage.

Inhalt

DatabaseName

Name der Timestream-Datenbank.

Typ: Zeichenfolge

Erforderlich: Nein

TableName

Name der Timestream-Tabelle.

Typ: Zeichenfolge

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen zur Verwendung API in einer der sprachspezifischen Sprachen finden Sie im AWS SDKs Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

Type
Bedienung: Amazon Timestream Query

Enthält den Datentyp einer Spalte in einem Abfrageergebnissatz. Der Datentyp kann skalar oder komplex sein. Die unterstützten skalaren Datentypen sind Integer, Boolean, String, Double, Timestamp, Date, Time und Intervalle. Die unterstützten komplexen Datentypen sind Arrays, Zeilen und Zeitreihen.

Inhalt

ArrayColumnInfo

Gibt an, ob es sich bei der Spalte um ein Array handelt.

Typ: ColumnInfo Objekt

Erforderlich: Nein

RowColumnInfo

Gibt an, ob es sich bei der Spalte um eine Zeile handelt.

Typ: Array von ColumnInfo-Objekten

Erforderlich: Nein

ScalarType

Gibt an, ob die Spalte vom Typ Zeichenfolge, Ganzzahl, Boolean, Double, Zeitstempel, Datum, Uhrzeit ist. Weitere Informationen finden Sie unter Unterstützte Datentypen.

Typ: Zeichenfolge

Zulässige Werte: `VARCHAR | BOOLEAN | BIGINT | DOUBLE | TIMESTAMP | DATE | TIME | INTERVAL_DAY_TO_SECOND | INTERVAL_YEAR_TO_MONTH | UNKNOWN | INTEGER`

Erforderlich: Nein

TimeSeriesMeasureValueColumnInfo

Gibt an, ob es sich bei der Spalte um einen Zeitreihen-Datentyp handelt.

Typ: ColumnInfo Objekt

Erforderlich: Nein

Weitere Informationen finden Sie unter:

Weitere Informationen dazu, wie Sie dies API in einer der sprachspezifischen Sprachen verwenden können AWS SDKs, finden Sie im Folgenden:

- AWS SDKfür C++
- AWS SDKfür Java V2
- AWS SDKfür Ruby V3

Häufige Fehler

In diesem Abschnitt werden die Fehler aufgeführt, die bei den API Aktionen aller AWS Dienste häufig auftreten. Informationen zu Fehlern, die sich auf eine API Aktion für diesen Dienst beziehen, finden Sie im Thema zu dieser API Aktion.

AccessDeniedException

Sie haben keinen ausreichenden Zugriff zum Durchführen dieser Aktion.

HTTPStatuscode: 400

IncompleteSignature

Die Signatur der Anfrage entspricht nicht den AWS Standards.

HTTPStatuscode: 400

InternalFailure

Die Anforderungsverarbeitung ist fehlgeschlagen, da ein unbekannter Fehler, eine Ausnahme oder ein Fehler aufgetreten ist.

HTTPStatuscode: 500

InvalidAction

Die angeforderte Aktion oder Operation ist ungültig. Überprüfen Sie, ob die Aktion ordnungsgemäß eingegeben wurde.

HTTPStatuscode: 400

InvalidClientTokenId

Das angegebene X.509-Zertifikat oder die angegebene AWS Zugriffsschlüssel-ID ist in unseren Aufzeichnungen nicht vorhanden.

HTTPStatuscode: 403

NotAuthorized

Sie haben keine Berechtigung zum Ausführen dieser Aktion.

HTTPStatuscode: 400

OptInRequired

Für die AWS Zugriffsschlüssel-ID ist ein Abonnement für den Dienst erforderlich.

HTTPStatuscode: 403

RequestExpired

Die Anfrage erreichte den Service mehr als 15 Minuten nach dem Datumsstempel auf der Anfrage oder mehr als 15 Minuten nach dem Ablaufdatum der Anfrage (z. B. bei vorsignierter AnfrageURLs), oder der Datumsstempel auf der Anfrage liegt mehr als 15 Minuten in der future.

HTTPStatuscode: 400

ServiceUnavailable

Die Anforderung ist aufgrund eines temporären Fehlers des Servers fehlgeschlagen.

HTTPStatuscode: 503

ThrottlingException

Die Anforderung wurde aufgrund der Drosselung von Anforderungen abgelehnt.

HTTPStatuscode: 400

ValidationError

Die Eingabe erfüllt die von einem AWS Dienst angegebenen Einschränkungen nicht.

HTTPStatuscode: 400

Geläufige Parameter

Die folgende Liste enthält die Parameter, die alle Aktionen zum Signieren von Signature-Version-4-Anforderungen mit einer Abfragezeichenfolge verwenden. Alle aktionsspezifischen Parameter werden im Thema für diese Aktion aufgelistet. Weitere Informationen zu Signature Version 4 finden Sie unter AWS APISignieranfragen im IAMBenutzerhandbuch.

Action

Die auszuführende Aktion.

Typ: Zeichenfolge

Erforderlich: Ja

Version

Die API Version, für die die Anfrage geschrieben wurde, ausgedrückt im Format YYYY-MM-DD.

Typ: Zeichenfolge

Erforderlich: Ja

X-Amz-Algorithm

Der Hashalgorithmus, den Sie zum Erstellen der Anforderungssignatur verwendet haben.

Bedingung: Geben Sie diesen Parameter an, wenn Sie Authentifizierungsinformationen in eine Abfragezeichenfolge statt in den HTTP Autorisierungsheader aufnehmen.

Typ: Zeichenfolge

Zulässige Werte: `AWS4-HMAC-SHA256`

Required: Conditional

X-Amz-Credential

Der Wert des Anmeldeinformationsumfangs. Dabei handelt es sich um eine Zeichenfolge, die Ihren Zugriffsschlüssel, das Datum, die gewünschte Region und eine Zeichenfolge zur Beendigung („aws4_request“) beinhaltet. Der Wert wird im folgenden Format ausgedrückt: access_key//region YYYYMMDD/service /aws4_request.

Weitere Informationen finden Sie im Benutzerhandbuch unter Erstellen einer signierten Anfrage. AWS API IAM

Bedingung: Geben Sie diesen Parameter an, wenn Sie Authentifizierungsinformationen in eine Abfragezeichenfolge statt in den HTTP Autorisierungsheader aufnehmen.

Typ: Zeichenfolge

Required: Conditional

X-Amz-Date

Das Datum, das zum Erstellen der Signatur verwendet wird. Das Format muss das Standardformat ISO 8601 (YYYYMMDD'T' 'ZHHMMSS') sein. Beispielsweise ist die folgende Datums- und Uhrzeitangabe ein gültiger X-Amz-Date Wert:`20120325T120000Z`.

Bedingung: X-Amz-Date ist für alle Anfragen optional; sie kann verwendet werden, um das Datum zu überschreiben, das für das Signieren von Anfragen verwendet wird. Wenn der Date-Header im ISO 8601-Standardformat angegeben ist, X-Amz-Date ist dies nicht erforderlich. Wenn verwendet X-Amz-Date wird, überschreibt er immer den Wert des Date-Headers. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter Elemente einer AWS API Anforderungssignatur.

Typ: Zeichenfolge

Required: Conditional

X-Amz-Security-Token

Das temporäre Sicherheitstoken, das durch einen Aufruf von AWS Security Token Service (AWS STS) abgerufen wurde. Eine Liste der Dienste, die temporäre Sicherheitsanmeldedaten von unterstützen AWS STSAWS-Services , finden Sie IAM im IAMBenutzerhandbuch unter Diese Dienste können verwendet werden.

Bedingung: Wenn Sie temporäre Anmeldeinformationen von verwenden AWS STS, müssen Sie das Sicherheitstoken angeben.

Typ: Zeichenfolge

Required: Conditional

X-Amz-Signature

Gibt die hex-codierte Signatur an, die aus der zu signierenden Zeichenfolge und dem abgeleiteten Signaturschlüssel berechnet wurde.

Bedingung: Geben Sie diesen Parameter an, wenn Sie Authentifizierungsinformationen in eine Abfragezeichenfolge statt in den HTTP Autorisierungsheader aufnehmen.

Typ: Zeichenfolge

Required: Conditional

X-Amz-SignedHeaders

Gibt alle HTTP Header an, die als Teil der kanonischen Anfrage enthalten waren. Weitere Informationen zur Angabe signierter Header finden Sie unter Erstellen einer signierten AWS API Anfrage im Benutzerhandbuch. IAM

Bedingung: Geben Sie diesen Parameter an, wenn Sie Authentifizierungsinformationen in eine Abfragezeichenfolge statt in den HTTP Autorisierungsheader aufnehmen.

Typ: Zeichenfolge

Required: Conditional

Dokumentverlauf

Änderung	Beschreibung	Datum
Update nur für die Dokumentation	Das Thema Kontingente wurde aktualisiert, um die Standardkontingente und Systemlimits voneinander zu trennen.	22. Oktober 2024
Amazon Timestream unterstützt jetzt Query-Insights	Timestream bietet jetzt Unterstützung für die Query Insights-Funktion, mit der Sie Ihre Abfragen optimieren, deren Leistung verbessern und Kosten senken können.	22. Oktober 2024
Amazon Timestream for InfluxDB aktualisiert eine bestehende Richtlinie.	Amazon Timestream for InfluxDB hat die `ec2:DescribeRouteTables` Aktion zur bestehenden `AmazonTimestreamInfluxDBFul`	8. Oktober 2024

	`lAccess` verwalteten Richtlinie zur Beschreib ung Ihrer Routing-Tabellen hinzugefügt	
`AmazonTimestreamRe adOnlyAccess` — Aktualisi erung einer bestehenden Richtlinie	Timestream for LiveAnaly tics hat der `AmazonTim estreamReadOnlyAcc ess` verwalteten Richtlinie die `DescribeAccountSet tings` Berechtigung zur Beschreibung von AWS-Konto Einstellungen hinzugefügt.	3. Juni 2024
Amazon Timestream unterstüt zt LiveAnalytics derzeit Timestream Compute Units () TCUs	Amazon Timestream bietet LiveAnalytics derzeit Unterstüt zung für Timestream Compute Units (TCUs) zur Messung der Rechenkapazität, die für Ihre Abfrageanforderungen zugewiesen wurde.	29. April 2024

Neue Richtlinien wurden hinzugefügt	Amazon Timestream for InfluxDB hat zwei neue Richtlinien hinzugefügt: Eine, die es dem Service ermöglicht, Netzwerkschnittstellen und Sicherheitsgruppen in Ihrem Konto zu verwalten. Weitere Informationen finden Sie unter. AmazonTimestreamInfluxDBServiceRolePolicy Eine weitere, die vollen Administratorzugriff zum Erstellen, Aktualisieren, Löschen und Auflisten von Amazon Timestream InfluxDB-Instances sowie zum Erstellen und Auflisten von Parametergruppen bietet. Weitere Informationen finden Sie unter. AmazonTimestreamInfluxDBFullAccess	14. März 2024
Amazon Timestream für InfluxDB ist jetzt allgemein verfügbar.	Diese Dokumentation behandelt die erste Version von Amazon Timestream für InfluxDB.	14. März 2024

Amazon Timestream for LiveAnalytics Query-Ereignisse sind verfügbar in AWS CloudTrail	Amazon Timestream veröffentlicht LiveAnalytics vorerst API Abfragedatenereignisse für. AWS CloudTrail Kunden können alle API Query-Anfragen, die in ihren AWS Konten gestellt wurden, prüfen und sich Informationen darüber anzeigen lassen, welcher IAM Benutzer/welche Rolle die Anfrage gestellt hat, wann die Anfrage gestellt wurde, welche Datenbanken und Tabellen abgefragt wurden und welche Abfrage-ID der Anfrage lautet.	12. September 2023
Amazon Timestream für LiveAnalytics UNLOAD	Amazon Timestream unterstützt LiveAnalytics derzeit UNLOAD den Export von Abfrageergebnissen nach S3.	12. Mai 2023
Amazon Timestream für die LiveAnalytics Aktualisierung einer bestehenden Richtlinie.	Batchladeberechtigungen wurden zu einer verwalteten Richtlinie hinzugefügt.	24. Februar 2023
Amazon Timestream für LiveAnalytics Batch-Load.	Amazon Timestream unterstützt LiveAnalytics derzeit Batch-Load-Funktionen.	24. Februar 2023
Amazon Timestream unterstützt LiveAnalytics AWS Backup derzeit.	Amazon Timestream unterstützt LiveAnalytics AWS Backup derzeit.	14. Dezember 2022

Amazon Timestream für LiveAnalytics Aktualisierungen AWS verwalteter Richtlinien	Neue Informationen zu AWS verwalteten Richtlinien und Amazon Timestream für LiveAnalytics, einschließlich Aktualisierungen vorhandener verwalteter Richtlinien.	29. November 2021
Amazon Timestream for LiveAnalytics unterstützt geplante Abfragen	Amazon Timestream unterstützt LiveAnalytics derzeit die Ausführung einer Abfrage in Ihrem Namen, basierend auf einem Zeitplan.	29. November 2021
Amazon Timestream for LiveAnalytics unterstützt Magnetic Store.	Amazon Timestream unterstützt LiveAnalytics derzeit die Verwendung von Magnetspeicher für Ihre Tabellenschreibvorgänge.	29. November 2021
Amazon Timestream für Datensätze LiveAnalytics mit mehreren Messwerten.	Amazon Timestream unterstützt LiveAnalytics derzeit ein kompakteres Format zum Speichern Ihrer Zeitreihendaten.	29. November 2021
Amazon Timestream für LiveAnalytics Aktualisierungen AWS verwalteter Richtlinien	Neue Informationen zu AWS verwalteten Richtlinien und Amazon Timestream für LiveAnalytics, einschließlich Aktualisierungen vorhandener verwalteter Richtlinien.	24. Mai 2021
Amazon Timestream for LiveAnalytics ist jetzt in der Region Europa (Frankfurt) verfügbar.	Amazon Timestream for LiveAnalytics ist jetzt in der Region Europa (Frankfurt) allgemein verfügbar (`eu-central-1`).	23. April 2021

Amazon Timestream for LiveAnalytics unterstützt jetzt VPC Endpoints ()AWS PrivateLink.	Amazon Timestream unterstützt LiveAnalytics derzeit die Verwendung von VPC Endpunkten ()AWS PrivateLink.	23. März 2021
Amazon Timestream unterstützt jetzt tabellenübergreifende Abfragen.	Sie können Amazon Timestream verwenden LiveAnalytics , um tabellenübergreifende Abfragen auszuführen.	10. Februar 2021
Amazon Timestream unterstützt LiveAnalytics derzeit erweiterte Statistiken zur Abfrageausführung.	Amazon Timestream unterstützt LiveAnalytics derzeit erweiterte Statistiken zur Abfrageausführung, wie z. B. die Menge der gescannten Daten.	10. Februar 2021
Amazon Timestream unterstützt LiveAnalytics derzeit erweiterte Zeitreihenfunktionen.	Sie können Amazon Timestream für verwenden LiveAnalytics , um SQL Abfragen mit erweiterten Zeitreihenfunktionen wie Ableitungen, Integralen und Korrelationen auszuführen.	10. Februar 2021
Amazon Timestream for LiveAnalytics ist jetzt HIPAA und ISO PCI konform.	Sie können Amazon Timestream jetzt für Workloads verwenden, LiveAnalytics für die eine HIPAAISO, und PCI -konforme Infrastruktur erforderlich ist.	27. Januar 2021

Amazon Timestream unterstützt LiveAnalytics derzeit Open-Source-Telegraf und Grafana.	Sie können jetzt Telegraf, den Plug-in-gesteuerten Open-Source-Serveragenten für die Erfassung und Berichterstattung von Metriken, und Grafana, die Open-Source-Analyse- und Überwachungsplattform für Datenbanken, mit Amazon Timestream für verwenden. LiveAnalytics	25. November 2020
Amazon Timestream for LiveAnalytics ist jetzt allgemein verfügbar.	Diese Dokumentation behandelt die erste Version von Amazon Timestream für LiveAnalytics.	30. September 2020

Was ist Timestream for InfluxDB?

Amazon Timestream for InfluxDB ist eine verwaltete Zeitreihen-Datenbank-Engine, die es Anwendungsentwicklern und DevOps Teams leicht macht, InfluxDB-Datenbanken für Echtzeit-Zeitreihenanwendungen mithilfe von Open Source AWS auszuführen. APIs Mit Amazon Timestream for InfluxDB ist es einfach, Zeitreihen-Workloads einzurichten, zu betreiben und zu skalieren, die Anfragen mit einer Antwortzeit im einstelligen Millisekundenbereich beantworten können.

Amazon Timestream for InfluxDB bietet Ihnen Zugriff auf die Funktionen der vertrauten Open-Source-Version von InfluxDB in der 2.x-Filiale. Das bedeutet, dass der Code, die Anwendungen und Tools, die Sie heute bereits mit Ihren vorhandenen InfluxDB-Open-Source-Datenbanken verwenden, nahtlos mit Amazon Timestream for InfluxDB funktionieren sollten. Amazon Timestream for InfluxDB kann Ihre Datenbank automatisch sichern und Ihre Datenbanksoftware mit der neuesten Version auf dem neuesten Stand halten. Darüber hinaus macht es Amazon Timestream for InfluxDB einfach, die Replikation zu verwenden, um die Datenbankverfügbarkeit zu verbessern und die Datenbeständigkeit zu verbessern. Wie bei allen AWS Services sind keine Vorabinvestitionen erforderlich, und Sie zahlen nur für die Ressourcen, die Sie nutzen.

DB-Instances

Eine DB-Instance ist eine isolierte Datenbankumgebung, die in der Cloud ausgeführt wird. Es ist der grundlegende Baustein von Amazon Timestream für InfluxDB. Eine DB-Instance kann mehrere vom Benutzer erstellte Datenbanken (oder Organisationen und Buckets im Fall von InfluxDb 2.x-Datenbanken) enthalten und kann mit denselben Client-Tools und -Anwendungen aufgerufen werden, die Sie für den Zugriff auf eine eigenständige, selbstverwaltete InfluxDB-Instance verwenden könnten. DB-Instances lassen sich einfach mit den AWS Befehlszeilentools, Amazon Timestream API InfluxDB-Operationen oder dem erstellen und ändern. AWS Management Console

> **Note**
>
> Amazon Timestream for InfluxDB unterstützt den Zugriff auf Datenbanken mithilfe der API Influx-Operationen und der Influx-Benutzeroberfläche. Amazon Timestream for InfluxDB erlaubt keinen direkten Hostzugriff.

Sie können bis zu 40 Amazon Timestream für InfluxDB-Instances haben.

Jede DB-Instance hat einen DB-Instance-Namen. Dieser vom Kunden bereitgestellte Name identifiziert die DB-Instance eindeutig, wenn sie mit Amazon Timestream for API InfluxDB und Befehlen interagiert. AWS CLI Der DB-Instance-Name muss für diesen Kunden in einer Region eindeutig sein. AWS

Der DB-Instance-Name ist Teil des DNS Hostnamens, der Ihrer Instance von Timestream for InfluxDB zugewiesen wurde. Wenn Sie beispielsweise influxdb1 als DB-Instance-Namen angeben, weist Timestream Ihrer Instance automatisch einen Endpunkt zu. DNS Ein Beispiel für einen Endpunkt ist`influxdb1-3ksj4dla5nfjhi.us-east-1.timestream-influxdb.amazonaws.com`, wo `influxdb1` Ihr Instance-Name ist.

Im Beispielendpunkt `3ksj4dla5nfjhi` ist `influxdb1-3ksj4dla5nfjhi.us-east-1.timestream-influxdb.amazonaws.com` die Zeichenfolge eine eindeutige Konto-ID, die von generiert wurde AWS. Die Kennung `3ksj4dla5nfjhi` im Beispiel ändert sich für das angegebene Konto in einer bestimmten Region nicht. Daher haben alle Ihre DB-Instances, die mit diesem Konto erstellt wurden, dieselbe feste ID. Beachten Sie die folgenden Merkmale der festen Kennung:

- Derzeit unterstützt Timestream for InfluxDB das Umbenennen von DB-Instances nicht.
- Wenn Sie eine DB-Instance mit derselben DB-Instance-Kennung löschen und neu erstellen, ist der Endpunkt identisch.
- Wenn Sie dasselbe Konto verwenden, um eine DB-Instance in einer anderen Region zu erstellen, unterscheidet sich die intern generierte Kennung, da die Region unterschiedlich ist, wie in `influxdb2.4a3j5du5ks7md2.us-west-1.timestream-influxdb.amazonaws.com`.

Jede DB-Instance unterstützt nur einen Timestream für die InfluxDB-Datenbank-Engine.

Beim Erstellen einer DB-Instance erfordert InfluxDB die Angabe eines Organisationsnamens. Eine DB-Instance kann mehrere Organisationen und mehrere Buckets hosten, die jeder Organisation zugeordnet sind.

Amazon Timestream for InfluxDB ermöglicht es Ihnen, im Rahmen des Erstellungsprozesses ein Master-Benutzerkonto und ein Passwort für Ihre DB-Instance zu erstellen. Dieser Master-Benutzer ist berechtigt, Organisationen und Buckets zu erstellen und Lese-, Schreib-, Lösch- und Upsert-Operationen mit Ihren Daten durchzuführen. Sie können auch auf die InfluxUI zugreifen und Ihr Operator-Token bei Ihrer ersten Anmeldung abrufen. Von dort aus können Sie auch alle Ihre Zugriffstoken verwalten. Sie müssen das Master-Benutzerpasswort festlegen, wenn Sie eine DB-

Instance erstellen. Sie können es jedoch jederzeit mithilfe von InfluxAPI, Influx oder CLI InfluxUI ändern.

DB-Instance-Klassen

Die DB-Instance-Klasse bestimmt die Berechnungs- und Speicherkapazität einer Amazon Timestream `fi UbfkyxDB` Timestream-DB-Instance. Die benötigte DB-Instance-Klasse richtet sich nach Ihren Rechen- und Speicheranforderungen.

Eine DB-Instance-Klasse besteht sowohl aus dem DB-Instance-Klassentyp als auch aus der -größe. `db.influx`Ist beispielsweise ein speicheroptimierter DB-Instance-Klassentyp, der für die hohen Speicheranforderungen im Zusammenhang mit laufenden Workloads geeignet ist. InfluxDb Innerhalb des `db.influx` Instance-Klassentyps `db.influx.2xlarge` befindet sich eine DB-Instance-Klasse. Die Größe dieser Klasse ist 2xlarge.

Weitere Informationen zu den Preisen für Instance-Klassen finden Sie unter Amazon Timestream für InfluxDB-Preise.

DB-Instance-Klassenarten

Amazon Timestream for InfluxDB unterstützt DB-Instance-Klassen für den folgenden Anwendungsfall, der für InfluxDB-Anwendungsfälle optimiert ist.

- **`db.influx`**— Diese Instance-Klassen eignen sich ideal für die Ausführung speicherintensiver Workloads in Open-Source-InfluxDB-Datenbanken

Hardwarespezifikationen für DB-Instance-Klassen

Die folgende Terminologie beschreibt die Hardwarespezifikationen für DB-Instance-Klassen:

- v CPU

  Die Anzahl der virtuellen Zentraleinheiten (CPUs). Eine virtuelle Einheit CPU ist eine Kapazitätseinheit, mit der Sie DB-Instance-Klassen vergleichen können.

- Arbeitsspeicher (GiB)

DieRAM, in Gibibyte, die der DB-Instance zugewiesen ist. Oft besteht ein konsistentes Verhältnis zwischen Speicher und V. CPU Nehmen wir als Beispiel die Instanzklasse db.influx, deren CPU Verhältnis von Speicher zu V dem der Instanzklasse EC2 r7g ähnelt.

- Für den Zufluss optimiert

  Eine DB-Instance nutzt einen optimierten Konfigurations-Stack und bietet zusätzliche dedizierte Kapazität für I/O-Vorgänge. Diese Optimierung bietet die beste Leistung, indem Konflikte zwischen I/O-Vorgängen und anderem Datenverkehr von Ihrer Instance minimiert werden.

- Netzwerkbandbreite

  Die Netzwerkgeschwindigkeit relativ zu anderen DB-Instance-Klassen. In der folgenden Tabelle finden Sie Hardwaredetails zu den Amazon Timestream for InfluxDB-Instance-Klassen.

Instanzen-Klasse	v CPU	Arbeitsspeicher (GiB)	Speichertyp	Netzwerkbandbreite (Gbit/s)
db.influx.medium	1	8	Zustrom eingeschlossen IOPS	10
db.influx.large	2	16	Zustrom eingeschlossen IOPS	10
db.influx.xlarge	4	32	Zustrom enthalten IOPS	10
db.influx.2xlarge	8	64	Zustrom enthalten IOPS	10
db.influx.4xlarge	16	128	Zustrom enthalten IOPS	10
db.influx.8xlarge	32	256	Zustrom enthalten IOPS	12

Instanzen-Klasse	v CPU	Arbeitsspeicher (GiB)	Speichertyp	Netzwerkbandbreite (Gbit/s)
db.influx.12xlarge	48	384	Zustrom enthalten IOPS	20
db.influx.16xlarge	64	512	Zustrom enthalten IOPS	25

InfluxDB-Instanzspeicher

DB-Instances für Amazon Timestream for InfluxDB verwenden Influx IOPS Included Volumes für Datenbanken und Protokollspeicher.

In einigen Fällen kann Ihre Datenbank-Arbeitslast möglicherweise nicht 100 Prozent der IOPS von Ihnen bereitgestellten Leistung erreichen. Weitere Informationen finden Sie unter Faktoren, die die Speicherleistung beeinflussen. Weitere Informationen zu den Preisen für Timestream for InfluxDB-Speicher finden Sie unter Amazon Timestream Timestream-Preise.

Amazon Timestream für InfluxDB-Speichertypen

Amazon Timestream for InfluxDB unterstützt einen Speichertyp, Influx Included. IOPS Sie können Timestream für InfluxDB-Instances mit bis zu 16 Tebibyte (TiB) Speicher erstellen.

Hier ist eine kurze Beschreibung des verfügbaren Speichertyps:

- Influx IO Inbegriffener Speicher: Die Speicherleistung ist die Kombination aus I/O-Vorgängen pro Sekunde (IOPS) und der Geschwindigkeit, mit der das Speichervolume Lese- und Schreibvorgänge ausführen kann (Speicherdurchsatz). Amazon Timestream for IOPS InfluxDB bietet auf Influx-Include-Speichervolumes drei Speicherstufen, die vorkonfiguriert sind IOPS und den optimalen Durchsatz bieten, der für verschiedene Arten von Workloads erforderlich ist.

Dimensionierung der InfluxDB-Instances

Die optimale Konfiguration einer Timestream for InfluxDB-Instance hängt von vielen Faktoren ab, darunter Aufnahmerate, Batchgrößen, Zeitreihenkardinalität, gleichzeitige Abfragen und Abfragetypen. In dem Bemühen, Größenempfehlungen zu geben, konzentrieren wir uns auf einen beispielhaften Workload mit den folgenden Merkmalen:

- Daten werden von einer Flotte von Telegraf-Agenten gesammelt und geschrieben, die SystemCPU, Speicher, Festplatte, I/O usw. von einem Rechenzentrum abrufen.

  Jede Schreibanforderung enthält 5000 Zeilen.
- Die Art der Abfragen, die auf dem System ausgeführt werden, werden als Abfragen mit „mäßiger Komplexität“ eingestuft. Diese Kategorie von Abfragen weist die folgenden Merkmale auf:
  - Sie haben mehrere Funktionen und einen oder zwei reguläre Ausdrücke
  - Sie können auch nach Klauseln gruppieren oder einen Zeitraum von mehreren Wochen als Stichprobe verwenden.
  - Die Ausführung dauert in der Regel einige hundert Millisekunden bis zu einigen tausend Millisekunden.
  - CPUbegünstigt hauptsächlich die Abfrageleistung.

Instance-Klasse	Speichertyp	Schreibvorgänge (Zeilen pro Sekunde)	Lesevorgänge (Abfragen pro Sekunde)
db.influx.large	Influx IO inklusive 3.000	~50.000	<10
db.influx.2xgroß	Influx IO inklusive 3.000	~150.000	<25
db.influx.4x groß	Influx IO inklusive 3.000	~200.000	~25
db.influx.4xlarge	Influx IO 12K enthalten	~250.000	~35
db.influx.8xlarge	Influx IO 12K enthalten	~500.000	~50
db.influx.12xlarge	Influx IO 12K enthalten	<750.000	<100

AWS Regionen und Verfügbarkeitszonen

Amazon Cloud Computing-Ressourcen werden an mehreren Standorten weltweit gehostet. Diese Standorte bestehen aus AWS Regionen und Verfügbarkeitszonen. Jede AWS Region ist ein separates geografisches Gebiet. Jede AWS Region hat mehrere isolierte Standorte, die als Verfügbarkeitszonen bezeichnet werden.

> **Note**
>
> Informationen zur Suche nach den Verfügbarkeitszonen für eine AWS Region finden Sie unter Regionen und Zonen im EC2Amazon-Benutzerhandbuch.

Amazon Timestream for InfluxDB ermöglicht es Ihnen, Ressourcen wie DB-Instances und Daten an mehreren Standorten zu platzieren.

Amazon betreibt state-of-the-art hochverfügbare Rechenzentren. Obwohl selten, können Fehler auftreten, die sich auf die Verfügbarkeit von DB-Instances auswirken, die sich am selben Speicherort befinden. Wenn Sie alle Ihre DB-Instances an einem Ort hosten, der von einem solchen Ausfall betroffen ist, ist keine Ihrer DB-Instances verfügbar.

Es ist wichtig, sich daran zu erinnern, dass jede AWS Region völlig unabhängig ist. Jede Amazon Timestream for InfluxDB-Aktivität, die Sie initiieren (z. B. das Erstellen von Datenbank-Instances oder das Auflisten verfügbarer Datenbank-Instances), wird nur in Ihrer aktuellen Standardregion ausgeführt. AWS Sie können die standardmäßige AWS -Region in der Konsole ändern, indem Sie die Umgebungsvariable `AWS_DEFAULT_REGION` festlegen. Oder sie kann überschrieben werden, indem der `--region` Parameter mit der () verwendet wird. AWS Command Line Interface AWS CLI Weitere Informationen finden Sie unter Konfiguration von AWS Command Line Interface, insbesondere in den Abschnitten zu Umgebungsvariablen und Befehlszeilenoptionen.

Um eine Amazon Timestream for InfluxDB-DB-Instance in einer bestimmten AWS Region zu erstellen oder mit ihr zu arbeiten, verwenden Sie den entsprechenden regionalen Service-Endpunkt.

AWS Verfügbarkeit in der Region

Weitere Informationen zu AWS Regionen, in denen Amazon Timestream for InfluxDB derzeit verfügbar ist, und zu den Endpunkten für jede Region finden Sie unter Amazon Timestream Timestream-Endpunkte und Kontingente.

AWS Gestaltung der Regionen

Jede AWS Region ist so konzipiert, dass sie von den anderen AWS Regionen isoliert ist. Dieser Entwurf sorgt für die größtmögliche Fehlertoleranz und Stabilität.

Wenn Sie Ihre Ressourcen anzeigen, sehen Sie nur die Ressourcen, die mit der von Ihnen angegebenen AWS Region verknüpft sind. Das liegt daran, dass AWS Regionen voneinander isoliert sind und wir Ressourcen nicht automatisch regionsübergreifend AWS replizieren.

AWS Verfügbarkeitszonen

Wenn Sie eine DB-Instance erstellen, wählt Amazon Timestream for InfluxDB basierend auf Ihrer Subnetzkonfiguration nach dem Zufallsprinzip eine für Sie aus. Eine Availability Zone wird durch einen AWS Regionalcode gefolgt von einer Buchstabenkennung (z. B.) dargestellt. `us-east-1a`

Verwenden Sie den EC2 Amazon-Befehl `describe-availability` -zones wie folgt, um die Availability Zones innerhalb der angegebenen Region zu beschreiben, die für Ihr Konto aktiviert sind.

```
aws ec2 describe-availability-zones --region region-name
```

Um beispielsweise die Availability Zones in der Region USA Ost (Nord-Virginia) (us-east-1) zu beschreiben, die für Ihr Konto aktiviert sind, führen Sie den folgenden Befehl aus:

```
aws ec2 describe-availability-zones --region us-east-1
```

Sie können die Availability Zones für die primären und sekundären DB-Instances in einer Multi-AZ-DB-Bereitstellung nicht auswählen. Amazon Timestream for InfluxDB wählt sie nach dem Zufallsprinzip für Sie aus. Weitere Informationen zu Multi-AZ-Bereitstellungen finden Sie unter.. Konfiguration und Verwaltung einer Multi-AZ-Bereitstellung

DB-Instance-Abrechnung für Amazon Timestream für InfluxDB

Amazon Timestream für InfluxDB-Instances werden auf der Grundlage der folgenden Komponenten abgerechnet:

- DB-Instance-Stunden (pro Stunde) — Basierend auf der DB-Instance-Klasse der DB-Instance, zum Beispiel db.influx.large. Die Preise werden auf Stundenbasis aufgeführt, aber Rechnungen werden jetzt auf die Sekunde genau kalkuliert und zeigen die Zeiten im Dezimalformat an. Die Nutzung von

Amazon Timestream für InfluxDB wird in 1-Sekunden-Schritten mit einem Minimum von 10 Minuten abgerechnet. Weitere Informationen finden Sie unter DB-Instance-Klassen. DB-Instance-Klassen

- Speicher (pro GiB pro Monat) — Speicherkapazität, die Sie für Ihre DB-Instance bereitgestellt haben. Weitere Informationen finden Sie unter InfluxDB-Instanzspeicher.
- Datenübertragung (pro GB) — Datenübertragung in und aus Ihrer DB-Instance vom oder zum Internet und anderen AWS Regionen.

Preisinformationen zu Amazon Timestream for InfluxDB finden Sie auf der Preisseite für Amazon Timestream for InfluxDB.

Amazon Timestream für InfluxDB einrichten

Bevor Sie Amazon Timestream for InfluxDB zum ersten Mal verwenden, führen Sie die folgenden Aufgaben aus:

Wenn Sie bereits ein AWS Konto haben, sollten Sie Ihre Amazon Timestream for InfluxDB-Anforderungen kennen und es vorziehen, die Standardeinstellungen für IAM und VPC Erste Schritte mit Timestream for InfluxDB Erste Schritte mit Amazon Timestream for InfluxDB zu verwenden.

Eröffnen Sie ein Konto AWS

Wenn Sie noch kein AWS Konto haben, führen Sie die folgenden Schritte aus, um eines zu erstellen.

Um sich für ein AWS Konto zu registrieren

- Gehen Sie zur AWS Anmeldeseite.
- Wählen Sie Neues Konto erstellen und folgen Sie dann den Anweisungen.

> **Note**
>
> Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

Wenn Sie sich für ein AWS Konto registrieren, wird ein Root-Benutzer für das AWS Konto erstellt. Der Root-Benutzer hat Zugriff auf alle AWS Dienste und Ressourcen im Konto. Als bewährte Sicherheitsmethode weisen Sie einem Administratorbenutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

AWS sendet Ihnen nach Abschluss des Anmeldevorgangs eine Bestätigungs-E-Mail. Sie können Ihre aktuellen Kontoaktivitäten jederzeit einsehen und Ihr Konto verwalten, indem Sie zu https://aws.amazon.com/gehen und Mein Konto auswählen.

Benutzerverwaltung

Erstellen Sie einen Administratorbenutzer

Erstellen eines Administratorbenutzers

Nachdem Sie sich für ein AWS Konto angemeldet haben, erstellen Sie einen Administratorbenutzer, sodass Sie den Root-Benutzer nicht für alltägliche Aufgaben verwenden.

Schützen Sie Ihr AWS Konto als Root-Benutzer

Melden Sie sich als Kontoinhaber bei der AWS Management Console an, indem Sie Root-Benutzer auswählen und die E-Mail-Adresse Ihres AWS Kontos eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein. Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Als Root-Benutzer anmelden im AWS Anmelde-Benutzerhandbuch

Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für Ihren Root-Benutzer. Anweisungen finden Sie im Benutzerhandbuch unter Aktivieren eines virtuellen MFA Geräts für den Root-Benutzer (Konsole) Ihres AWS Kontos. IAM

Gewähren Sie programmatischen Zugriff

Benutzer benötigen programmatischen Zugriff, wenn sie mit AWS außerhalb des interagieren möchten. AWS Management Console Die Vorgehensweise, um programmgesteuerten Zugriff zu gewähren, hängt davon ab, welcher Benutzertyp auf zugreift AWS.

Um Benutzern programmgesteuerten Zugriff zu gewähren, wählen Sie eine der folgenden Optionen:

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
Identität der Belegschaft (im IAM Identity Center verwaltete Benutzer)	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an das AWS CLI AWS SDKs, oder AWS APIs zu signieren.	Folgen Sie den Anweisungen für die Schnittstelle, die Sie verwenden möchten.* Informationen zum AWS CLI

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
		Konfiguration von für die Verwendung AWS IAM von AWS CLI Identity Center in der AWS-Benutzerhandbuch für die Befehlszeilenschnittstelle .* Informationen zu AWS SDKs Tools und finden Sie AWS APIs unter IAMIdentity Center-Authentifizierung in der AWSSDKsund Referenzhandbuch für Tools.
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an AWS CLISDKs, und APIs zu signieren.	Folgen Sie den Anweisungen unter Verwenden temporärer Anmeldeinformationen mit AWS Ressourcen im IAMBenutzerhandbuch.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um programmatische Anfragen für AWS CLISDKs, und APIs zu signieren.	Folgen Sie den Anweisungen für die Schnittstelle, die Sie verwenden möchten. Informationen zu den finden Sie unter AWS CLI Authentifizierung mit Benutzeranmeldedaten IAM in der AWS Benutzerhandbuch für die Befehlszeilenschnittstelle .Informationen AWS SDKs und Tools finden Sie unter Authentifizieren Sie sich mit langfristigen Anmeldeinformationen in der AWS SDKsund Referenzhandbuch für Tools . Weitere Informationen finden Sie AWS APIs unter Zugriffsschlüssel für IAM Benutzer verwalten in der IAM-Benutzerhandbuch.

Ermitteln der Anforderungen

Der grundlegende Baustein von Amazon Timestream for Influx ist die DB-Instance. In einer DB-Instance erstellen Sie Ihre Buckets. Eine DB-Instance gibt eine Netzwerkadresse, den sogenannten

Endpunkt, an. Ihre Anwendungen verwenden diesen Endpunkt, um eine Verbindung mit Ihrer DB-Instance einzurichten. Sie werden auch über denselben Endpunkt von Ihrem Browser aus auf Ihr InfluxUI zugreifen. Wenn Sie eine DB-Instance erstellen, geben Sie Details wie Speicher, Arbeitsspeicher, Datenbank-Engine und Version, Netzwerkkonfiguration und Sicherheit an. Der Netzwerkzugriff auf eine DB-Instance wird über eine Sicherheitsgruppe kontrolliert.

Bevor Sie eine DB-Instance und eine Sicherheitsgruppe erstellen, müssen Sie Ihre DB-Instance und die Netzwerkanforderungen kennen. Hier einige wichtige Dinge, die Sie berücksichtigen sollten:

- Ressourcenanforderungen — Was sind die Speicher- und Prozessoranforderungen für Ihre Anwendung oder Ihren Dienst? Sie verwenden diese Einstellungen, um zu bestimmen, welche DB-Instance-Klasse Sie verwenden sollten. Spezifikationen zu DB-Instance-Klassen finden Sie unter DB-Instance-Klassen.
- VPCund Sicherheitsgruppe — Ihre DB-Instance wird sich höchstwahrscheinlich in einer virtuellen privaten Cloud befinden (VPC). Um eine Verbindung zu Ihrer DB-Instance einzurichten, müssen Sie Sicherheitsgruppenregeln festlegen. Diese Regeln sind unterschiedlich eingerichtet, je nachdem, welche Art von Regeln VPC Sie verwenden und wie Sie sie verwenden. Sie können beispielsweise Folgendes verwenden: eine Standardeinstellung VPC oder eine benutzerdefinierteVPC.

  In der folgenden Liste werden die Regeln für jede VPC Option beschrieben:

  - Standard VPC — Wenn Ihr AWS Konto VPC in der aktuellen AWS Region einen Standard hat, der für die Unterstützung von DB-Instances konfiguriert VPC ist. Wenn Sie VPC bei der Erstellung der DB-Instance den Standard angeben, stellen Sie sicher, dass Sie eine VPCSicherheitsgruppe erstellen, die Verbindungen von der Anwendung oder dem Service zur Amazon Timestream for InfluxDB-DB-Instance autorisiert. Verwenden Sie die Option Sicherheitsgruppe auf der VPC Konsole oder der, um Sicherheitsgruppen AWS CLI zu erstellen. VPC Weitere Informationen finden Sie unter Schritt 3: VPC Sicherheitsgruppe erstellen.
- Benutzerdefiniert VPC — Wenn Sie VPC beim Erstellen einer DB-Instance eine benutzerdefinierte angeben möchten, beachten Sie Folgendes:

  - Stellen Sie sicher, dass Sie eine VPCSicherheitsgruppe erstellen, die Verbindungen von der Anwendung oder dem Service zur Amazon Timestream for InfluxDB-DB-Instance autorisiert. Verwenden Sie die Option Sicherheitsgruppe auf der VPC Konsole oder um Sicherheitsgruppen AWS CLI zu erstellen. VPC Weitere Informationen finden Sie unter Schritt 3: VPC Sicherheitsgruppe erstellen.

  - Sie VPC müssen bestimmte Anforderungen erfüllen, um DB-Instances hosten zu können, z. B. mindestens zwei Subnetze, die sich jeweils in einer separaten Availability Zone befinden. Informationen finden Sie unter Amazon VPC VPCs und Amazon Timestream für InfluxDB.
- Hohe Verfügbarkeit — Benötigen Sie Failover-Unterstützung? Auf Amazon Timestream for InfluxDB erstellt eine Multi-AZ-Bereitstellung eine primäre DB-Instance und eine sekundäre Standby-DB-Instance in einer anderen Availability Zone für die Failover-Unterstützung. Wir empfehlen Multi-AZ-Bereitstellungen, um die hohe Verfügbarkeit von Produktions-Workloads sicherzustellen. Für Entwicklungs- und Testzwecke reicht gewöhnlich eine Bereitstellung ohne Multi-AZ. Weitere Informationen finden Sie unter Multi-AZ-DB-Instance-Bereitstellungen.
- IAMRichtlinien — Verfügt Ihr AWS Konto über Richtlinien, die die für die Durchführung von Amazon Timestream for InfluxDB-Vorgängen erforderlichen Berechtigungen gewähren? Wenn Sie eine Verbindung AWS mit IAM Anmeldeinformationen herstellen, muss Ihr IAM Konto über IAM Richtlinien verfügen, die die für die Durchführung von Amazon Timestream for InfluxDB-Steuerungsebenenoperationen erforderlichen Berechtigungen gewähren. Weitere Informationen finden Sie unter Identity and Access Management für Amazon Timestream für InfluxDB.
- Offene Ports — Auf welchem TCP /IP-Port wartet Ihre Datenbank? Die Firewalls einiger Unternehmen blockieren möglicherweise Verbindungen zum Standard-Port für Ihre Datenbank-Engine. Der Standard für Timestream for InfluxDB ist 8086.
- AWS Region — In welcher AWS Region möchten Sie Ihre Datenbank haben? Indem sich Ihre Datenbank nahe bei der Anwendung oder dem Webdienst befindet, könnten Netzwerklatenzen verringert werden. Weitere Informationen finden Sie unter AWS Regionen und Verfügbarkeitszonen.
- DB-Festplattensubsystem — Was sind Ihre Speicheranforderungen? Amazon Timestream for InfluxDB bietet drei Konfigurationen für den Speichertyp Influx IOPS Included:
  - Influx Io Inbegriffen: 3.000 () IOPS SSD
  - Influx Io inklusive 12 k () IOPS SSD
  - Influx Io inklusive 25k () IOPS SSD

  Weitere Informationen zu Amazon Timestream für InfluxDB-Speicher finden Sie unter Amazon Timestream für InfluxDB-DB-Instance-Speicher. Sobald Ihnen die benötigten Informationen zur Erstellung der Sicherheitsgruppe und der DB-Instance vorliegen, fahren Sie mit dem nächsten Schritt fort.

Ermöglichen Sie Zugriff auf Ihre DB-Instance in Ihrem, indem Sie eine Sicherheitsgruppe erstellen VPC

VPCSicherheitsgruppen ermöglichen den Zugriff auf DB-Instances in einerVPC. Sie fungieren als Firewall für die zugeordneten DB-Instances und steuern den ein- und ausgehenden Datenverkehr auf der DB-Instance-Ebene. DB-Instances werden standardmäßig mit einer Firewall und einer Standard-Sicherheitsgruppe erstellt, die DB-Instance schützen.

Bevor Sie eine Verbindung zu Ihrer DB-Instance einrichten können, müssen Sie einer Sicherheitsgruppe Regeln hinzufügen. Verwenden Sie Ihre Netzwerk- und Konfigurationsinformationen, um Regeln für den Zugriff auf Ihre DB-Instance festzulegen.

Nehmen wir beispielsweise an, Sie haben eine Anwendung, die auf eine Datenbank in Ihrer DB-Instance in einem VPC zugreift. In diesem Fall müssen Sie eine benutzerdefinierte TCP Regel hinzufügen, die den Portbereich und die IP-Adressen angibt, die Ihre Anwendung für den Zugriff auf die Datenbank verwendet. Wenn Sie eine Anwendung auf einer EC2 Amazon-Instance haben, können Sie die Sicherheitsgruppe verwenden, die Sie für die EC2 Amazon-Instance eingerichtet haben.

Eine Sicherheitsgruppe für den VPC Zugriff erstellen

Um eine VPC Sicherheitsgruppe zu erstellen, melden Sie sich bei der an AWS Management Console und wählen Sie VPC.

> **Note**
>
> Stellen Sie sicher, dass Sie sich in der VPC Konsole befinden, nicht in der Amazon Timesteam for InfluxDB-Konsole.

- Wählen Sie in der oberen rechten Ecke von die AWSRegion aus AWS Management Console, in der Sie Ihre VPC Sicherheitsgruppe und DB-Instance erstellen möchten. In der Liste der VPC Amazon-Ressourcen für diese AWS Region sollten Sie mindestens ein VPC und mehrere Subnetze sehen. Wenn Sie dies nicht tun, haben Sie VPC in dieser AWS Region keinen Standard. .
- Wählen Sie im Navigationsbereich Sicherheitsgruppen aus.
- Wählen Sie Create security group (Sicherheitsgruppe erstellen) aus.

- Geben Sie auf der Seite Sicherheitsgruppe im Abschnitt Grundlegende Details den Namen und die Beschreibung der Sicherheitsgruppe ein. Wählen Sie für die aus VPCVPC, in der Sie Ihre DB-Instance erstellen möchten.
- Wählen Sie in Eingehende Regeln die Option Regel hinzufügen.
  - Wählen Sie für Typ die Option Benutzerdefiniert TCP aus.
  - Wählen Sie als Quelle einen Namen für die Sicherheitsgruppe oder geben Sie den IP-Adressbereich (CIDRWert) ein, von dem aus Sie auf die DB-Instance zugreifen. Wenn Sie My IP (Meine IP) auswählen, ermöglicht dies den Zugriff auf die DB-Instance von der in Ihrem Browser erkannten IP-Adresse.

  Wählen Sie als Quelle einen Sicherheitsgruppennamen oder geben Sie den IP-Adressbereich (CIDRWert) ein, von dem aus Sie auf die DB-Instance zugreifen. Wenn Sie My IP (Meine IP) auswählen, ermöglicht dies den Zugriff auf die DB-Instance von der in Ihrem Browser erkannten IP-Adresse.
- (Optional) Fügen Sie inRegeln für ausgehenden Datenverkehr Regeln für ausgehenden Datenverkehr hinzu. Standardmäßig ist der gesamte ausgehende Datenverkehr zugelassen.
- Wählen Sie Sicherheitsgruppe erstellen aus.

Sie können diese VPCSicherheitsgruppe als Sicherheitsgruppe für Ihre DB-Instance verwenden, wenn Sie sie erstellen.

> **Note**
>
> Wenn Sie eine Standardsubnetzgruppe verwendenVPC, wird für Sie eine Standard-Subnetzgruppe erstellt, die VPC sich über alle Subnetze erstreckt. Wenn Sie eine DB-Instance erstellen, können Sie die Standardeinstellung eiifccntf VPC und die Standardeinstellung für DB-Subnetzgruppe wählen.

Nachdem Sie die erforderliche Einrichtung abgeschlossen haben, können Sie eine DB-Instance unter Verwendung Ihrer Anforderungen und Sicherheitsgruppe erstellen. Befolgen Sie hierzu die Anweisungen unter Erstellen einer DB-Instance.

Erste Schritte mit Timestream for InfluxDB

In den folgenden Beispielen erfahren Sie, wie Sie mithilfe von Amazon Timestream for InfluxDB Service eine DB-Instance erstellen und eine Verbindung zu ihr herstellen.

> **Note**
>
> Sie müssen die Aufgaben im Abschnitt Amazon Timestream für InfluxDB einrichten abschließen, um eine DB-Instance erstellen oder eine Verbindung mit einer DB-Instance herstellen zu können.

Themen

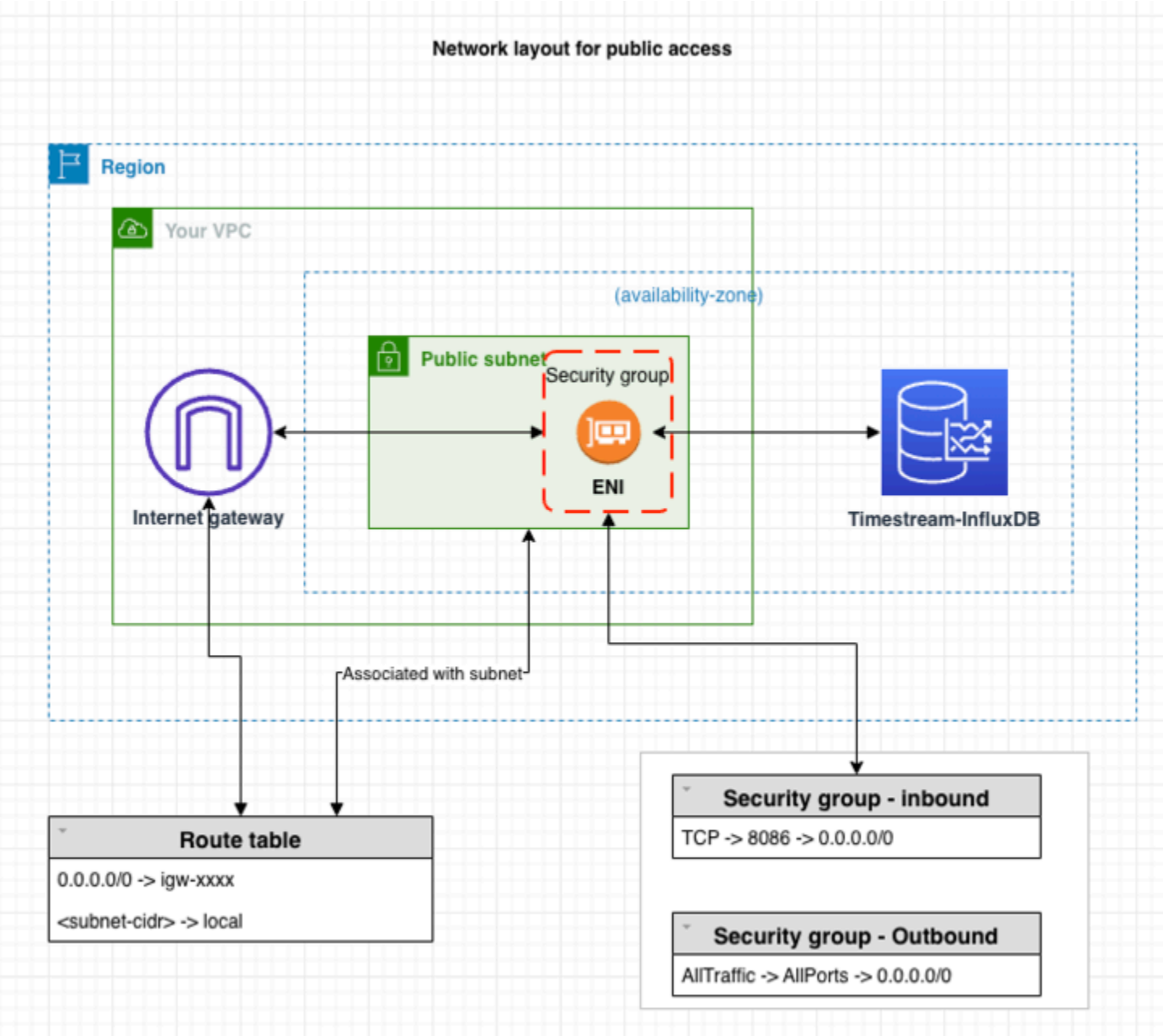
Eine Timestream for InfluxDB-Instance erstellen und eine Verbindung zu ihr herstellen

In diesem Tutorial werden eine EC2 Amazon-Instance und eine Amazon Timestream für InfluxDB-DB-Instance erstellt. Das Tutorial zeigt Ihnen, wie Sie mit dem Telegraf-Client Daten von der Instance in die EC2 DB-Instance schreiben. Als bewährte Methode erstellt dieses Tutorial eine private DB-Instance in einer virtuellen privaten Cloud ()VPC. In den meisten Fällen können andere Ressourcen in derselben InstanzVPC, wie z. B. EC2 Instances, auf die DB-Instance zugreifen, aber Ressourcen außerhalb der VPC können nicht darauf zugreifen.

Nachdem Sie das Tutorial abgeschlossen haben, gibt es in jeder Availability Zone in Ihrer VPC ein öffentliches und ein privates Subnetz. In einer Availability Zone befindet sich die EC2 Instance im öffentlichen Subnetz und die DB-Instance im privaten Subnetz.

> **Note**
>
> Für die Erstellung eines Kontos fallen keine Gebühren an AWS . Wenn Sie dieses Tutorial abschließen, können Ihnen jedoch Kosten für die von Ihnen verwendeten AWS Ressourcen entstehen. Sie können diese Ressourcen nach Abschluss des Tutorials löschen, wenn sie nicht mehr benötigt werden.

Das folgende Diagramm zeigt die Konfiguration, wenn die Barrierefreiheit öffentlich ist.



Warning

Wir empfehlen nicht, 0.0.0.0/0 für den HTTP Zugriff zu verwenden, da Sie allen IP-Adressen den Zugriff auf Ihre öffentliche InfluxDB-Instance ermöglichen. HTTP Dieser Ansatz ist in einer Testumgebung nicht einmal für kurze Zeit akzeptabel. Autorisieren Sie nur eine bestimmte IP-Adresse oder einen bestimmten Adressbereich für den Zugriff auf Ihre InfluxDB-Instances, indem Sie HTTP Being for WebUI oder Access verwenden. API

Dieses Tutorial erstellt eine DB-Instance, auf der InfluxDB ausgeführt wird, mit dem. AWS Management Console Wir werden uns nur auf die Größe der DB-Instance und die DB-Instance-ID konzentrieren. Wir werden die Standardeinstellungen für die anderen Konfigurationsoptionen verwenden. Die in diesem Beispiel erstellte DB-Instance wird privat sein.

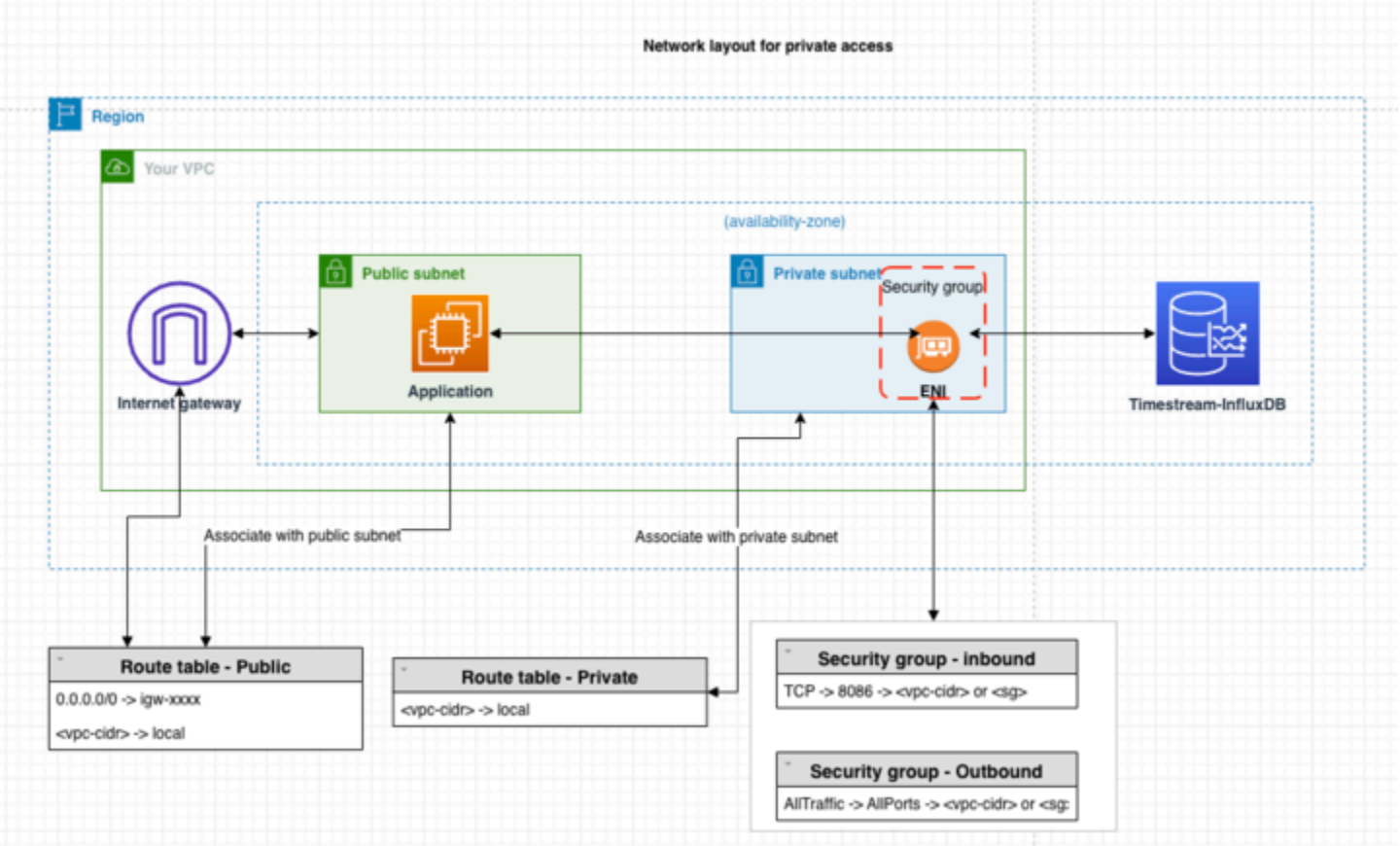
Zu den weiteren Einstellungen, die Sie konfigurieren könnten, gehören Verfügbarkeit, Sicherheit und Protokollierung. Um eine öffentliche DB-Instance zu erstellen, müssen Sie im Abschnitt Konnektivitätskonfiguration auswählen, ob Ihre Instance „Öffentlich zugänglich“ sein soll. Informationen zum Erstellen von DB-Instances finden Sie unterErstellen einer DB-Instance..

Wenn Ihre Instance nicht öffentlich zugänglich ist, gehen Sie wie folgt vor:

- Erstellen Sie auf VPC der Instance einen Host, über den Sie den Datenverkehr tunneln können.
- Richten Sie SSH-Tunneling zur Instanz ein. Weitere Informationen finden Sie unter EC2Amazon-Instance-Portweiterleitung mit AWS Systems Manager
- Damit das Zertifikat funktioniert, fügen Sie der `/etc/hosts` Datei Ihres Client-Computers die folgende Zeile hinzu:`127.0.0.1`. Dies ist die DNS Adresse Ihrer Instanz.
- Stellen Sie mithilfe des vollqualifizierten Domainnamens, z. B. https: //< DNS >:8086, eine Connect zu Ihrer Instance her.

> **Note**
>
> Localhost kann das Zertifikat nicht validieren, da localhost nicht Teil des Zertifikats ist. SAN

Das folgende Diagramm zeigt die Konfiguration, wenn der Zugriff privat ist:



Voraussetzungen

Bevor Sie die Schritte in diesem Abschnitt abschließen, stellen Sie sicher, dass Sie folgende Voraussetzungen erfüllen:

- Eröffnen Sie ein AWS Konto.
- Erstellen Sie einen Administratorbenutzer.

Schritt 1: EC2 Amazon-Instance erstellen

Erstellen Sie eine EC2 Amazon-Instance, mit der Sie eine Verbindung zu Ihrer Datenbank herstellen.

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die EC2 Amazon-Konsole unter https://console.aws.amazon.com/ec2/.
2. Wählen Sie in der oberen rechten Ecke von die AWS Region aus AWS Management Console, in der Sie die Instance erstellen möchten. EC2
3. Wählen Sie EC2Dashboard und dann Launch instance aus.

4. Wenn die Seite „Eine Instanz starten“ geöffnet wird, wählen Sie auf der Seite „Instanz starten“ die folgenden Einstellungen aus.

   a. Geben Sie unter Name und Tags für Name ec2-database-connect ein.

   b. Wählen Sie unter Anwendungs- und Betriebssystem-Images (Amazon Machine Image) Amazon Linux und dann Amazon Linux 2023 ausAMI. Übernehmen Sie für alle anderen Einstellungen die Standardwerte.

   c. Wählen Sie unter Instance type (Instance-Typ) den Wert t2.micro aus.

   d. Wählen Sie unter Key pair (login) (Schlüsselpaar (Anmeldung)) einen Key pair name (Schlüsselpaarname), um ein vorhandenes Schlüsselpaar zu verwenden. Um ein neues key pair für die EC2 Amazon-Instance zu erstellen, wählen Sie Neues key pair erstellen und verwenden Sie dann das Fenster key pair erstellen, um es zu erstellen. Weitere Informationen zum Erstellen eines neuen Schlüsselpaars finden Sie unter Create a key pair im EC2Amazon-Benutzerhandbuch für Linux-Instances.

   e. Wählen Sie unter SSH Traffic zulassen in den Netzwerkeinstellungen die Quelle der SSH Verbindungen zur EC2 Instance aus. Sie können Meine IP wählen, wenn die angezeigte IP-Adresse für SSH Verbindungen korrekt ist. Andernfalls können Sie mithilfe von Secure Shell (SSH) die IP-Adresse ermitteln, die für die Verbindung zu Ihren EC2 Instanzen VPC verwendet werden soll. Um Ihre öffentliche IP-Adresse in einem anderen Browserfenster oder einer anderen Registerkarte zu ermitteln, können Sie den Dienst unter verwenden https://checkip.amazonaws.com. Ein Beispiel für eine IP-Adresse ist 192.0.2.1/32. In vielen Fällen stellen Sie möglicherweise eine Verbindung über einen Internetdienstanbieter (ISP) oder hinter Ihrer Firewall ohne statische IP-Adresse her. Bestimmen Sie in diesem Fall den Bereich der IP-Adressen, die von Client-Computern verwendet werden.

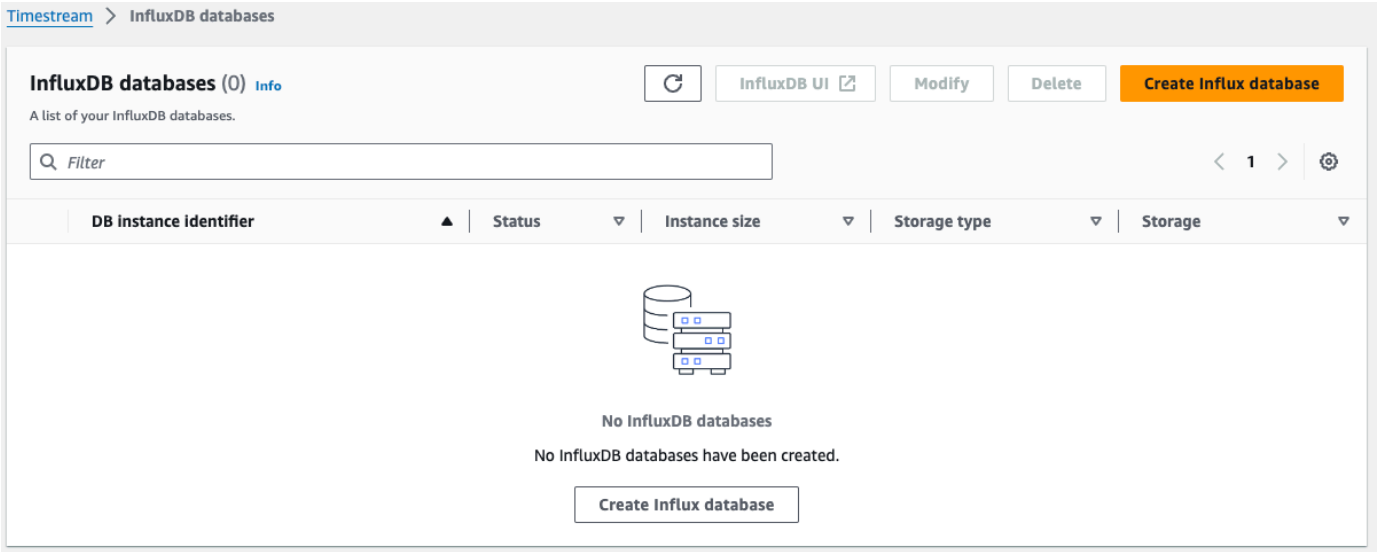
      > **Warning**
      >
      > Wir empfehlen nicht, 0.0.0.0/0 für den SSH Zugriff zu verwenden, da Sie damit allen IP-Adressen den Zugriff auf Ihre öffentlichen EC2 Instances ermöglichen. SSH Dieser Ansatz ist in einer Testumgebung nicht einmal für kurze Zeit akzeptabel. Autorisieren Sie nur eine bestimmte IP-Adresse oder einen Adressbereich für den Zugriff auf Ihre Instances. EC2 SSH

Schritt 2: Erstellen Sie eine InfluxDB-DB-Instance

Der grundlegende Baustein von Amazon Timestream for InfluxDB ist die DB-Instance. In dieser Umgebung betreiben Sie Ihre InfluxDB-Datenbanken.

In diesem Beispiel erstellen Sie eine DB-Instance, auf der die InfluxDB-Datenbank-Engine mit einer db.influx.large DB-Instance-Klasse ausgeführt wird.

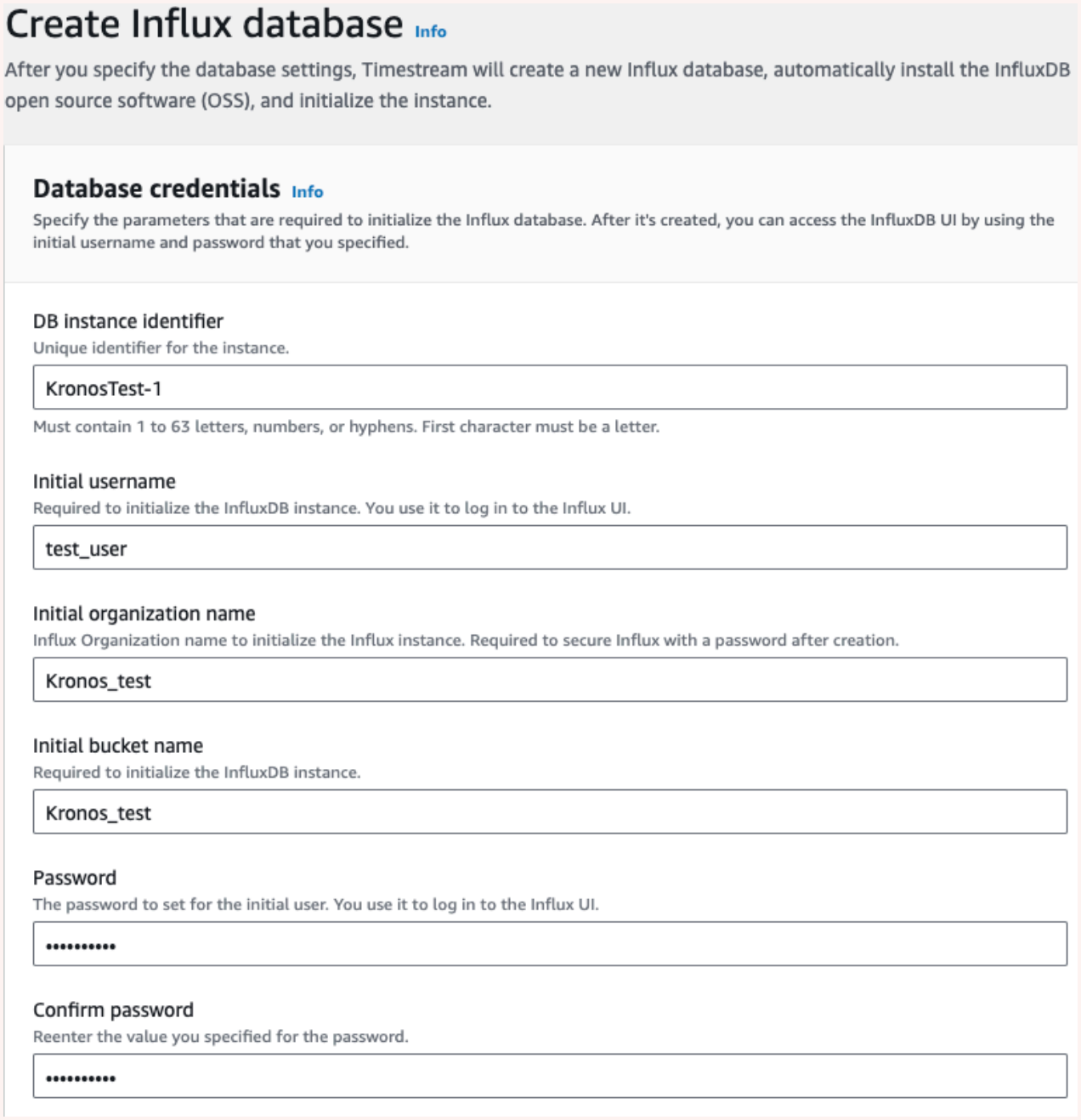
1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Timestream for InfluxDB-Konsole unter. https://console.aws.amazon.com/timestream/
2. Wählen Sie in der oberen rechten Ecke der Amazon Timestream for InfluxDB-Konsole die AWS Region aus, in der Sie die DB-Instance erstellen möchten.
3. Wählen Sie im Navigationsbereich InfluxDB-Datenbanken aus.
4. Wählen Sie Create Influx database.



5. Geben Sie für DB Instance Identifier KronosTest -1 ein.
6. Geben Sie die grundlegenden Konfigurationsparameter von InfluxDB an: Benutzername, Organisation, Bucket-Name und Passwort.

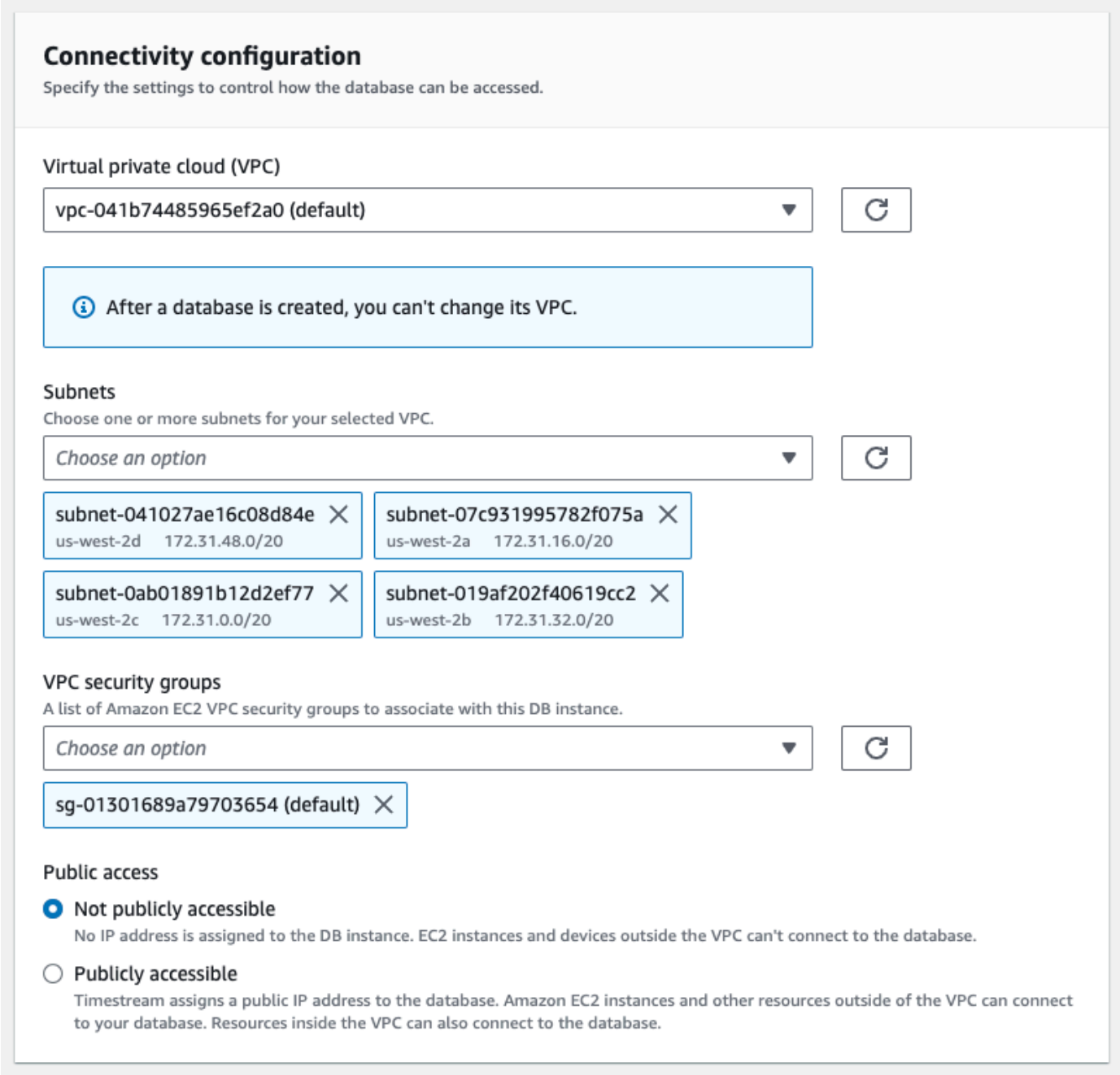
> **Important**
>
> Sie können das Benutzerpasswort nicht erneut anzeigen. Ohne Ihr Passwort können Sie nicht auf Ihre Instance zugreifen und kein Operator-Token erhalten. Wenn Sie es nicht notieren, müssen Sie es möglicherweise ändern. Siehe Erstellen Sie ein neues Operator-Token für Ihre InfluxDB-Instance.

Wenn Sie das Benutzerkennwort ändern müssen, nachdem die DB-Instance verfügbar ist, können Sie die DB-Instance entsprechend ändern. Weitere Informationen zum Ändern einer DB-Instance finden Sie unter Aktualisieren von DB-Instances.



7. Wählen Sie für DB Instance Class db.influx.large aus.
8. Wählen Sie für DB Storage Class die Option Influx Included 3K aus. IOPS
9. Konfigurieren Sie Ihre Protokolle. Weitere Informationen finden Sie unter Einrichtung zum Anzeigen von InfluxDB-Protokollen auf Timestream-InfluxDB-Instances.

10. Stellen Sie im Abschnitt Konnektivitätskonfiguration sicher, dass sich Ihre InfluxDB-Instance im selben Subnetz wie Ihre neu erstellte Instance befindet. EC2



11. Wählen Sie Create Influx database.
12. Wählen Sie in der Liste Datenbanken den Namen Ihrer neuen InfluxDB-Instanz aus, um deren Details anzuzeigen. Die DB-Instance hat den Status Creating, bis sie einsatzbereit ist.

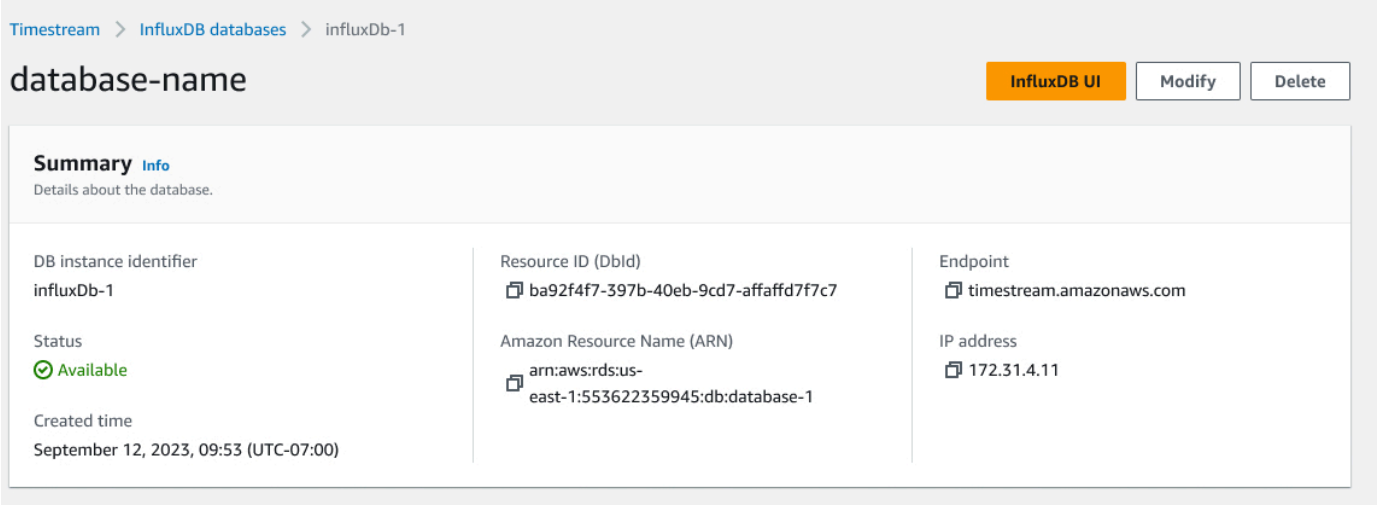
Sie können eine Verbindung zur DB-Instance herstellen, wenn sich der Status auf Verfügbar ändert. Abhängig von der Klasse der DB-Instance und vom verfügbaren Speicherplatz kann es bis zu 20 Minuten dauern, bis die neue DB-Instance verfügbar ist.

> **Important**
>
> Derzeit können Sie die Compute- (Instance-Typen) und die Speicherkonfiguration (Storage-Typen) vorhandener Instances nicht ändern.

Schritt 3: Senden Sie Telegraf-Daten an Ihre InfluxDB-Instanz

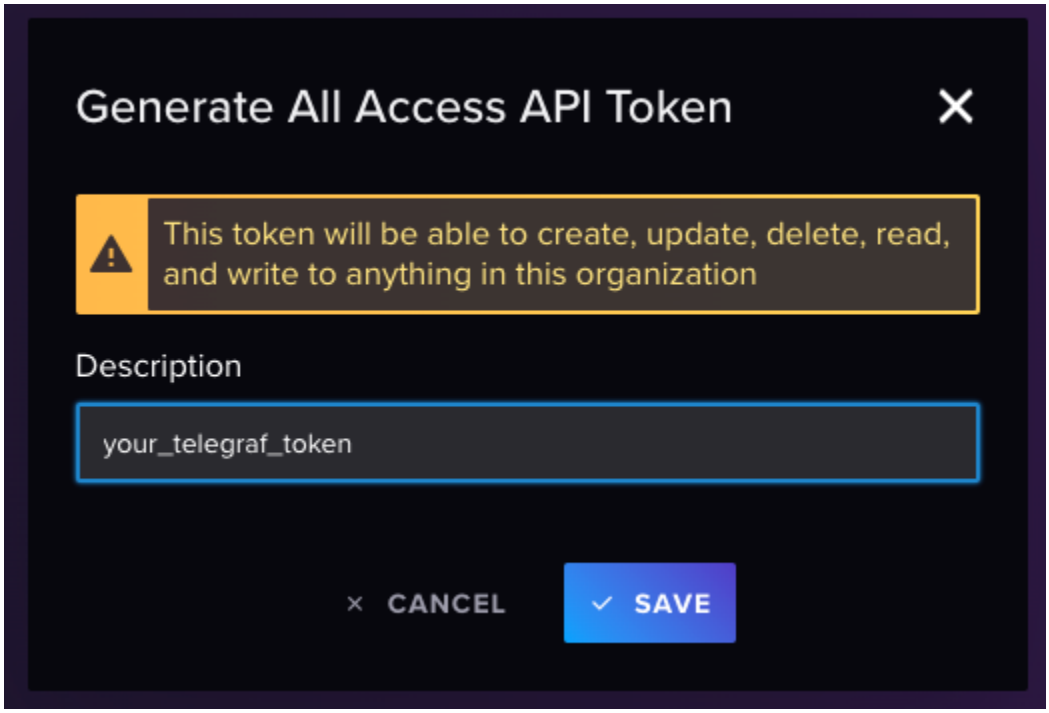
Sie können jetzt mit dem Telegraf-Agenten beginnen, Telemetriedaten an Ihre InfluxDB-DB-Instance zu senden. In diesem Beispiel installieren und konfigurieren Sie einen Telegraf-Agenten, um Leistungsmetriken an Ihre InfluxDB-DB-Instance zu senden.

1. Suchen Sie den Endpunkt (DNSName) und die Portnummer für Ihre DB-Instance.
   a. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Timestream Timestream-Konsole unter https://console.aws.amazon.com/timestream/.
   b. Wählen Sie in der oberen rechten Ecke der Amazon Timestream Timestream-Konsole die AWS Region für die DB-Instance aus.
   c. Wählen Sie im Navigationsbereich InfluxDB-Datenbanken aus.
   d. Wählen Sie den Namen der InfluxDB-DB-Instanz, um deren Details anzuzeigen.
   e. Kopieren Sie im Abschnitt Zusammenfassung den Endpunkt. Notieren Sie sich auch die Portnummer. Sie benötigen sowohl den Endpunkt als auch die Portnummer, um eine Verbindung zur DB-Instance herzustellen (die Standard-Portnummer für InfluxDB ist 8086).
2. Wählen Sie als Nächstes InfluxDB UI aus.



3. Dadurch wird ein neues Browserfenster geöffnet, in dem Sie eine Anmeldeaufforderung sehen sollten. Geben Sie die Anmeldeinformationen ein, die Sie zuvor verwendet haben, um Ihre InfluxDB-Db-Instance zu erstellen.
4. Klicken Sie im Navigationsbereich auf den Pfeil und wählen API Sie Tokens aus.
5. Generieren Sie für diesen Test ein All-Access-Token.

> **Note**
>
> Für Produktionsszenarien empfehlen wir, Token mit spezifischem Zugriff auf die erforderlichen Buckets zu erstellen, die für bestimmte Telegraf-Anforderungen erstellt wurden.



6. Ihr Token wird auf dem Bildschirm angezeigt.

> **Important**
>
> Achten Sie darauf, das Token zu kopieren und zu speichern, da Sie es nicht erneut anzeigen können.

7. Stellen Sie Connect zu der EC2 Instance her, die Sie zuvor erstellt haben, indem Sie die Schritte unter Connect Ihrer Linux-Instance herstellen im EC2 Amazon-Benutzerhandbuch für Linux-Instances befolgen.

   Wir empfehlen, dass Sie eine Verbindung zu Ihrer EC2 Instance herstellen mithilfe vonSSH. Wenn das SSH Client-Utility unter Windows, Linux oder Mac installiert ist, können Sie mithilfe des folgenden Befehlsformats eine Verbindung zur Instance herstellen:

   ```
   ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
   ```

   Gehen Sie beispielsweise davon aus, dass `ec2-database-connect-key-pair.pem` es `/dir1` unter Linux gespeichert ist und die Öffentlichkeit IPv4 DNS für Ihre EC2 Instance schon`ec2-12-345-678-90.compute-1.amazonaws.com`. Ihr SSH Befehl würde wie folgt aussehen:

```
ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
user@ec2-12-345-678-90.compute-1.amazonaws.com
```

8. Installieren Sie die neueste Version von Telegraf auf Ihrer Instanz. Verwenden Sie dazu den folgenden Befehl:

```
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. Konfigurieren Sie Ihre Telegraf-Instanz.

> **Note**
>
> Wenn telegraf.conf nicht existiert oder keinen `timestream` Abschnitt enthält, können Sie einen generieren mit:
>
> ```
> telegraf –section-filter agent:inputs:outputs –input-filter cpu:mem –output-
> filter timestream config > telegraf.conf
> ```

   a. Bearbeiten Sie die Konfigurationsdatei, die sich normalerweise unter befindet. `/etc/telegraf`

   ```
   sudo nano /etc/telegraf/telegraf.conf
   ```

   b. Konfigurieren Sie grundlegende Eingaben fürCPU, MEM undDISK.

   ```
   [[inputs.cpu]]
     percpu = true
     totalcpu = true
     collect_cpu_time = false
     report_active = false
   ```

```
[[inputs.mem]]

[[inputs.disk]]
  ignore_fs = ["tmpfs", "devtmpfs", "devfs"]
```

c. Konfigurieren Sie das Output-Plug-In so, dass es Daten an Ihre InfluxDB-DB-Instance sendet und Ihre Änderungen speichert.

```
[[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. Konfigurieren Sie das Timestream-Ziel.

```
# 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. Aktivieren und starten Sie den Telegraf-Dienst.

```
$ sudo systemctl enable telegraf
```

```
$ sudo systemctl start telegraf
```

Schritt 4: Löschen Sie die EC2 Amazon-Instance und die InfluxDB-DB-Instance

Nachdem Sie die von Telegraph generierten Daten mithilfe Ihrer InfluxDB-DB-Instance mit der InfluxUI untersucht haben, löschen Sie sowohl Ihre als auch Ihre InfluxDB-DB-Instances, EC2 damit Ihnen keine Gebühren mehr berechnet werden.

Um die Instanz zu löschen: EC2

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die EC2 Amazon-Konsole unter https://console.aws.amazon.com/ec2/.
2. Wählen Sie im Navigationsbereich Instances aus.
3. Wählen Sie die EC2 Instance aus, wählen Sie Instance-Status und anschließend Instanz beenden.
4. Wenn Sie zur Bestätigung aufgefordert werden, wählen Sie Beenden aus.

Weitere Informationen zum Löschen einer EC2 Instance finden Sie unter Kündigen Ihrer Instance im EC2 Amazon-Benutzerhandbuch.

Um die DB-Instance ohne endgültigen DB-Snapshot zu löschen:

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Timestream for InfluxDB-Konsole unter. https://console.aws.amazon.com/timestream/
2. Wählen Sie im Navigationsbereich InfluxDB-Datenbanken aus.
3. Wählen Sie die zu löschende DB-Instance aus.
4. Klicken Sie bei Actions auf Delete.
5. Bestätigen Sie, und wählen Sie Löschen.

(Optional) Stellen Sie mithilfe von Amazon Managed Grafana eine Verbindung zu Ihrer DB-Instance her

Sie können Amazon Managed Grafana verwenden, um Dashboards zu erstellen und die Leistung Ihrer EC2 Instances mithilfe von Amazon Timestream for InfluxDB zu überwachen. Amazon Managed Grafana ist ein vollständig verwalteter Service für Grafana, eine beliebte Open-Source-

Analyseplattform, mit der Sie Ihre Metriken, Protokolle und Traces abfragen, visualisieren und Warnmeldungen dazu erhalten können.

Erstellen Sie ein neues Operator-Token für Ihre InfluxDB-Instance

Wenn Sie das Operator-Token für Ihre neue InfluxDB-Instanz benötigen, führen Sie die folgenden Schritte aus:

1. Um Ihr Operator-Token zu ändern, empfehlen wir, den Influx zu verwenden. CLI Eine Anleitung finden Sie unter: Influx CLI installieren und verwenden.
2. Konfigurieren Sie Ihren CLI Computer`--username-password`, um den Operator erstellen zu können:

   ```
   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. Erstellen Sie Ihr neues Operator-Token. Sie werden nach Ihrem Passwort gefragt, um diesen Schritt zu bestätigen.

   ```
   influx auth create --org [YOURORG] --operator
   ```

> **Important**
>
> Sobald ein neues Operator-Token erstellt wurde, müssen Sie alle Clients aktualisieren, die derzeit das alte verwenden.

Migrieren Sie Daten von selbstverwaltetem InfluxDB zu Timestream for InfluxDB

Das Influx-Migrationsskript ist ein Python-Skript, das Daten zwischen OSS InfluxDB-Instanzen migriert, unabhängig davon, ob diese Instanzen von verwaltet werden oder nicht. AWS

InfluxDB ist eine Zeitreihendatenbank. InfluxDB enthält Punkte, die eine Reihe von Schlüssel-Wert-Paaren und einen Zeitstempel enthalten. Wenn Punkte nach Schlüssel-Wert-Paaren gruppiert werden, bilden sie eine Reihe. Eine Reihe wird nach einer Zeichenketten-ID gruppiert, die als

Messung bezeichnet wird. InfluxDB wird häufig für Betriebsüberwachung, IOT Daten und Analysen verwendet. Ein Bucket ist eine Art Container innerhalb von InfluxDB zum Speichern von Daten. AWS-managed InfluxDB ist InfluxDB innerhalb des Ökosystems. AWS InfluxDB stellt InfluxDB v2 für den Zugriff auf Daten und das Vornehmen von Änderungen an der Datenbank API bereit. Die InfluxDB v2 verwendet das API Influx-Migrationsskript, um Daten zu migrieren.

- Das Influx-Migrationsskript kann Buckets und ihre Metadaten migrieren, alle Buckets aus allen Organisationen migrieren oder eine vollständige Migration durchführen, bei der alle Daten auf der Zielinstanz ersetzt werden.
- Das Skript sichert Daten aus der Quellinstanz lokal auf dem System, auf dem das Skript ausgeführt wird, und stellt die Daten dann auf der Zielinstanz wieder her. Die Daten werden in den Verzeichnissen code>influxdb-backup-<timestamp></timestamp> aufbewahrt, eines für jede Migration.
- Das Skript bietet eine Reihe von Optionen und Konfigurationen, darunter das Mounten von S3-Buckets zur Begrenzung der lokalen Speichernutzung während der Migration und die Auswahl der Organisationen, die während der Migration verwendet werden sollen.

Themen

Vorbereitung

Die Datenmigration für InfluxDB wird mit einem Python-Skript durchgeführt, das CLI InfluxDB-Funktionen und InfluxDB v2 nutzt. API Die Ausführung des Migrationsskripts erfordert die folgende Umgebungskonfiguration:

- Unterstützte Versionen: Eine Mindestversion von 2.3 von InfluxDB und Influx CLI wird unterstützt.
- Token-Umgebungsvariablen
  - Erstellen Sie die Umgebungsvariable`INFLUX_SRC_TOKEN`, die das Token für Ihre InfluxDB-Quellinstanz enthält.
  - Erstellen Sie die Umgebungsvariable, die das Token für Ihre InfluxDB-Zielinstanz `INFLUX_DEST_TOKEN` enthält.
- Python 3

  - Installation überprüfen:`python3 --version`.
  - Falls nicht installiert, installieren Sie von der Python-Website. Mindestversion 3.7 erforderlich. Unter Windows ist der Standard-Python-3-Alias einfach Python.
  - Die Python-Modulanforderungen sind erforderlich. Installiere es mit: `shell python3 -m pip install requests`
  - TTheDas Python-Modul influxdb_client ist erforderlich. Installiere es mit: `shell python3 -m pip install influxdb_client`
- InfluxDB CLI
  - Installation bestätigen:. `influx version`
  - Falls nicht installiert, folgen Sie der Installationsanleitung in der InfluxDB-Dokumentation.

    Füge Zustrom zu deinem $ hinzu. PATH
- S3-Montagewerkzeuge (optional)

  Wenn S3-Mount verwendet wird, werden alle Sicherungsdateien in einem benutzerdefinierten S3-Bucket gespeichert. Das S3-Mounten kann nützlich sein, um Speicherplatz auf dem ausführenden Computer zu sparen oder wenn Backup-Dateien gemeinsam genutzt werden müssen. Wenn das S3-Mounten nicht verwendet wird, wird durch Weglassen der `--s3-bucket` Option ein lokales `influxdb-backup-<millisecond timestamp>` Verzeichnis erstellt, um die Sicherungsdateien in demselben Verzeichnis zu speichern, in dem das Skript ausgeführt wurde.

  Für Linux: mountpoint-s3.

  Für Windows: rclone (eine vorherige rclone-Konfiguration ist erforderlich).
- Festplattenkapazität
  - Der Migrationsprozess erstellt automatisch eindeutige Verzeichnisse zum Speichern von Gruppen von Sicherungsdateien und speichert diese Sicherungsverzeichnisse je nach den angegebenen Programmargumenten entweder in S3 oder im lokalen Dateisystem.
  - Stellen Sie sicher, dass genügend Speicherplatz für die Datenbanksicherung vorhanden ist, idealerweise die doppelte Größe der vorhandenen InfluxDB-Datenbank, wenn Sie die `--s3-bucket` Option weglassen und lokalen Speicher für Sicherung und Wiederherstellung verwenden.
  - Überprüfen Sie den Speicherplatz mit `df -h (UNIX/Linux)` oder, indem Sie die Laufwerkseigenschaften unter Windows überprüfen.
- Direkte Verbindung

Stellen Sie sicher, dass eine direkte Netzwerkverbindung zwischen dem System, auf dem das Migrationsskript ausgeführt wird, und den Quell- und Zielsystemen besteht. `influx ping --host <host>`ist eine Möglichkeit, eine direkte Verbindung zu überprüfen.

Wie benutzt man das Skript

Ein einfaches Beispiel für die Ausführung des Skripts ist der Befehl:

```
python3 influx_migration.py --src-host <source host> --src-bucket <source bucket> --
dest-host <destination host>
```

Welches migriert einen einzelnen Bucket.

Alle Optionen können angezeigt werden, indem Sie Folgendes ausführen:

```
python3 influx_migration.py -h
```

Usage

```
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]
```

Optionen

- -confirm-full (optional): Bei Verwendung von `--full` without `--csv` werden alle Token, Benutzer, Buckets, Dashboards und alle anderen Schlüsselwertdaten in der Zieldatenbank durch die Token, Benutzer, Buckets, Dashboards und alle anderen Schlüsselwertdaten in der Quelldatenbank ersetzt. `--full`with migriert `--csv` nur alle Bucket- und Bucket-Metadaten, einschließlich Bucket-Organisationen. Diese Option (`--confirm-full`) bestätigt eine vollständige Migration und wird ohne Benutzereingabe fortgesetzt. Wenn diese Option nicht zur Verfügung steht und sie bereitgestellt und `--csv` nicht bereitgestellt `--full` wurde, wird die Ausführung des Skripts angehalten und auf die Bestätigung durch den Benutzer gewartet. Dies ist eine wichtige Aktion. Gehen Sie vorsichtig vor. Standardwert "false".
- -csv (optional): Gibt an, ob CSV-Dateien zum Sichern und Wiederherstellen verwendet werden sollen. Wenn ebenfalls übergeben `--full` wird, werden alle benutzerdefinierten Buckets in allen Organisationen migriert, nicht System-Buckets, Benutzer, Token oder Dashboards. Wenn für alle

Buckets auf dem Zielserver anstelle der bereits vorhandenen Quellorganisationen eine einzige Organisation gewünscht wird, verwenden Sie. `--dest-org`

- -dest-bucket DEST _ BUCKET (optional): Der Name des InfluxDB-Buckets auf dem Zielserver darf kein bereits vorhandener Bucket sein. Standardmäßig der Wert von oder, falls nicht angegeben. `--src-bucket` `None` `--src-bucket`
- -dest-host DEST _HOST: Der Host für den Zielserver. Beispiel: http://localhost:8086.
- -dest-org DEST _ ORG (optional): Der Name der Organisation, in der Buckets auf dem Zielserver wiederhergestellt werden sollen. Wenn dieser Wert weggelassen wird, behalten alle migrierten Buckets vom Quellserver ihre ursprüngliche Organisation bei und migrierte Buckets sind möglicherweise nicht auf dem Zielserver sichtbar, ohne dass Organisationen erstellt und gewechselt werden. Dieser Wert wird bei allen Arten der Wiederherstellung verwendet, unabhängig davon, ob es sich um einen einzelnen Bucket, eine vollständige Migration oder eine Migration mit CSV-Dateien für Sicherung und Wiederherstellung handelt.
- -dir-name DIR _ NAME (optional): Der Name des zu erstellenden Backup-Verzeichnisses. Standardeinstellung: `influxdb-backup-<timestamp>`. Darf noch nicht existieren.
- -full (optional): Gibt an, ob eine vollständige Wiederherstellung durchgeführt werden soll, bei der alle Daten auf dem Zielserver durch alle Daten vom Quellserver aller Organisationen ersetzt werden, einschließlich aller Schlüsselwertdaten wie Token, Dashboards, Benutzer usw. `--src-bucket`Überschreibt und. `--dest-bucket` Bei Verwendung mit `--csv` werden nur Daten und Metadaten von Buckets migriert. Standardwert "false".
- h, --help: Zeigt eine Hilfemeldung an und beendet das Programm.
- -log-level LOG _ LEVEL (optional): Die Protokollebene, die während der Ausführung verwendet werden soll. Die Optionen sind Debug, Error und Info. Die Standardeinstellung ist info.
- - retry-restore-dir RETRY _ RESTORE _ DIR (optional): Verzeichnis, das für die Wiederherstellung verwendet werden soll, wenn eine vorherige Wiederherstellung fehlgeschlagen ist, überspringt die Sicherung und die Verzeichniserstellung, schlägt fehl, wenn das Verzeichnis nicht existiert, kann ein Verzeichnis in einem S3-Bucket sein. Wenn eine Wiederherstellung fehlschlägt, wird der Pfad des Backup-Verzeichnisses, der für die Wiederherstellung verwendet werden kann, relativ zum aktuellen Verzeichnis angegeben. S3-Buckets werden in der folgenden Form vorliegen`influxdb-backups/<s3 bucket>/<backup directory>`. Der Standardname des Backup-Verzeichnisses lautet`influxdb-backup-<timestamp>`.
- -s3-bucket S3_ BUCKET (optional): Der Name des S3-Buckets, der zum Speichern von Sicherungsdateien verwendet werden soll. Unter Linux ist dies einfach der Name des S3-Buckets, z. B. `my-bucket` wurden angegebene Variablen `AWS_ACCESS_KEY_ID` und `AWS_SECRET_ACCESS_KEY` Umgebungsvariablen gesetzt oder existieren. `${HOME}/.aws/`

`credentials` Unter Windows ist dies der `rclone` konfigurierte Remote- und Bucket-Name, z. `my-remote:my-bucket` B. Alle Sicherungsdateien verbleiben nach der Migration in einem erstellten `influxdb-backups-<timestamp>` Verzeichnis im S3-Bucket. In dem Verzeichnis, von dem aus dieses Skript ausgeführt `influx-backups` wird, wird ein temporäres Mount-Verzeichnis mit dem Namen erstellt. Falls nicht angegeben, werden alle Sicherungsdateien lokal in einem erstellten `influxdb-backups-<timestamp>` Verzeichnis gespeichert, von dem aus das Skript ausgeführt wird.

- -skip-verify (optional): Überspringt TLS die Zertifikatsüberprüfung.
- -src-bucket SRC _ BUCKET (optional): Der Name des InfluxDB-Buckets auf dem Quellserver. Wenn nicht angegeben, muss es bereitgestellt werden. `--full`
- -src-host SRC _ HOST (optional): Der Host für den Quellserver. Die Standardeinstellung ist http://localhost:8086.

Wie bereits erwähnt, `mountpoint-s3` `rclone` werden sie benötigt, wenn sie `--s3-bucket` verwendet werden sollen, können aber ignoriert werden`--s3-bucket`, wenn der Benutzer keinen Wert für angibt. In diesem Fall werden die Sicherungsdateien lokal in einem eindeutigen Verzeichnis gespeichert.

Überblick über die Migration

Nach Erfüllung der Voraussetzungen:

1. Migrationsskript ausführen: Führen Sie mit einer Terminal-App Ihrer Wahl das Python-Skript aus, um Daten von der InfluxDB-Quellinstanz zur InfluxDB-Zielinstanz zu übertragen.
2. Anmeldeinformationen angeben: Geben Sie Hostadressen und Ports als Optionen an. CLI
3. Daten überprüfen: Stellen Sie sicher, dass die Daten korrekt übertragen werden, indem Sie:

    a. Verwenden der InfluxDB-Benutzeroberfläche und Überprüfen von Buckets.

    b. Buckets auflisten mit. `influx bucket list -t <destination token> --host <destination host address> --skip-verify`

    c. Wird verwendet `influx v1 shell -t <destination token> --host <destination host address> --skip-verify` und ausgeführt`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>`, um zu überprüfen, ob die korrekte Anzahl von Datensätzen migriert wurde.

Example Beispiel für einen Lauf

1. Öffnen Sie eine Terminal-App Ihrer Wahl und stellen Sie sicher, dass die erforderlichen Voraussetzungen ordnungsgemäß installiert sind:

```
~ > 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. Navigieren Sie zum Migrationsskript:

```
~ > 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. Bereiten Sie die folgenden Informationen vor:

   a. Name des Quell-Buckets, der migriert werden soll.

   b. (Optional) Wählen Sie einen neuen Bucket-Namen für den migrierten Bucket auf dem Zielserver.

   c. Root-Token für Quell- und Ziel-Influx-Instances.

   d. Hostadresse der Quell- und Ziel-Influx-Instances.

   e. (Optional) Name und Anmeldeinformationen für den S3-Bucket; die AWS Command Line Interface Anmeldeinformationen sollten in den Umgebungsvariablen des Betriebssystems festgelegt werden.

```
# AWS credentials (for timestream testing)
    export AWS_ACCESS_KEY_ID="xxx"
    export AWS_SECRET_ACCESS_KEY="xxx"
```

   f. Konstruieren Sie den Befehl wie folgt:

```
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. Führen Sie das Skript aus:

h. Warten Sie, bis die Ausführung des Skripts abgeschlossen ist.

i. Überprüfen Sie den neu migrierten Bucket auf Datenintegrität,`performance.txt`. Diese Datei befindet sich in demselben Verzeichnis, in dem das Skript ausgeführt wurde, und enthält einige grundlegende Informationen darüber, wie lange die einzelnen Schritte gedauert haben.

Migrationszenarien

Example Beispiel 1: Einfache Migration mit lokalem Speicher

Sie möchten einen einzelnen Bucket, den Primär-Bucket, vom Quellserver (`http://localhost:8086`) auf einen Zielserver migrieren. (`http://dest-server-address:8086`)

Nachdem Sie sichergestellt haben, dass Sie TCP Zugriff (für den HTTP Zugriff) auf beide Computer haben, auf denen die InfluxDB-Instanzen auf Port 8086 gehostet werden, und Sie haben sowohl Quell- als auch Ziel-Token und diese als Umgebungsvariablen bzw. zur `INFLUX_SRC_TOKEN` zusätzlichen `INFLUX_DEST_TOKEN` Sicherheit gespeichert:

```
python3 influx_migration.py --src-bucket primary-bucket --src-host http://
localhost:8086 --dest-host http://dest-server-address:8086
```

Die Ausgabe sollte folgendermaßen oder ähnlich aussehen:

```
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
```

Das Verzeichnis `influxdb-backup-<timestamp>` wird erstellt und in dem Verzeichnis gespeichert, von dem aus das Skript ausgeführt wurde. Es enthält Sicherungsdateien.

Example Beispiel 2: Vollständige Migration mit lokalem Speicher und Debug-Protokollierung

Wie oben, außer dass Sie mit der Option alle Buckets, Token, Benutzer und Dashboards migrieren, die Buckets auf dem Zielserver löschen und ohne Benutzerbestätigung einer vollständigen Datenbankmigration fortfahren möchten. `--confirm-full` Sie möchten auch sehen, was die Leistungsmesswerte sind, damit Sie die Debug-Protokollierung aktivieren können.

```
python3 influx_migration.py --full --confirm-full --src-host http://localhost:8086 --
dest-host http://dest-server-address:8086 --log-level debug
```

Die Ausgabe sollte folgendermaßen oder ähnlich aussehen:

```
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 Beispiel 3: Vollständige Migration mithilfe von CSV Zielorganisation und S3-Bucket

Wie im vorherigen Beispiel, aber mit Linux oder Mac und Speichern der Dateien im S3-Bucket`my-s3-bucket`. Dadurch wird vermieden, dass Backup-Dateien die lokale Speicherkapazität überlasten.

```
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
```

Die Ausgabe sollte folgendermaßen oder ähnlich aussehen:

```
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
```

Konfigurieren einer DB-Instance

Dieser Abschnitt zeigt, wie Sie Ihre Amazon Timestream for InfluxDB-DB-Instance einrichten. Bevor Sie eine DB-Instance erstellen, entscheiden Sie sich für die DB-Instance-Klasse, die die DB-Instance ausführt. Entscheiden Sie außerdem, wo die DB-Instance ausgeführt werden soll, indem Sie eine Region auswählen. AWS Als Nächstes erstellen Sie die DB-Instance.

Sie können eine DB-Instance mit einer DB-Parametergruppe konfigurieren. Eine DB-Parametergruppe fungiert als Container für Engine-Konfigurationswerte, die auf eine oder mehrere DB-Instances angewendet werden.

Die verfügbaren Parameter hängen von der DB-Engine und der DB-Engine-Version ab. Sie können eine DB-Parametergruppe angeben, wenn Sie eine DB-Instance erstellen. Sie können eine DB-Instance auch ändern, um sie anzugeben.

> **Important**
>
> Derzeit können Sie die Compute- (Instance-Typen) und die Speicherkonfiguration (Storage-Typen) vorhandener Instances nicht ändern.

Erstellen einer DB-Instance

Verwenden der Konsole

1. Melden Sie sich bei Amazon Timestream for InfluxDB an AWS Management Console und öffnen Sie es.
2. Wählen Sie in der oberen rechten Ecke der Amazon Timestream for InfluxDB-Konsole die AWS Region aus, in der Sie die DB-Instance erstellen möchten.
3. Wählen Sie im Navigationsbereich InfluxDB-Datenbanken aus.
4. Wählen Sie Create Influx database.
5. Geben Sie für DB Instance Identifier einen Namen ein, der Ihre Instance identifiziert.
6. Geben Sie die grundlegenden InfluxDB-Konfigurationsparameter Benutzername, Organisation, Bucket-Name und Passwort an.

   > **Important**
   >
   > Ihr Benutzername, Ihre Organisation, Ihr Bucket-Name und Ihr Passwort werden als Geheimnis in AWS Secrets Manager gespeichert, das für Ihr Konto erstellt wird.

   Wenn Sie das Benutzerpasswort ändern müssen, nachdem die DB-Instance verfügbar ist, können Sie es mithilfe von Influx CLI ändern.
7.
8. Wählen Sie für DB Instance Class eine Instance-Größe aus, die Ihren Workload-Anforderungen besser entspricht.

9. Wählen Sie für DB-Speicherklasse eine Speicherklasse aus, die Ihren Anforderungen entspricht. In allen Fällen müssen Sie nur den zugewiesenen Speicher konfigurieren.
10. Stellen Sie im Abschnitt Konnektivitätskonfiguration sicher, dass sich Ihre InfluxDB-Instance im selben Subnetz befindet wie Ihre neuen Clients, die Konnektivität zu Ihrer Timestream for InfluxDB-DB-Instance benötigen. Sie können sich auch dafür entscheiden, Ihre DB-Instance öffentlich verfügbar zu machen.
11. Wählen Sie Create Influx database.
12. Wählen Sie in der Liste Datenbanken den Namen Ihrer neuen InfluxDB-Instanz aus, um deren Details anzuzeigen. Die DB-Instance hat den Status Creating, bis sie einsatzbereit ist.
13. Wenn sich der Status in Available (Verfügbar) ändert, können Sie die Verbindung zur DB-Instance herstellen. Abhängig von der Klasse der DB-Instance und vom verfügbaren Speicherplatz kann es bis zu 20 Minuten dauern, bis die neue DB-Instance verfügbar ist.

Unter Verwendung der CLI

Um eine DB-Instance mit dem zu erstellen AWS Command Line Interface, rufen Sie den `create-db-instance` Befehl mit den folgenden Parametern auf:

```
--name
--vpc-subnet-ids
--vpc-security-group-ids
--db-instance-type
--db-storage-type
--username
--organization
--password
--allocated-storage
```

Weitere Informationen zu den einzelnen Einstellungen finden Sie unter Einstellungen für DB-Instances.

Example Beispiel: Verwenden von Standard-Engine-Konfigurationen

Für Linux, macOS oder 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
```

Für 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
```

Mit dem API

Um eine DB-Instance mit dem zu erstellen AWS Command Line Interface, rufen Sie den `CreateDBInstance` Befehl mit den folgenden Parametern auf:

Weitere Informationen zu den einzelnen Einstellungen finden Sie unter Einstellungen für DB-Instances.

> **Important**
>
> Teil des DBInstance Antwortobjekts, das Sie erhalten influxAuthParametersSecretArn. Dies wird ein ARN SecretsManager Geheimnis in Ihrem Konto halten. Es wird erst gefüllt, wenn Ihre InfluxDB-DB-Instances verfügbar sind. Das Geheimnis enthält Parameter für die Eingangsauthentifizierung, die während des Prozesses bereitgestellt werden. `CreateDbInstance` Dabei handelt es sich updates/modifications/deletions um eine READONLY Kopie, da sich ein beliebiger Teil dieses Geheimnisses nicht auf die erstellte DB-Instance auswirkt. Wenn Sie dieses Geheimnis löschen, bezieht sich unsere API Antwort immer noch auf das gelöschte GeheimnisARN.

Sobald Sie mit der Erstellung Ihrer Timestream for InfluxDB-DB-Instance fertig sind, empfehlen wir Ihnen, den Influx herunterzuladen, zu installieren und zu konfigurieren. CLI

Der Influx CLI bietet eine einfache Möglichkeit, über eine Befehlszeile mit InfluxDB zu interagieren. Detaillierte Installations- und Einrichtungsanweisungen finden Sie unter Use the Influx. CLI

Einstellungen für DB-Instances

Sie können eine DB-Instance mithilfe der Konsole, des `create-db-instance` CLI Befehls oder des `CreateDBInstance` Timestreams für den InfluxDB-Betrieb erstellen. API

Die folgende Tabelle enthält Details zu den Einstellungen, die Sie beim Erstellen einer DB-Instance auswählen.

Konsoleneinstellung	Beschreibung	CLIOption und Timestream-Parameter API
Allocated storage	Die Größe des zuzuteilenden Speichers für die DB-Instance in Gibibytes. In einigen Fällen verbessert das Zuweisen einer die Größe Ihrer Datenbank übertreffenden Speicherkapazität für Ihre DB-Instance die I/O-Leistung. Weitere Informationen finden Sie unter InfluxDB-Instanzspeicher.	CLI: `allocated-storage` API: `allocatedstorage`
Bucket-Name	Ein Name für den Bucket, um die Instanz zu initialisieren InfluxDb	CLI: `bucket` API: `bucket`
DB-Instance-Typ	Die Konfiguration für Ihre DB-Instance. Eine DB-Instance-Klasse db.influx.large hat beispielsweise 16 GiB Arbeitsspeicher, 2, speicheroptimiert. vCPUs Wählen Sie nach Möglichkeit einen DB-Instance-Typ, der groß genug ist, dass ein typischer Abfrage-Arbeitssatz im Speicher gespeichert werden kann. Wenn Arbeitssätze im Arbeitsspeicher gehalten werden, kann das System das Schreiben auf die Festplatte vermeiden, was die Leistung verbessert.	CLI: `db-instance-type` API: `Dbinstancetype`

Konsoleneinstellung	Beschreibung	CLIOption und Timestream-Parameter API
	Weitere Informationen finden Sie unter DB-Instance-Klassenarten.	
DB-Instance-Kennung	Der Name der DB-Instance. Benennen Sie Ihre DB-Instances auf die gleiche Weise wie Ihre lokalen Server. Ihre DB-Instance-ID kann bis zu 63 alphanumerische Zeichen enthalten und muss für Ihr Konto in der von Ihnen ausgewählten AWS Region eindeutig sein.	CLI: `db-instance-identifier` API: `Dbinstanceidentifier`
DB-Parametergruppe	Eine Parametergruppe für die DB-Instance. Sie können die Standardparametergruppe wählen oder eine benutzerdefinierte Parametergruppe erstellen. Weitere Informationen finden Sie unter Arbeiten mit DB-Parametergruppen..	CLI: `db-parameter-group-name` API: `DBParameterGroupName`
Einstellung für die Protokollzustellung	Der Name des S3-Buckets, in dem die InfluxDB-Protokolle gespeichert werden.	CLI: `LogDeliveryConfiguration` API: `log-delivery-configuration`

Konsoleneinstellung	Beschreibung	CLIOption und Timestream-Parameter API
Multi-AZ-Bereitstellung	Erstellen Sie eine Standby-Instance, um eine passive, sekundäre Kopie Ihrer DB-Instance in einer anderen Availability Zone für die Failover-Unterstützung zu erstellen. Wir empfehlen Multi-AZ, um die hohe Verfügbarkeit von Produktions-Workloads sicherzustellen. Für die Entwicklung und das Testen können Sie Do not create a standby instance (Keine Standby-Instance erstellen) auswählen. Weitere Informationen finden Sie unter Konfiguration und Verwaltung einer Multi-AZ-Bereitstellung.	CLI: `MultiAz` API: `multi-az`
Passwort	Dies wird Ihr Master-Passwort sein, mit dem Sie Ihre InfluxDB-DB-Instance initialisieren. Sie werden dieses Passwort verwenden, um sich bei InfluxUI anzumelden und Ihr Operator-Token zu erhalten.	CLI: `password` API: `password`

Konsoleneinstellung	Beschreibung	CLIOption und Timestream-Parameter API
Öffentlicher Zugang	Ja, um der DB-Instance eine öffentliche IP-Adresse zu geben, was bedeutet, dass sie von außerhalb des zugänglich istVPC. Um öffentlich zugänglich zu sein, muss sich die DB-Instance auch in einem öffentlichen Subnetz im VPC befinden. Nein, um die DB-Instance nur von innerhalb des VPC aus zugänglich zu machen. Um von außerhalb eine Verbindung zu einer DB-Instance herzustellenVPC, muss die DB-Instance öffentlich zugänglich sein. Außerdem muss der Zugriff unter Verwendung der Regeln für eingehenden Datenverkehr der Sicherheitsgruppe der DB-Instance gewährt werden. Darüber hinaus müssen andere Anforderungen erfüllt sein.	CLI: `publicly-accessible` API: `PubliclyAccessible`
Speichertyp	Der Speichertyp für Ihre DB-Instance Sie können je nach Ihren Workload-Anforderungen zwischen 3 verschiedenen Typen wählen: Speicher IOPS inklusive bereitgestellter Zufluss: * Influx inklusive: 3000 IOPS IOPS * Zustrom inklusive 12000 IOPS IOPS * 16000 enthalten INflux IOPS IOPS Weitere Informationen finden Sie unter InfluxDB-Instanzspeicher.	CLI: `db-storage-type` API: `DbStorageType`

Konsoleneinstellung	Beschreibung	CLIOption und Timestream-Parameter API
Ursprünglicher Nutzername	Dies ist der Master-Benutzer, mit dem Sie Ihre InfluxDB-DB-Instance initialisieren. Sie werden diesen Benutzernamen verwenden, um sich bei der InfluxUI anzumelden und Ihr Operator-Token zu erhalten.	CLI: `username` API: `Username`
Subnetze	Ein VPC-Subnetz, das dieser DB-Instance zugeordnet werden soll.	CLI: `vpc-subnet-ids` API: `VPCSubnetIds`
VPCSicherheitsgruppe (Firewall)	Die der DB-Instance zugeordneten Sicherheitsgruppe.	CLI: `vpc-security-group-ids` API: `VPCSecurityGroupIds`

Verbindung zu einer Amazon Timestream for InfluxDB-DB-Instance herstellen

Bevor Sie eine Verbindung mit einer DB-Instance herstellen können, müssen Sie die DB-Instance erstellen. Weitere Informationen finden Sie unter Erstellen einer DB-Instance. Nachdem Amazon Timestream Ihre DB-Instance bereitgestellt hat, verwenden Sie InfluxDBAPI, Influx oder einen anderen kompatiblen Client CLI oder Hilfsprogramm für InfluxDB, um eine Verbindung zur DB-Instance herzustellen.

Themen

- Suchen der Verbindungsinformationen für eine Amazon Timestream for InfluxDB-DB-Instance
- Datenbankauthentifizierungsoptionen
- Arbeiten mit Parametergruppen

Suchen der Verbindungsinformationen für eine Amazon Timestream for InfluxDB-DB-Instance

Die Verbindungsinformationen für eine DB-Instance umfassen ihren Endpunkt, Port, Benutzernamen, Passwort und ein gültiges Zugriffstoken, z. B. den Operator oder das All-Access-Token.
Nehmen wir beispielsweise für eine InfluxDB-DB-Instance an, dass der Endpunktwert ist. `influxdb1-123456789.us-east-1.timestream-influxdb.amazonaws.com` In diesem Fall ist der Portwert 8086 und der Datenbankbenutzer ist admin. Angesichts dieser Informationen geben Sie die folgenden Werte in einer Verbindungszeichenfolge an:

- Geben `influxdb1-123456789.us-east-1.timestream-influxdb.amazonaws.com` Sie für Host oder Hostname oder DNS -namen an.
- Geben Sie als Port 8086 an.
- Geben Sie als Benutzer admin an.
- Geben Sie als Passwort das an, das Sie bei der Erstellung Ihrer DB-Instance angegeben haben.

> **Important**
>
> Wenn Sie Ihren Timestream für die InfluxDB-DB-Instance erstellt haben, erhalten Sie einen Teil des DBInstance Antwortobjekts. `influxAuthParametersSecretArn` Dadurch wird ein SecretsManager Geheimnis in Ihrem Konto geheim gehalten. Es wird erst gefüllt, wenn Ihre InfluxDB-DB-Instances verfügbar sind. Das Geheimnis enthält Parameter für die Eingangsauthentifizierung, die während des Prozesses bereitgestellt werden. `CreateDbInstance` Dabei handelt es sich updates/modifications/deletions um eine READONLY Kopie, da sich ein beliebiger Teil dieses Geheimnisses nicht auf die erstellte DB-Instance auswirkt. Wenn Sie dieses Geheimnis löschen, bezieht sich unsere API Antwort immer noch auf das gelöschte Geheimnis arn.

Der Endpunkt ist für jede DB-Instance eindeutig, und die Werte des Ports und des Benutzers können variieren. Um eine Verbindung zu einer DB-Instance herzustellen, können Sie InfluxCLI, Influx oder einen beliebigen mit API InfluxDB kompatiblen Client verwenden.

Verwenden Sie die Management Console, um die Verbindungsinformationen für eine DB-Instance zu finden. AWS Sie können auch den Befehl AWS Command Line Interface (AWS CLI) `describe-db-instances` oder den Vorgang API `GetDBInstance` TimeStream-InfluxDB verwenden.

Mit dem AWS Management Console

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Timestream Timestream-Konsole.
2. Wählen Sie im Navigationsbereich InfluxDB-Datenbanken aus, um eine Liste Ihrer DB-Instances anzuzeigen.
3. Wählen Sie den Namen der DB-Instance aus, um ihre Details anzuzeigen.
4. Kopieren Sie im Abschnitt Zusammenfassung den Endpunkt. Notieren Sie sich auch die Portnummer. Sie benötigen sowohl den Endpunkt als auch die Portnummer, um die Verbindung zur DB-Instance herzustellen.

Wenn Sie den Benutzernamen und das Passwort suchen müssen, wählen Sie die Registerkarte Konfigurationsdetails und dann die, `influxAuthParametersSecretArn` um auf Ihren Secrets Manager zuzugreifen.

Verwenden Sie den CLI

- Um die Verbindungsinformationen für eine InfluxDB-DB-Instance mithilfe von zu finden AWS CLI, rufen Sie den get-db-instance Befehl auf. Fragen Sie im Aufruf nach der DB-Instance-ID, dem Endpunkt, dem Port und influxAuthParameters dem Secret ab.

  Für Linux, macOS oder Unix:

  ```
  aws timestream-influxdb get-db-instance --identifier id \
   --query "[name,endpoint,influxAuthParametersSecretArn]"
  ```

  Für Windows:

  ```
  aws timestream-influxdb get-db-instance --identifier id \
   --query "[name,endpoint,influxAuthParametersSecretArn]"
  ```

  Die Ausgabe sollte in etwa wie folgt aussehen. Um auf die Benutzernameninformationen zuzugreifen, müssen Sie das überprüfen`InfluxAuthParameterSecret`.

  ```
  [
      [
          "mydb",
          "mydb-123456789012.us-east-1.timestream-influxdb.amazonaws.com",
  ```

```
        8086,
    ]
]
```

Zugriffstoken erstellen

Mit diesen Informationen können Sie eine Verbindung für Ihre Instance herstellen, um Ihre Zugriffstoken abzurufen oder zu erstellen. Es gibt mehrere Möglichkeiten, dies zu erreichen:

Mit dem CLI

1. Falls Sie es noch nicht getan haben, laden Sie den Influx CLI herunter, installieren und konfigurieren Sie ihn.
2. Verwenden Sie bei der Konfiguration Ihrer CLI Influx-Konfiguration die Option `--username-password` zur Authentifizierung.

   ```
   influx config create --config-name YOUR_CONFIG_NAME  --host-url "https://
   yourinstance.timestream-influxdb.amazonaws.com:8086"  --org yourorg --username-
   password admin --active
   ```

3. Verwenden Sie den Befehl influx auth create, um Ihr Operator-Token neu zu erstellen. Beachten Sie, dass durch diesen Vorgang das alte Operator-Token ungültig wird.

   ```
   influx auth create --org kronos --operator
   ```

4. Sobald Sie das Operator-Token haben, können Sie den Befehl influx auth list verwenden, um alle Ihre Tokens anzuzeigen. Sie können den Befehl influx auth create verwenden, um ein All-Access-Token zu erstellen.

> **Important**
>
> Sie müssen diesen Schritt ausführen, um zuerst Ihr Operator-Token zu erhalten und dann neue Token mit der API InfluxDB oder erstellen zu können. CLI

Verwenden der Influx-Benutzeroberfläche

1. Navigieren Sie mithilfe des erstellten Endpunkts zu Ihrer Timestream for InfluxDB-Instance, um sich anzumelden und auf die InfluxDB-Benutzeroberfläche zuzugreifen. Sie müssen den

Benutzernamen und das Passwort verwenden, mit denen Sie Ihre InfluxDB-DB-Instance erstellt haben. Sie können diese Informationen aus dem abrufen`influxAuthParametersSecretArn`, das im Antwortobjekt von angegeben wurde. `CreateDbInstance`

Alternativ können Sie die InfluxUI über die Timestream for InfluxDB-Managementkonsole öffnen:

a. Melden Sie sich bei der an AWS Management Console und öffnen Sie die Amazon Timestream for InfluxDB-Konsole unter. https://console.aws.amazon.com/timestream/
b. Wählen Sie in der oberen rechten Ecke der Amazon Timestream for InfluxDB-Konsole die AWS Region aus, in der Sie die DB-Instance erstellt haben.
c. Wählen Sie in der Liste Datenbanken den Namen Ihrer InfluxDB-Instance aus, um deren Details anzuzeigen. Wählen Sie in der oberen rechten Ecke Open Influx UI aus.

2. Sobald Sie in Ihrer InfluxUI angemeldet sind, navigieren Sie mit der linken Navigationsleiste zu Daten laden und dann zu APITokens.
3. Wählen Sie + GENERATE API TOKEN und wählen Sie All Access API Token aus.
4. Geben Sie eine Beschreibung für das API Token ein und wählen Sie SAVE.
5. Kopieren Sie das generierte Token und speichern Sie es zur sicheren Aufbewahrung.

> **Important**
>
> Beim Erstellen von Token aus der InfluxUI werden die neu erstellten Token nur einmal angezeigt. Stellen Sie sicher, dass Sie diese kopieren, da Sie sie andernfalls neu erstellen müssen.

Verwenden der InfluxDB API

- Senden Sie mit der Request-Methode eine Anfrage an den API `/api/v2/authorizations` InfluxDB-Endpunkt. POST

  Fügen Sie Ihrer Anfrage Folgendes bei:

  a. Überschriften:

     i. Autorisierung: Token < INFLUX _ OPERATOR _ TOKEN >
     ii. Inhaltstyp: application/json

  b. Hauptteil der Anfrage: JSON Körper mit den folgenden Eigenschaften:

i. Status: „aktiv“

ii. Beschreibung: Beschreibung des API Tokens

iii. OrgID: InfluxDB-Organisations-ID

iv. permissions: Array von Objekten, wobei jedes Objekt Berechtigungen für einen InfluxDB-Ressourcentyp oder eine bestimmte Ressource darstellt. Jede Berechtigung enthält die folgenden Eigenschaften:

A. Aktion: „lesen“ oder „schreiben“

B. Ressource: JSON Objekt, das die InfluxDB-Ressource darstellt, für die die Erlaubnis erteilt werden soll. Jede Ressource enthält mindestens die folgende Eigenschaft: OrgID: InfluxDB-Organisations-ID

C. Typ: Ressourcentyp. Für Informationen darüber, welche InfluxDB-Ressourcentypen existieren, verwenden Sie the /api/v2/resources endpoint.

Das folgende Beispiel verwendet `curl` und die InfluxDB, um ein API All-Access-Token zu generieren:

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

Datenbankauthentifizierungsoptionen

Amazon Timestream for InfluxDB unterstützt die folgenden Methoden zur Authentifizierung von Datenbankbenutzern:

- Passwortauthentifizierung – Ihre DB-Instance führt die gesamte Verwaltung von Benutzerkonten durch. Sie erstellen Benutzer, geben Passwörter an und verwalten Token mithilfe von InfluxUI, Influx oder Influx. CLI API

- Token-Authentifizierung — Ihre DB-Instance führt die gesamte Verwaltung von Benutzerkonten durch. Sie können Benutzer erstellen, ein Passwort angeben und Token mit Ihrem Operator-Token mithilfe von Influx und Influx CLI verwalten. API

Verschlüsselte Verbindungen

Sie können Secure Socket Layer (SSL) oder Transport Layer Security (TLS) in Ihrer Anwendung verwenden, um eine Verbindung zu einer DB-Instance zu verschlüsseln. Die Zertifikate, die für den TLS Handshake zwischen InfluxDB und den vom Kronos-Dienst erstellten und verwalteten Anwendungen benötigt werden. Wenn das Zertifikat erneuert wird, wird die Instanz automatisch mit der neuesten Version aktualisiert, ohne dass ein Benutzereingriff erforderlich ist.

Arbeiten mit Parametergruppen

Database parameters (Datenbankparameter) – geben Sie an, wie die Datenbank konfiguriert wird. Datenbankparameter können z. B. die Menge der Ressourcen angeben, die einer Datenbank zugewiesen werden sollen, wie etwa den Speicher.

Sie verwalten Ihre Datenbankkonfiguration, indem Sie Ihre DB-Instances Parametergruppen zuordnen. Amazon Timestream for InfluxDB definiert Parametergruppen mit Standardeinstellungen. Sie können auch eigene Parametergruppen mit angepassten Einstellungen definieren.

Übersicht über Parametergruppen

Eine DB-Parametergruppe dient als Container für Engine-Konfigurationswerte, die auf eine oder mehrere DB-Instances angewendet werden.

Themen

- Standard- und benutzerdefinierte Parametergruppen
- Erstellen einer DB-Parametergruppe
- Statische und dynamische DB-Instance-Parameter
- Unterstützte Parameter und Parameterwerte

Standard- und benutzerdefinierte Parametergruppen

DB-Instances verwenden DB-Parametergruppen. Die folgenden Abschnitte beschreiben das Konfigurieren und Verwalten von DB-Instance-Parametergruppen.

Erstellen einer DB-Parametergruppe

Sie können eine neue DB-Parametergruppe mit dem AWS Management Console, dem oder dem Timestream AWS Command Line Interface erstellen. API

Die folgenden Einschränkungen gelten für den Namen der DB-Parametergruppe:

- Der Name muss zwischen 1 und 255 Buchstaben, Zahlen oder Bindestriche enthalten.
- Standardnamen für Parametergruppen können einen Punkt enthalten, z. B. `default.InfluxDB.2.7`. Namen von benutzerdefinierten Parametergruppen dürfen jedoch keinen Punkt enthalten.
- Das erste Zeichen muss ein Buchstabe sein.
- Der Name darf nicht mit „dbpg-“ beginnen
- Der Name darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.
- Wenn Sie eine DB-Instance erstellen, ohne eine DB-Parametergruppe anzugeben, verwendet die DB-Instance die Standardeinstellungen der InfluxDB-Engine.

Sie können die Parametereinstellungen für eine Standard-Parametergruppe nicht ändern. Stattdessen können Sie Folgendes tun:

1. Neue Parametergruppe erstellen.
2. Ändern Sie die Einstellungen Ihrer gewünschten Parameter. In einer Parametergruppe können nicht alle DB-Engine-Parameter geändert werden.
3. Aktualisieren Sie Ihre DB-Instance, um die benutzerdefinierte Parametergruppe zu verwenden. Hinweise zum Aktualisieren einer DB-Instance finden Sie unterAktualisieren von DB-Instances.

> **Note**
>
> Wenn Sie Ihre DB-Instance so geändert haben, dass sie eine benutzerdefinierte Parametergruppe verwendet, und Sie die DB-Instance starten, startet Amazon Timestream for InfluxDB die DB-Instance im Rahmen des Startvorgangs automatisch neu.
> Derzeit können Sie benutzerdefinierte Parametergruppen nicht mehr ändern, nachdem sie erstellt wurden. Wenn Sie einen Parameter ändern müssen, müssen Sie eine neue benutzerdefinierte Parametergruppe erstellen und sie den Instanzen zuweisen, für die diese Konfigurationsänderung erforderlich ist. Wenn Sie eine bestehende DB-Instance

aktualisieren, um eine neue Parametergruppe zuzuweisen, wird diese immer sofort angewendet und Ihre Instance wird neu gestartet.

Statische und dynamische DB-Instance-Parameter

Die Parameter der InfluxDB-DB-Instance sind immer statisch. Sie verhalten sich wie folgt:

Wenn Sie einen statischen Parameter ändern, die DB-Parametergruppe speichern und sie einer Instance zuweisen, wird die Parameteränderung nach dem Neustart der Instance automatisch wirksam.

Wenn Sie einer DB-Instance eine neue DB-Parametergruppe zuordnen, wendet Timestream die geänderten statischen Parameter erst an, nachdem die DB-Instance neu gestartet wurde. Derzeit ist die einzige Option „Sofort anwenden“.

Weitere Informationen zum Ändern der DB-Parametergruppe finden Sie unter Aktualisieren von DB-Instances.

Unterstützte Parameter und Parameterwerte

Um die unterstützten Parameter für Ihre DB-Instance zu ermitteln, sehen Sie sich die Parameter in der DB-Parametergruppe an, die von der DB-Instance verwendet wird. Weitere Informationen finden Sie unter Parameterwerte für eine DB-Parametergruppe anzeigen.

Weitere Informationen zu allen Parametern, die von der Open-Source-Version von InfluxDB unterstützt werden, finden Sie unter InfluxDB-Konfigurationsoptionen. Derzeit können Sie nur die folgenden InfluxDB-Parameter ändern:

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
flux-log-enabled	Fügen Sie die Option hinzu, um detaillierte Protokolle für Flux-Abfragen anzuzeigen	FALSE	true, false	N/A	

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
auf Protokollebene	Ausgangspegel protokollieren. InfluxDB gibt Protokolleinträge mit Schweregraden aus, die größer oder gleich dem angegebenen Schweregrad sind.	info	Debug, Info, Fehler	N/A	
keine Aufgaben	Anzahl der Abfragen, die gleichzeitig ausgeführt werden dürfen. Die Einstellung auf 0 ermöglicht eine unbegrenzte Anzahl gleichzeitiger Abfragen.	FALSE	true, false	N/A	

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
Parallelität von Abfragen	Deaktivieren Sie den Taskplaner. Wenn problematische Aufgaben den Start von InfluxDB verhindern, verwenden Sie diese Option, um InfluxDB zu starten, ohne Aufgaben zu planen oder auszuführen.	1024		N/A	

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
query-queue-size	Maximale Anzahl von Abfragen, die in der Ausführungswartesc hlange zulässig sind. Wenn das Warteschlangenlimit erreicht ist, werden neue Abfragen zurückgewiesen. Die Einstellung auf 0 ermöglicht eine unbegrenzte Anzahl von Abfragen in der Warteschlange.	1024		N/A	

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
Ablaufverfolgungstyp	Aktiviert die Ablaufverfolgung in InfluxDB und gibt den Tracing-Typ an. Die Ablaufverfolgung ist standardmäßig deaktiviert.	""	log, Jaeger	N/A	
Metriken sind deaktiviert	Deaktivieren Sie den Endpunkt HTTP /metrics, der interne InfluxDB-Metriken verfügbar macht.	FALSE		N/A	

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
http-idle-timeout	Maximale Dauer, für die der Server bestehende Verbindungen aufrechterhalten sollte, während er auf neue Anfragen wartet. Auf „Kein Timeout“ setzen. `0`	30 ms	Dauer mit Einheit`hours`,,, `seconds` `milliseconds` Beispiel: `durationType=minutes,value=10`	Stunden: -Minimum: 0 -Maximal: 256205 Minuten: -Minimum: 0 -Maximal: 15372286 Sekunden: -Minimum: 0 -Maximal: 922337203 Millisekunden: -Minimum: 0 -Maximal: 9223372036685	

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
http-read-header-timeout	Maximale Dauer, für die der Server versuchen sollte, Header für neue Anfragen zu lesen. HTTP Auf „Ohne Zeitlimit“ einstellen. `0`	10s	Dauer mit Einheit`hours`,, `minutes``seconds` . Beispiel: `durationType=minutes,value=10`	Stunden: -Minimum: 0 -Maximal: 256205 Minuten: -Minimum: 0 -Maximal: 15372286 Sekunden: -Minimum: 0 -Maximal: 922337203 Millisekunden: -Minimum: 0 -Maximal: 922337203685	

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
http-read-timeout	Maximale Dauer, für die der Server versuchen sollte, alle neuen Anfragen zu lesen. Auf „Ohne Zeitlimit“ einstellen. `0`	0	Dauer mit Einheit `hours`,`minutes` . Beispiel: `durationType=minutes,value=10`	Stunden: -Minimum: 0 -Maximal: 256205 Minuten: -Minimum: 0 -Maximal: 15372286 Sekunden: -Minimum: 0 -Maximal: 922337203 Millisekunden: -Minimum: 0 -Maximal: 9223372036685	

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
http-write-timeout	Maximale Dauer, die der Server mit der Verarbeitung und Beantwortung von Schreibanforderungen verbringen sollte. Auf „Ohne Zeitlimit“ einstellen. `0`	0	Dauer mit Einheit `hours,minds` . Beispiel: `durationType=minutes,value=10`	Stunden: -Minimum: 0 -Maximal: 256205 Minuten: -Minimum: 0 -Maximal: 15372286 Sekunden: -Minimum: 0 -Maximal: 922337203 Millisekunden: -Minimum: 0 -Maximal: 922337203685	

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
influxql-max-select-buckets	Maximale Anzahl von gruppierten Zeitfenstern, die eine Anweisung erstellen kann. `SELECT 0` erlaubt eine unbegrenzte Anzahl von Buckets.	0	Long	Minimum: 0 Maximum: 9.223.372.036.854.775.807	

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
influxql-max-select-point	Maximale Anzahl von Punkten, die eine Anweisung verarbeiten kann. `SELECT 0` erlaubt eine unbegrenzte Anzahl von Punkten. InfluxDB überprüft die Punkteanzahl jede Sekunde (sodass Abfragen, die das Maximum überschreiten, nicht sofort abgebrochen werden).	0	Long	Minimum: 0 Maximum: 9.223.372.036.854.775.807	

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
influxql-max-select-series	Maximale Anzahl von Serien, die eine Anweisung zurückgeben kann. `SELECT 0` erlaubt eine unbegrenzte Anzahl von Serien.	0	Long	Minimum: 0 Maximum: 9.223.372.036.854.775.807	
pprof-deaktiviert	Deaktiviert den Endpunkt. `/debug/pprof` HTTP Dieser Endpunkt stellt Runtime-Profilerstellungsdaten bereit und kann beim Debuggen hilfreich sein.	FALSE	Boolesch	N/A	

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
query-initial-memory-bytes	Anfängliche Speicherbytes, die einer Abfrage zugewiesen wurden.	0	Long	Minimum: 0 Maximal: query-memory-bytes	
query-max-memory-bytes	Maximal zulässige Gesamtanzahl an Speicherbytes für Abfragen.	0	Long	Minimum: 0 Maximum: 9.223.372.036.854.775.807	
query-memory-bytes	Gibt die Gültigkeitsdauer () in Minuten für neu erstellte Benutzersitzungen an. TTL	0	Long	Minimum: 0 Maximum: 2.147.483.647	Muss größer oder gleich sein. query-initial-memory-bytes
Länge der Sitzung	Gibt die Gültigkeitsdauer (TTL) in Minuten für neu erstellte Benutzersitzungen an.	60	Ganzzahl	Minimum: 0 Maximum: 2880	

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
session-renew-disabled	Deaktiviert die automatische Verlängerung der Sitzung eines Benutzers TTL bei jeder Anfrage. Standardmäßig legt jede Anfrage die Ablaufzeit der Sitzung auf fünf Minuten fest. Wenn diese Option deaktiviert ist, laufen Sitzungen nach der angegebenen Sitzungsdauer ab und der Benutzer wird auf die Anmeldeseite umgeleitet, auch wenn er kürzlich aktiv war.	FALSE	Boolesch	N/A	

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
storage-cache-max-memory-Größe	Die maximale Größe (in Byte), die der Cache eines Shards erreichen kann, bevor er anfängt, Schreibvorgänge zurückzuweisen.	1073741824	Long	Minimum: 0 Maximum: 549755813888	Muss niedriger als die Gesamtspeicherkapazität der Instanz sein. Wir empfehlen, den Wert auf unter 15% der Gesamtspeicherkapazität einzustellen.
storage-cache-snapshot-memory-Größe	Größe (in Byte), bei der die Speicher-Engine einen Snapshot des Caches erstellt und ihn in eine TSM Datei schreibt, um mehr Speicher verfügbar zu machen.	26214400	Long	Minimum: 0 Maximal: 549755813888	Muss kleiner als -size sein. storage-cache-max-memory

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
storage-cache-snapshot-write-Kältedauer	Dauer, in der die Speicher-Engine einen Snapshot des Caches erstellt und ihn in eine neue TSM Datei schreibt, falls der Shard keine Schreib- oder Löschvorgänge empfangen hat.	100 ms	Dauer mit Einheithours,,, seconds milliseconds Beispiel: durationType=minutes,value=10	Stunden: -Minimum: 0 -Maximal: 256205 Minuten: -Minimum: 0 -Maximal: 15372286 Sekunden: -Minimum: 0 -Maximal: 922337203 Millisekunden: -Minimum: 0 -Maximal: 9223372036685	

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
storage-compact-full-write-Kältedauer	Dauer, nach der die Speicher-Engine alle TSM Dateien in einem Shard komprimiert, wenn sie keine Schreib- oder Löschvorgänge empfangen hat.	40 Stunden	Dauer mit Einheit,, ,. `hours` `minutes` `seconds` `milliseconds` Beispiel: `durationType=minutes,value=10`	Stunden: -Minimum: 0 -Maximal: 256205 Minuten: -Minimum: 0 -Maximal: 15372286 Sekunden: -Minimum: 0 -Maximal: 922337203 Millisekunden: -Minimum: 0 -Maximal: 9223372036685	

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
storage-compact-throughput-burst	Ratenlimit (in Byte pro Sekunde), das bei Komprimierungen auf die TSM Festplatte geschrieben werden kann.	50331648	Long	Minimum: 0 Maximum: 9.223.372.036.854.775.807	

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
storage-max-concurrent-compactions	Maximale Anzahl von Vollkompr imierunge n und Level-Ver dichtungen, die gleichzei tig ausgeführ t werden können. Ein Wert von `0` ergibt, dass 50% von `runtime.G OMAXPROCS (0)` zur Laufzeit verwendet werden. Jede Zahl, die größer als Null ist, begrenzt die Komprimie rung auf diesen Wert. Diese Einstellung gilt nicht für Cache-Sna pshots.	0	Ganzzahl	Minimum: 0 Maximum: 64	

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
storage-max-index-log-Dateigröße	Größe (in Byte), bei der eine Index-Write-Ahead-Logdatei (WAL) zu einer Indexdatei komprimiert wird. Niedrigere Größen führen dazu, dass Protokolldateien schneller komprimiert werden und führen zu einer geringeren Heap-Nutzung auf Kosten des Schreibdurchsatzes.	1048576	Long	Minimum: 0 Maximum: 9.223.372.036.854.775.807	

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
storage-no-validate-field-Größe	Überspringt die Überprüfung der Feldgröße bei eingehenden Schreibanforderungen.	FALSE	Boolesch	N/A	

storage-retention-check-interval	Intervall, in dem die Einhaltung der Aufbewahrungsrichtlinien überprüft wird.	300 ms	Dauer mit Einheit
			`hours`,,,
			`seconds`
			`milliseconds`
			Beispiel:
			`durationType=minutes,value=10`
				N/A	Stunden:
					-Minimum: 0
					-Maximal: 256205
					Minuten:
					-Minimum: 0
					-Maximal: 15372286
					Sekunden:
					-Minimum: 0
					-Maximal: 922337203
					Millisekunden:
					-Minimum: 0
					-Maximal: 922337203685

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
storage-series-file-max-concurrent-snapshot-compactions	Maximale Anzahl von Snapshot-Komprimierungen, die gleichzeitig auf allen Serienpartitionen in einer Datenbank ausgeführt werden können.	0	Ganzzahl	Minimum: 0 Maximum: 64	

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
storage-series-id-set-Cache-Größe	Größe des internen Caches, der im TSI Index verwendet wird, um zuvor berechnete Reihenergebnisse zu speichern. Zwischengespeicherte Ergebnisse werden schnell zurückgegeben und müssen nicht neu berechnet werden, wenn eine nachfolgende Abfrage mit demselben Tag-Schlüssel-/Wertprädikat ausgeführt wird. Wenn Sie diesen Wert auf	100	Long	Minimum: 0 Maximum: 9.223.372.036.854.775.807	

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
	festlegen , wird der Cache deaktiviert und die Abfragele istung kann beeinträchtigt werden.	0			
storage-wal-max-concurrent-schreibt	Maximale Anzahl von Schreibversuchen in das WAL Verzeichnis, die gleichzeitig versucht werden sollen.	0	Ganzzahl	Minimum: 0 Maximal: 256	

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
storage-wal-max-write-Verzögerung	Höchstdauer, für die eine Schreibanforderung in das WAL Verzeichnis wartet, bis die maximale Anzahl gleichzeitiger aktiver Schreibvorgänge in das WAL Verzeichnis erreicht ist. Auf einstellen, `0` um das Timeout zu deaktivieren.	10m	Dauer mit Einheit`hours`,, `minutesseconds` . Beispiel: `durationType=minutes,value=10`	Stunden: -Minimum: 0 -Maximal: 256205 Minuten: -Minimum: 0 -Maximal: 15372286 Sekunden: -Minimum: 0 -Maximal: 922337203 Millisekunden: -Minimum: 0 -Maximal: 9223372036685	

Parameter	Beschreibung	Standardwert	Wert	Gültiger Bereich	Hinweis
UI-deaktiviert	Deaktivieren Sie die InfluxDB-Benutzeroberfläche (UI). Die Benutzeroberfläche ist standardmäßig aktiviert.	FALSE	Boolesch	N/A	

Werden die Parameter in einer Parametergruppe unpassend eingestellt, kann dies unbeabsichtigte unerwünschte Auswirkungen haben, einschließlich verminderter Leistung und Systeminstabilität. Seien Sie immer vorsichtig, wenn Sie Datenbankparameter ändern. Versuchen Sie, die Parametergruppeneinstellungen in einer Test-DB-Instance zu ändern, bevor Sie diese Änderungen an den Parametergruppen auf eine Produktions-DB-Instance anwenden.

Arbeiten mit DB-Parametergruppen

DB-Instances verwenden DB-Parametergruppen. Die folgenden Abschnitte beschreiben das Konfigurieren und Verwalten von DB-Instance-Parametergruppen.

Themen

- Erstellen einer DB-Parametergruppe
- Verknüpfen einer DB-Parametergruppe mit einer DB-Instance
- Auflisten von DB-Parametergruppen
- Anzeigen von Parameterwerten für eine DB-Parametergruppe

Erstellen einer DB-Parametergruppe

Verwenden Sie den AWS Management Console

1. Melden Sie sich bei der Amazon Timestream for InfluxDB-Konsole an AWS Management Console und öffnen Sie sie.

2. Wählen Sie im Navigationsbereich Parameter groups (Parametergruppen) aus.
3. Wählen Sie Create parameter group (Parametergruppe erstellen).
4. Geben Sie im Feld Group name (Gruppenname) den Namen der neuen DB-Parametergruppe ein.
5. Geben Sie im Feld Description (Beschreibung) eine Beschreibung für die neue DB-Parametergruppe ein.
6. Wählen Sie die Parameter aus, die geändert werden sollen, und wenden Sie die gewünschten Werte an. Weitere Informationen zu unterstützten Parametern finden Sie unterUnterstützte Parameter und Parameterwerte.
7. Wählen Sie Save (Speichern) aus.

Unter Verwendung der AWS Command Line Interface

- Um eine DB-Parametergruppe mit dem zu erstellen AWS CLI, rufen Sie den `create-db-parameter-group` Befehl mit den folgenden Parametern auf:

```
--db-parameter-group-name <value>
--description <value>
--endpoint_url <value>
--region <value>
--parameters (list) (string)
```

Example Beispiel

Weitere Informationen zu den einzelnen Einstellungen finden Sie unter Einstellungen für DB-Instances. In diesem Beispiel werden Standard-Engine-Konfigurationen verwendet.

```
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
```

Verknüpfen einer DB-Parametergruppe mit einer DB-Instance

Sie können Ihre eigenen DB-Parametergruppen mit benutzerdefinierten Einstellungen erstellen. Sie können eine DB-Parametergruppe mit einer DB-Instance verknüpfen, indem Sie die AWS Management Console AWS Command Line Interface, oder die API TimeStream-InfluxDB verwenden. Das können Sie tun, wenn Sie eine DB-Instance erstellen oder ändern.

Informationen über das Erstellen einer Parametergruppe finden Sie unter Erstellen einer DB-Parametergruppe. Weitere Informationen zum Erstellen einer DB-Instance finden Sie unter Erstellen einer DB-Instance. Hinweise zum Ändern einer DB-Instance finden Sie unter.. Aktualisieren von DB-Instances

> **Note**
>
> Wenn Sie einer DB-Instance eine neue DB-Parametergruppe zuordnen, werden die geänderten statischen Parameter erst nach dem Neustart der DB-Instance angewendet. Derzeit wird nur „Sofort anwenden“ unterstützt. Timestream für InfluxDB unterstützt nur statische Parameter.

Mit dem AWS Management Console

1. Melden Sie sich bei der Amazon Timestream for InfluxDB-Konsole an AWS Management Console und öffnen Sie sie.
2. Wählen Sie im Navigationsbereich InfluxDB-Datenbanken und dann die DB-Instance aus, die Sie ändern möchten.
3. Wählen Sie Aktualisieren. Die Seite DB-Instance aktualisieren wird angezeigt.
4. Ändern Sie die Einstellung für DB-Parametergruppen .
5. Klicken Sie auf Weiter und überprüfen Sie die Zusammenfassung aller Änderungen.
6. Derzeit wird nur „Sofort anwenden“ unterstützt. Diese Option kann in einigen Fällen zu einem Ausfall führen, da dadurch Ihre DB-Instance neu gestartet wird.
7. Überprüfen Sie auf der Bestätigungsseite Ihre Änderungen. Wenn sie korrekt sind, wählen Sie DB-Instance aktualisieren, um Ihre Änderungen zu speichern und anzuwenden. Oder klicken Sie auf Zurück, um Ihre Änderungen zu bearbeiten, oder auf Abbrechen, um Ihre Änderungen zu verwerfen.

Verwenden Sie den AWS Command Line Interface

Für Linux, macOS oder 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 }}"
```

Für 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 }}"
```

Auflisten von DB-Parametergruppen

Sie können die DB-Parametergruppen auflisten, die Sie für Ihr AWS Konto erstellt haben.

Mit dem AWS Management Console

1. Melden Sie sich bei der Amazon Timestream for InfluxDB-Konsole an AWS Management Console und öffnen Sie sie.
2. Wählen Sie im Navigationsbereich Parameter groups (Parametergruppen) aus.
3. Die verfügbaren DB-Parametergruppen erscheinen in einer Liste.

Verwenden Sie die AWS Command Line Interface

Um alle DB-Parametergruppen für ein AWS Konto aufzulisten, verwenden Sie den AWS Command Line Interface `list-db-parameter-groups` Befehl.

```
aws timestream-influxdb list-db-parameter-groups --region region
```

Verwenden Sie den AWS Command Line Interface `get-db-parameter-group` Befehl, um bestimmte DB-Parametergruppen für ein AWS Konto zurückzugeben.

```
aws timestream-influxdb get-db-parameter-group --region region --identifier identifier
```

Anzeigen von Parameterwerten für eine DB-Parametergruppe

Sie können eine Liste aller Parameter in einer DB-Parametergruppe und ihren Werten erhalten.

Verwenden Sie den AWS Management Console

1. Melden Sie sich bei der Amazon Timestream for InfluxDB-Konsole an AWS Management Console und öffnen Sie sie.
2. Wählen Sie im Navigationsbereich Parameter groups (Parametergruppen) aus.
3. Die verfügbaren DB-Parametergruppen erscheinen in einer Liste.
4. Wählen Sie den Namen der Parametergruppe, um deren Parameterliste anzuzeigen.

Verwenden Sie die AWS Command Line Interface

Um die Parameterwerte für eine DB-Parametergruppe anzuzeigen, verwenden Sie den AWS Command Line Interface `get-db-parameters` Befehl mit dem folgenden erforderlichen Parameter.

```
--db-parameter-group-name
```

Verwenden Sie den API

Um die Parameterwerte für eine DB-Parametergruppe anzuzeigen, verwenden Sie den API `GetDBParameters` Timestream-Befehl mit dem folgenden erforderlichen Parameter.

```
DBParameterGroupName
```

DB-Instances verwalten

In diesem Abschnitt werden verschiedene Aspekte der Verwaltung von Timestream für die InfluxDB-Instance behandelt, um optimale Leistung, Verfügbarkeit und Überwachungsfunktionen sicherzustellen. Er enthält Anleitungen zur Aktualisierung der Konfigurationen Ihrer Datenbank-Instances, zum Umgang mit Multi-AZ-Bereitstellungen und zu Failover-Prozessen. Außerdem wird erklärt, wie Sie Datenbankinstanzen löschen und die Protokollansicht für Ihre InfluxDB-Instances einrichten.

Themen

Aktualisieren von DB-Instances

Sie können die folgenden Konfigurationsparameter Ihrer Timestream for InfluxDB-Instance aktualisieren:

- Instance-Klasse
- Deployment type (Bereitstellungstyp)
- Parametergruppe
- Konfiguration der Protokollzustellung

> **Important**
>
> Wir empfehlen Ihnen, alle Änderungen auf einer Testinstanz zu testen, bevor Sie die Produktionsinstanz ändern, um ihre Auswirkungen zu verstehen, insbesondere beim Upgrade von Datenbankversionen. Überprüfen Sie die Auswirkungen auf Ihre Datenbank und Anwendungen, bevor Sie die Einstellungen aktualisieren. Einige Änderungen erfordern einen Neustart der DB-Instance, was zu Ausfallzeiten führt.

Mit dem AWS Management Console

1. Melden Sie sich bei der Amazon Timestream for InfluxDB-Konsole an AWS Management Console und öffnen Sie sie.
2. Wählen Sie im Navigationsbereich InfluxDB-Datenbanken und dann die DB-Instance aus, die Sie ändern möchten.
3. Wählen Sie Ändern aus.
4. Nehmen Sie auf der Seite DB-Instance modifizieren die gewünschten Änderungen vor.
5. Klicken Sie auf Weiter und überprüfen Sie die Zusammenfassung aller Änderungen.

6. Wählen Sie Weiter.
7. Überprüfen Sie Ihre Änderungen.
8. Wählen Sie Instanz ändern, um Ihre Änderungen zu übernehmen.

> **Note**
>
> Diese Änderungen erfordern einen Neustart der Influx DB-Instance und können in einigen Fällen zu einem Ausfall führen.

Mit dem AWS Command Line Interface

Um eine DB-Instance mit dem zu aktualisieren AWS Command Line Interface, rufen Sie den `update-db-instance` Befehl auf. Geben Sie die DB-Instance-Kennung und die Werte für die Optionen an, die geändert werden sollen. Informationen zu den jeweiligen Optionen finden Sie unter Einstellungen für DB-Instances.

Example Beispiel

Der folgende Code ändert mydbinstance, indem er eine andere dbparametergroup festlegt. Die Änderungen werden sofort übernommen.

Für Linux, macOS oder 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
```

Für 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
```

Warten einer DB-Instance

In regelmäßigen Abständen führt Amazon Timestream for InfluxDB Wartungsarbeiten an Amazon Timestream for InfluxDB-Ressourcen durch. Die Wartung beinhaltet meistens Aktualisierungen der folgenden Ressourcen in Ihrer DB-Instance:

- Zugrundeliegende Hardware
- Zugrundeliegendes Betriebssystem
- Datenbank-Engine-Version

Häufig werden Betriebssystemupdates wegen Sicherheitsproblemen herausgegeben.

Bei einigen Wartungsarbeiten muss Amazon Timestream for InfluxDB Ihre DB-Instance für kurze Zeit offline schalten. Zu den Wartungselementen, für die eine Ressource offline sein muss, gehört z. B. das Ausführen erforderlicher Patches für das Betriebssystem oder die Datenbank. Das erforderliche Patching wird automatisch und nur für Patches eingeplant, welche die Sicherheit und Instance-Zuverlässigkeit betreffen. Solche Patches treten selten auf, in der Regel einmal alle paar Monate. Es ist selten mehr als ein Bruchteil Ihres Wartungsfensters dafür erforderlich.

- Die Wartungsfenster sind so konfiguriert, dass sie täglich zwischen 12 Uhr und 4 Uhr Ortszeit für die Region, in der Ihre Instance gehostet wird, stattfinden.
- Kundenressourcen können einmal pro Woche in einem der sieben Wartungsfenster der Woche gepatcht werden.

Löschen einer DB-Instance

Das Löschen einer DB-Instance hat Auswirkungen auf die Wiederherstellbarkeit der Instance und die Verfügbarkeit von Snapshots. Betrachten Sie die folgenden Punkte:

- Wenn Sie alle Timestream-Ressourcen für InfluxDB löschen möchten, beachten Sie, dass für die Ressourcen der DB-Instances Abrechnungsgebühren anfallen.
- Wenn der Status einer DB-Instance gelöscht ist, erscheint ihr CA-Zertifikatswert nicht in der Timestream for InfluxDB-Konsole oder in der Ausgabe für Befehle oder Timestream-Operationen. AWS Command Line Interface API

- Die zum Löschen einer DB-Instance benötigte Zeit hängt davon ab, wie viele Daten gelöscht werden und ob ein letzter Snapshot erstellt wird.

Sie können eine DB-Instance mit dem AWS Management Console, dem oder dem AWS Command Line Interface Timestream API löschen. Sie müssen den Namen der DB-Instance angeben:

Mit dem AWS Management Console

1. Melden Sie sich bei der Amazon Timestream for InfluxDB-Konsole an AWS Management Console und öffnen Sie sie.
2. Wählen Sie im Navigationsbereich InfluxDB-Datenbanken und dann die DB-Instance aus, die Sie löschen möchten.
3. Wählen Sie Löschen.
4. Geben Sie in das Feld Bestätigen ein.
5. Wählen Sie Löschen.

Mit dem AWS Command Line Interface

Rufen Sie den `list-db-instances` folgenden Befehl auf, um die Instanz IDs der DB-Instances in Ihrem Konto zu finden:

```
aws timestream-influxdb list-db-instances \
--endpoint-url YOUR_ENDPOINT \
--region YOUR_REGION
```

Um eine DB-Instance mithilfe von zu löschen AWSCLI, rufen Sie den `delete-db-instance` Befehl mit den folgenden Optionen auf:

```
aws timestream-influxdb list-db-instances \
--identifier YOUR_DB_INSTANCE \
```

Example Beispiel

Für Linux, macOS oder Unix:

```
aws timestream-influxdb delete-db-instance \
    --identifier mydbinstance
```

Für Windows:

```
aws timestream-influxdb delete-db-instance ^
    --identifier mydbinstance
```

Multi-AZ-DB-Instance-Bereitstellungen

Amazon Timestream for InfluxDB bietet Hochverfügbarkeit und Failover-Unterstützung für DB-Instances, die Multi-AZ-Bereitstellungen mit einer einzigen Standby-DB-Instance verwenden. Diese Art der Bereitstellung wird als Multi-AZ-DB-Instance-Bereitstellung bezeichnet. Amazon Timestream for InfluxDB verwendet die Amazon-Failover-Technologie.

In einer Multi-AZ-DB-Instance-Bereitstellung stellt Amazon Timestream automatisch ein synchrones Standby-Replikat in einer anderen Availability Zone bereit und verwaltet es. Die primäre DB-Instance wird über die Availability Zone synchron auf ein Standby-Replikat repliziert, um Datenredundanz zu erzielen. Der Betrieb einer DB-Instance mit hoher Verfügbarkeit kann die Verfügbarkeit bei einem Ausfall der DB-Instance und bei einer Unterbrechung der Availability Zone verbessern. Weitere Informationen über Availability Zones finden Sie unter AWS Regionen und Verfügbarkeitszonen.

> **Note**
>
> Die Option für hohe Verfügbarkeit ist keine Skalierungslösung für schreibgeschützte Szenarien. Sie können kein Standby-Replikat verwenden, um Leseverkehr bereitzustellen.

Mithilfe der Amazon Timestream Timestream-Konsole können Sie eine Multi-AZ-DB-Instance-Bereitstellung erstellen, indem Sie bei der Erstellung einer DB-Instance einfach die Option Standby-Instance erstellen im Abschnitt Konfiguration für Verfügbarkeit und Haltbarkeit angeben. Sie können auch eine Multi-AZ-DB-Instance-Bereitstellung mit dem AWS Command Line Interface oder Amazon API Timestream angeben. Verwenden Sie den CLI Befehl `create-db-instance` oder oder die `CreateDBInstance` API Operation.

DB-Instances, die Multi-AZ-DB-Instance-Bereitstellungen verwenden, können im Vergleich zu einer Single-AZ-Bereitstellung eine höhere Schreib- und Commit-Latenz aufweisen. Dies kann aufgrund der auftretenden synchronen Datenreplikation geschehen. Es kann zu einer Änderung der Latenz kommen, wenn bei Ihrer Bereitstellung ein Failover auf das Standby-Replikat erfolgt, obwohl es AWS für Netzwerkverbindungen mit niedriger Latenz zwischen Availability Zones konzipiert wurde. Für Produktionsworkloads empfehlen wir, für eine schnelle, konsistente Leistung die Verwendung

IOPS von 12 oder 16 KB enthaltenem Speicher IOPS zu verwenden. Weitere Informationen zu DB-Instance-Klassen finden Sie unter DB-Instance-Klassen.

Konfiguration und Verwaltung einer Multi-AZ-Bereitstellung

Timestream für InfluxDB Multi-AZ-Bereitstellungen kann nur einen Standby-Modus haben. Wenn die Bereitstellung über eine Standby-DB-Instance verfügt, wird sie als Multi-AZ-DB-Instance-Bereitstellung bezeichnet. Eine Multi-AZ DB-Instance-Bereitstellung verfügt über eine Standby-DB-Instance, die Failover-Unterstützung bietet, aber keinen Lesedatenverkehr bereitstellt.

> **Important**
>
> Ihrer Instance müssen mindestens zwei Subnetze zugeordnet sein, um Single-AZ- bis Multi-AZ-Updates ausführen zu können. Sobald die Instance erstellt wurde, können Sie ihren Bereitstellungsmodus nicht mehr von Single-AZ auf Multi-AZ ändern.

Sie können den verwenden AWS Management Console , um festzustellen, ob es sich bei Ihrer DB-Instance um eine Single-AZ- oder Multi-AZ-Bereitstellung handelt.

Mit dem AWS Management Console

1. Melden Sie sich bei der Amazon Timestream for InfluxDB-Konsole an AWS Management Console und öffnen Sie sie.
2. Wählen Sie im Navigationsbereich InfluxDB-Datenbanken und dann DB-Identifier aus.

Eine Multi-AZ-DB-Instance-Bereitstellung hat folgende Eigenschaften:

- Es gibt nur eine Zeile für die DB-Instance.
- Der Wert von Rolle ist Instance oder Primär.
- Der Wert von Multi-AZ ist Yes (Ja).

Failover-Prozess für Amazon Timestream

Wenn ein geplanter oder ungeplanter Ausfall Ihrer DB-Instance auf einen Infrastrukturdefekt zurückzuführen ist, wechselt Amazon Timestream for InfluxDB automatisch zu einem Standby-Replikat in einer anderen Availability Zone, wenn Sie Multi-AZ aktiviert haben. Die Dauer, bis der

Failover-Prozess für die Instance abgeschlossen ist, hängt von der Datenbankaktivität sowie von anderen Bedingungen zu dem Zeitpunkt ab, an dem die primäre DB-Instance ausgefallen ist. Der Failover-Prozess dauert normalerweise 60–120 Sekunden. Diese Failover-Dauer kann sich verlängern, wenn umfangreiche Transaktionen oder zeitintensive Wiederherstellungsprozesse durchgeführt werden. Wenn der Failover abgeschlossen ist, kann es zusätzliche Zeit dauern, bis die Timestream-Konsole die neue Availability Zone wiedergibt.

> **Note**
>
> Amazon Timestream verarbeitet Failovers automatisch, sodass Sie den Datenbankbetrieb ohne administrativen Eingriff so schnell wie möglich wieder aufnehmen können. Die primäre DB-Instance schaltet automatisch auf das Standby-Replikat um, wenn eine der in der folgenden Tabelle beschriebenen Bedingungen eintritt:

Failover-Grund	Beschreibung
Das Betriebssystem, das der Timestream-Datenbank-Instance zugrunde liegt, wird in einem Offline-Vorgang gepatcht.	Ein Failover wurde während des Wartungsfensters für einen Betriebssystem-Patch oder ein Sicherheitsupdate ausgelöst.
Der primäre Host der Timestream Multi-AZ-Instance ist fehlerhaft.	Die Multi-AZ-DB-Instance-Bereitstellung hat eine beeinträchtigte primäre DB-Instance erkannt und ein Failover eingeleitet.
Der primäre Host der Timestream Multi-AZ-Instance ist aufgrund eines Verlusts der Netzwerkkonnektivität nicht erreichbar.	Die Timestream-Überwachung hat einen Fehler bei der Netzwerkerreichbarkeit der primären DB-Instance erkannt und einen Failover ausgelöst.
Die Timestream-Instance wurde vom Kunden geändert.	Eine Änderung der Timesteam for InfluxDB-DB-Instance löste einen Failover aus. Weitere Informationen finden Sie unter Aktualisieren von DB-Instances.
Die primäre Timestream Multi-AZ-Instance ist ausgelastet und reagiert nicht.	Die primäre DB-Instance reagiert nicht. Wir empfehlen Ihnen, wie folgt vorzugehen: * Untersuchen Sie das Ereignis auf übermäßig

Failover-Grund	Beschreibung
	e CPU Speicher- oder Auslagerungsspeich erbelegung. * Bewerten Sie Ihre Arbeitslast, um festzustellen, ob Sie die entsprechende DB-Instance-Klasse verwenden. Weitere Informati onen finden Sie unter DB-Instance-Klassen.
Auf dem Speichervolumen, das dem primären Host der Timestream Multi-AZ-Instance zugrunde liegt, ist ein Fehler aufgetreten.	Die Multi-AZ-DB-Instance-Bereitstellung hat ein Speicherproblem der primären DB-Instance erkannt und ein Failover eingeleitet.

Einstellung von JVM TTL für Namenssuchen DNS

Der Failover-Mechanismus ändert den Domain Name System (DNS) -Datensatz der DB-Instance automatisch so, dass er auf die Standby-DB-Instance verweist. Als Ergebnis müssen alle bestehenden Verbindungen zu Ihrer DB-Instance neu hergestellt werden. In einer Java-Virtual-Machine-Umgebung (JVM) müssen Sie aufgrund der Funktionsweise des DNS Java-Caching-Mechanismus möglicherweise die Einstellungen neu JVM konfigurieren.

Die Namen der JVM Caches suchen DNS nach Namen. Wenn der einen Hostnamen in eine IP-Adresse JVM auflöst, speichert er die IP-Adresse für einen bestimmten Zeitraum im Cache, der als () bezeichnet wird. time-to-liveTTL

Da AWS Ressourcen DNS Namenseinträge verwenden, die sich gelegentlich ändern, empfehlen wir, dass Sie Ihren JVM mit einem TTL Wert von nicht mehr als 60 Sekunden konfigurieren. Dadurch wird sichergestellt, dass Ihre Anwendung, wenn sich die IP-Adresse einer Ressource ändert, die neue IP-Adresse der Ressource empfangen und verwenden kann, indem sie die erneut abfragt. DNS

Bei einigen Java-Konfigurationen TTL ist die JVM Standardeinstellung so eingestellt, dass DNS Einträge erst aktualisiert werden, wenn der JVM neu gestartet wird. Wenn sich also die IP-Adresse für eine AWS Ressource ändert, während Ihre Anwendung noch läuft, kann sie diese Ressource erst verwenden, wenn Sie die manuell neu starten JVM und die zwischengespeicherten IP-Informationen aktualisiert werden. In diesem Fall ist es wichtig, die IDs TTL so einzustellen, JVM dass die zwischengespeicherten IP-Informationen regelmäßig aktualisiert werden.

Sie können die JVM Standardeinstellung abrufen, TTL indem Sie den Eigenschaftswert abrufen: `networkaddress.cache.ttl`

```
String ttl = java.security.Security.getProperty("networkaddress.cache.ttl");
```

> **Note**
>
> Die Standardeinstellung TTL kann je nach Ihrer Version JVM und je nachdem, ob ein Sicherheitsmanager installiert ist, variieren. Viele JVMs bieten eine Standardeinstellung von TTL weniger als 60 Sekunden. Wenn Sie einen solchen JVM und keinen Sicherheitsmanager verwenden, können Sie den Rest dieses Themas ignorieren.
> Um die JVM s zu ändernTTL, legen Sie den Eigenschaftswert networkaddress.cache.ttl fest. Nutzen Sie dazu eine der folgenden Methoden je nach Ihrem Bedarf:
>
> - Um den Eigenschaftswert global für alle Anwendungen festzulegen, die den in der JVM Datei festgelegten Wert verwenden. `networkaddress.cache.ttl` `$JAVA_HOME/jre/lib/security/java.security`
>
>   ```
>   networkaddress.cache.ttl=60
>   ```
>
> - Um die Eigenschaft nur für Ihre Anwendung lokal festzulegen, legen Sie `networkaddress.cache.ttl` im Initialisierungscode Ihrer Anwendung fest, bevor Netzwerkverbindungen hergestellt werden.
>
>   ```
>   java.security.Security.setProperty("networkaddress.cache.ttl" , "60");
>   ```

Einrichtung zum Anzeigen von InfluxDB-Protokollen auf Timestream-InfluxDB-Instances

Standardmäßig generiert InfluxDB Protokolle, die nach stdout gehen. Weitere Informationen finden Sie unter InfluxDB-Logs verwalten

Um InfluxDB-Protokolle anzuzeigen, die von der Instance generiert wurden, die Sie über Timestream InfluxDB erstellt haben, bieten wir die Möglichkeit, stündliche Protokolle bereitzustellen. Diese Protokolle werden in einen bestimmten S3-Bucket verschoben, den Sie erstellen müssen, bevor Sie Ihre Instance erstellen.

- Vor dem Erstellen der Instance muss der bereitgestellte Amazon S3 S3-Bucket Timestream-InfluxDB auch die Erlaubnis erteilen, Logs an diesen Bucket zu senden, indem eine Bucket-

Richtlinie mit Timestream InfluxDB Service Principal wie folgt bereitgestellt wird (ersetzen *{BUCKET_NAME}* mit dem tatsächlichen Namen Ihres Amazon S3 S3-Buckets:

```
{
    "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/*"
        }
    ]
}
```

- Der angegebene Bucket muss sich im selben Konto und in derselben Region wie Ihre erstellte Timestream InfluxDB-Instance befinden

  Hier ist der Befehl, den Sie aufrufen können, um eine Instanz für den Empfang von Influx-Logs zu erstellen:

```
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
```

  Hier ist das Format dieses Parameters.

```
-- log-delivery-configuration
{
    "S3Configuration": {
      "BucketName": "string",
      "Enabled": true|false
    }
```

```
}
```

- Dieses Feld ist nicht erforderlich und die Protokollierung ist standardmäßig nicht aktiviert.
- Dieses Feld nicht zu setzen bedeutet, dass Protokolle nicht aktiviert sind.
- Protokolle werden an den angegebenen Bucket mit dem Präfix gesendet`InfluxLogs/`.
- Nachdem Sie die Instanz erstellt haben, können Sie die Konfiguration der Protokollzustellung mit dem `update-db-instance` API Befehl ändern.

InfluxDB bietet verschiedene Arten von Protokollen an. Diese können durch das Einstellen der InfluxDB-Parameter konfiguriert werden. Verwenden Sie die Parameter flux-log-enabled und die Parameter auf Protokollebene, um den Typ der Protokolle zu konfigurieren, die von der Instanz ausgegeben werden. Weitere Informationen finden Sie unter Unterstützte Parameter und Parameterwerte.

Hinzufügen von Tags und Etiketten zu Ressourcen

Sie können Amazon Timestream für InfluxDB-Ressourcen mithilfe von Tags kennzeichnen. Mithilfe von Tags können Sie Ihre Ressourcen auf unterschiedliche Weise kategorisieren, z. B. nach Zweck, Eigentümer, Umgebung oder anderen Kriterien. Tags können Sie bei Folgendem unterstützen:

- Eine Ressource basierend auf den Tags, die Sie ihr zugeordnet haben, schnell zu erkennen.
- Sehen Sie sich AWS Rechnungen an, die nach Schlagwörtern aufgeschlüsselt sind.

Tagging wird von AWS Diensten wie Amazon Elastic Compute Cloud (AmazonEC2), Amazon Simple Storage Service (Amazon S3), Timestream for InfluxDB und mehr unterstützt. Effizientes Tagging kann Kosteneinblicke bereistellen, mit denen Sie Berichte für Services erstellen können, die ein spezifisches Tag aufweisen.

Schließlich wird empfohlen, optimale Tagging-Strategien zu befolgen. Weitere Informationen finden Sie unter AWS-Markierungsstrategien.

Tagging-Einschränkungen

Jedes Tag besteht aus einem Schlüssel und einem Wert, die Sie beide selbst definieren können. Beachten Sie die folgenden Einschränkungen:

- Jede Timestream for InfluxDB-DB-Instance kann nur ein Tag mit demselben Schlüssel haben. Wenn Sie versuchen, ein vorhandenes Tag hinzuzufügen, wird der vorhandene Tag-Wert auf den neuen Wert aktualisiert.
- Ein Wert fungiert als Deskriptor innerhalb einer Tag-Kategorie. In Timestream for InfluxDB kann der Wert nicht leer oder null sein.
- Bei Tag-Schlüsseln und -Werten muss die Groß- und Kleinschreibung beachtet werden.
- Die maximale Schlüssellänge beträgt 128 Unicode-Zeichen.
- Die maximale Wertlänge beträgt 256 Unicode-Zeichen.
- Namen dürfen Buchstaben, Leerzeichen und Zahlen sowie die folgenden Sonderzeichen enthalten: `+ - = . _ : /`
- Die maximale Anzahl an Tags pro Ressource beträgt 50.
- AWS-zugewiesenen Tag-Namen und -Werten wird automatisch das `aws:` Präfix zugewiesen, das Sie nicht zuweisen können. AWS-zugewiesene Tagnamen werden nicht auf das Tag-Limit von 50 angerechnet. Von Benutzern zugewiesene Tag-Namen haben das Präfix `user:` im Kostenzuordnungsbericht.
- Sie können die Anwendung eines Tags nicht rückdatieren.

Bewährte Sicherheitsmethoden für Timestream for InfluxDB

Optimiere Schreibvorgänge in InfluxDB

Wie jede andere Zeitreihendatenbank ist InfluxDB so konzipiert, dass Daten in Echtzeit aufgenommen und verarbeitet werden können. Um die optimale Leistung des Systems zu gewährleisten, empfehlen wir beim Schreiben von Daten in InfluxDB die folgenden Optimierungen:

- Batch-Schreibvorgänge: Schreiben Sie beim Schreiben von Daten in InfluxDB Daten in Batches, um den Netzwerkaufwand für jede Schreibanforderung zu minimieren. Die optimale Batchgröße beträgt 5000 Zeilen Leitungsprotokoll pro Schreibanforderung. Um mehrere Zeilen in eine Anforderung zu schreiben, muss jede Zeile des Zeilenprotokolls durch eine neue Zeile (\n) getrennt werden.
- Tags nach Schlüssel sortieren: Bevor Sie Datenpunkte in InfluxDB schreiben, sortieren Sie die Tags nach Schlüsseln in lexikografischer Reihenfolge.

```
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
```

- Verwenden Sie die größtmögliche Zeitgenauigkeit: — InfluxDB schreibt Daten mit Nanosekundengenauigkeit. Wenn Ihre Daten jedoch nicht in Nanosekunden erfasst werden, müssen Sie nicht mit dieser Genauigkeit schreiben. Verwenden Sie für eine bessere Leistung die größtmögliche Genauigkeit für Zeitstempel. Sie können die Schreibgenauigkeit angeben, wenn:
  - Wenn SDK Sie das verwenden, können Sie das angeben WritePrecision , wenn Sie das Zeitattribut Ihres Punktes festlegen. Weitere Informationen zu InfluxDB-Clientbibliotheken finden Sie in der InfluxDB-Dokumentation.
  - Wenn Sie Telegraf verwenden, konfigurieren Sie die Zeitgenauigkeit in der Telegraf-Agentenkonfiguration. Die Genauigkeit wird als Intervall mit einer Ganzzahl (+) angegeben (z. B. 0s,10ms,2us,4s). Gültige Zeiteinheiten sind „ns“, „us“, „ms“ und „s“.

    ```
    [agent]
     interval ="10s"
     metric_batch_size="5000"
     precision = "0s"
    ```

- Gzip-Komprimierung verwenden: — Verwenden Sie die Gzip-Komprimierung, um Schreibvorgänge in InfluxDB zu beschleunigen und die Netzwerkbandbreite zu reduzieren. Benchmarks haben gezeigt, dass die Geschwindigkeit bei der Komprimierung von Daten um das Fünffache verbessert wird.
  - Wenn Sie Telegraf verwenden, setzen Sie in der Konfiguration des InfluxDB_v2-Ausgabe-Plugins in Ihrer telegraf.conf die Option content_encoding auf gzip:

    ```
    [[outputs.influxdb_v2]]
      urls = ["http://localhost:8086"]
      # ...
      content_encoding = "gzip"
    ```

  - Bei der Verwendung von Clientbibliotheken bietet jede InfluxDB-Clientbibliothek Optionen zum Komprimieren von Schreibanforderungen oder erzwingt standardmäßig die Komprimierung. Die Methode zum Aktivieren der Komprimierung ist für jede Bibliothek unterschiedlich. Spezifische Anweisungen finden Sie in der InfluxDB-Dokumentation
  - Wenn Sie den API `/api/v2/write` InfluxDB-Endpunkt zum Schreiben von Daten verwenden, komprimieren Sie die Daten mit gzip und setzen Sie den Content-Encoding-Header auf gzip.

Design für Leistung

Entwerfen Sie Ihr Schema für einfachere und leistungsfähigere Abfragen. Die folgenden Richtlinien stellen sicher, dass Ihr Schema einfach abzufragen ist und die Abfrageleistung maximiert wird:

- Auf Abfragen zugeschnittenes Design: Wählen Sie Maße, Tagschlüssel und Feldschlüssel, die einfach abzufragen sind. Um dieses Ziel zu erreichen, folgen Sie diesen Prinzipien:
  - Verwenden Sie Messungen, die einen einfachen Namen haben und das Schema genau beschreiben.
  - Vermeiden Sie es, denselben Namen für einen Tag-Schlüssel und einen Feldschlüssel innerhalb desselben Schemas zu verwenden.
  - Vermeiden Sie die Verwendung von reservierten Flux-Schlüsselwörtern und Sonderzeichen in Tag- und Feldschlüsseln.
  - Tags speichern Metadaten, die die Felder beschreiben und in vielen Datenpunkten vorkommen.
  - Felder speichern eindeutige oder stark variable Daten, in der Regel numerische Datenpunkte.
  - Messungen und Schlüssel sollten keine Daten enthalten, sondern dazu verwendet werden, Daten entweder zu aggregieren oder zu beschreiben. Daten werden in Tag- und Feldwerten gespeichert.
- Behalten Sie Ihre Zeitreihenkardinalität unter Kontrolle Eine hohe Reihenkardinalität ist eine der Hauptursachen für eine verringerte Schreib- und Leseleistung in InfluxDB. Im Kontext von InfluxDB bezieht sich eine hohe Kardinalität auf das Vorhandensein einer sehr großen Anzahl eindeutiger Tag-Werte. Tag-Werte werden in InfluxDB indexiert, was bedeutet, dass eine sehr hohe Anzahl eindeutiger Werte einen größeren Index generiert, was die Datenaufnahme und die Abfrageleistung verlangsamen kann.

  Um potenzielle Probleme im Zusammenhang mit hoher Kardinalität besser zu verstehen und zu lösen, können Sie die folgenden Schritte ausführen:
  - Machen Sie sich mit den Ursachen einer hohen Kardinalität vertraut
  - Messen Sie die Kardinalität Ihrer Buckets
  - Ergreifen Sie Maßnahmen, um eine hohe Kardinalität zu beheben
- Ursachen für eine hohe Serienkardinalität InfluxDB indexiert die Daten auf der Grundlage von Messungen und Tags, um das Lesen von Daten zu beschleunigen. Jeder Satz indizierter Datenelemente bildet einen Serienschlüssel. Tags, die sehr variable Informationen wie eindeutigeIDs, Hashes und zufällige Zeichenketten enthalten, führen zu einer großen Anzahl von

Serien, was auch als hohe Serienkardinalität bezeichnet wird. Eine hohe Serienkardinalität ist der Hauptgrund für die hohe Speicherauslastung in InfluxDB.

- Messung der Serienkardinalität Wenn Sie Leistungseinbußen oder eine ständig steigende Speicherauslastung in Ihrer Timestream for InfluxDB-Instance feststellen, empfehlen wir, die Serienkardinalität Ihrer Buckets zu messen.

  InfluxDB bietet Funktionen, mit denen Sie die Serienkardinalität sowohl in Flux als auch in InfluxQL messen können.

  - Verwenden Sie in Flux die Funktion `influxdb.cardinality()`
  - Verwenden Sie in FluxQL den Befehl `SHOW SERIES CARDINALITY`

  In beiden Fällen gibt die Engine die Anzahl der eindeutigen Serienschlüssel in Ihren Daten zurück. Denken Sie daran, dass es nicht empfohlen wird, mehr als 10 Millionen Serienschlüssel auf einer Ihrer Timestream-Instanzen für InfluxDB-Instances zu haben.

- Ursachen für eine hohe Serienkardinalität Wenn Sie feststellen, dass einer Ihrer Buckets eine hohe Kardinalität aufweist, können Sie einige Korrekturmaßnahmen ergreifen, um das Problem zu beheben:

  - Überprüfen Sie Ihre Stichwörter: Stellen Sie sicher, dass Ihre Workloads nicht zu Fällen führen, in denen Tags für die meisten Einträge eindeutige Werte haben. Dies kann in Fällen der Fall sein, in denen die Anzahl der eindeutigen Tag-Werte im Laufe der Zeit immer zunimmt, oder wenn Nachrichten vom Typ Protokoll in die Datenbank geschrieben werden, wo jede Nachricht eine eindeutige Kombination aus Zeitstempel, Tags usw. haben würde. Sie können den folgenden Flux-Code verwenden, um herauszufinden, welche Tags am meisten zu Ihren Problemen mit hoher Kardinalität beitragen:

```
// 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")
```

Wenn Sie eine sehr hohe Kardinalität feststellen, kann es bei der obigen Abfrage zu einem Timeout kommen. Wenn es zu einem Timeout kommt, führen Sie die folgenden Abfragen nacheinander aus.

Generieren Sie eine Liste von Tags:

```
// Generate a list of tagsimport "influxdata/influxdb/schema"

schema.tagKeys(bucket: "example-bucket")
```

Zählen Sie eindeutige Tag-Werte für jedes 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()
```

Wir empfehlen, diese zu unterschiedlichen Zeitpunkten auszuführen, um festzustellen, welches Tag schneller wächst.

- Verbessern Sie Ihr Schema: Folgen Sie den Modellierungsempfehlungen, die wir in unserem beschrieben habenBewährte Sicherheitsmethoden für Timestream for InfluxDB.
- Entfernen oder aggregieren Sie ältere Daten, um die Kardinalität zu reduzieren: Überlegen Sie, ob Ihre Anwendungsfälle alle Daten benötigen, die Ihre Probleme mit hoher Kardinalität verursachen. Wenn diese Daten nicht mehr benötigt werden oder nicht mehr häufig darauf zugegriffen wird, können Sie sie aggregieren, löschen oder zur langfristigen Speicherung und Analyse in eine andere Engine wie Timestream for Live Analytics exportieren.

Fehlerbehebung

Warnung: Die „Dev“ -Version wurde nicht erkannt

Die Warnung 'WARN: Die vom Server gemeldete Version „Dev“ konnte nicht analysiert werden, vorausgesetzt, die letzten Sicherungen/Wiederherstellungen APIs werden unterstützt' wird möglicherweise während der Migration angezeigt. Diese Warnung kann ignoriert werden.

Die Migration ist während der Wiederherstellungsphase fehlgeschlagen

Falls die Migration während der Wiederherstellungsphase fehlschlägt, können Benutzer die `--retry-restore-dir` Markierung verwenden, um die Wiederherstellung erneut zu versuchen. Verwenden Sie das `--retry-restore-dir` Flag mit einem Pfad zu einem zuvor gesicherten Verzeichnis, um die Sicherungsphase zu überspringen und die Wiederherstellungsphase erneut zu versuchen. Das erstellte Backup-Verzeichnis, das für eine Migration verwendet wurde, wird angezeigt, wenn eine Migration während der Wiederherstellung fehlschlägt.

Zu den möglichen Gründen für das Fehlschlagen einer Wiederherstellung gehören:

- Ungültiges InfluxDB-Zieltoken — Ein in der Zielinstanz vorhandener Bucket mit demselben Namen wie in der Quellinstanz. Verwenden Sie für einzelne Bucket-Migrationen die `--dest-bucket` Option, einen eindeutigen Namen für den migrierten Bucket festzulegen
- Verbindungsfehler, entweder mit den Quell- oder Zielhosts oder mit einem optionalen S3-Bucket.

Grundlegende Betriebsrichtlinien für Amazon Timestream für InfluxDB

Im Folgenden finden Sie grundlegende Betriebsrichtlinien, die jeder bei der Arbeit mit Amazon Timestream for InfluxDB befolgen sollte. Beachten Sie, dass das Amazon Timestream for InfluxDB Service Level Agreement vorschreibt, dass Sie diese Richtlinien befolgen:

- Verwenden Sie Metriken, um Ihren Arbeitsspeicher und Ihre CPU Speichernutzung zu überwachen. Sie können Amazon so einrichten CloudWatch , dass Sie benachrichtigt werden, wenn sich die Nutzungsmuster ändern oder wenn Sie sich der Kapazität Ihrer Bereitstellung nähern. Auf diese Weise können Sie leichter die Leistung und Verfügbarkeit des Systems wahren.
- Skalieren Sie Ihre DB-Instance, wenn Sie die Grenzen der Speicherkapazität beinahe erreicht haben. Sie sollten etwas Puffer in Speicher und Arbeitsspeicher haben, um unvorhergesehene Nachfragesteigerungen seitens Ihrer Anwendungen bewältigen zu können. Beachten Sie, dass Sie

zu diesem Zeitpunkt eine neue Instance erstellen und Ihre Daten migrieren müssen, um dies zu erreichen.

- Wenn Ihr Datenbank-Workload mehr I/O benötigt, als Sie bereitgestellt haben, ist die Wiederherstellung nach einem Failover oder einem Datenbankfehler langsam. Führen Sie einen der folgenden Schritte aus, um die I/O-Kapazität einer DB-Instance zu steigern:
  - Migrieren Sie zu einer anderen DB-Instance mit höherer I/O-Kapazität.
  - Wenn Sie bereits integrierten Influx-Speicherspeicher IOPS verwenden, stellen Sie einen Speichertyp bereit, bei dem ein höherer Speichertyp IOPS enthalten ist.
- Wenn Ihre Client-Anwendung die Domain Name Service (DNS) -Daten Ihrer DB-Instances zwischenspeichert, legen Sie einen time-to-live (TTL) -Wert von weniger als 30 Sekunden fest. Die zugrunde liegende IP-Adresse einer DB-Instance kann sich nach einem Failover ändern. Das Zwischenspeichern der DNS Daten über einen längeren Zeitraum kann daher zu Verbindungsausfällen führen. Ihre Anwendung versucht möglicherweise, eine Verbindung zu einer IP-Adresse herzustellen, die nicht mehr in Betrieb ist.

Empfehlungen für DB-Instances RAM

Eine bewährte Methode für die Leistung von Amazon Timestream for InfluxDB besteht darin, RAM so viel zuzuweisen, dass sich Ihr Arbeitssatz fast vollständig im Speicher befindet. Der Arbeitssatz umfasst die Daten und Indizes, die häufig auf Ihrer Instance verwendet werden. Je häufiger Sie die DB-Instance verwenden, desto größer wird der Arbeitssatz.

Sicherheit in Timestream für InfluxDB

Cloud-Sicherheit hat höchste Priorität. AWS Als AWS Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame Verantwortung von Ihnen AWS und Ihnen. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, die AWS Dienste in der AWS Cloud ausführt. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Die Wirksamkeit unserer Sicherheitsfunktionen wird regelmäßig von externen Prüfern im Rahmen des AWS -Compliance-Programms getestet und überprüft. Weitere Informationen zu den Compliance-

Programmen, die für Timestream for InfluxDB gelten, finden Sie unter AWS Services in Scope by Compliance Program.

- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS Dienst, den Sie nutzen. In Ihre Verantwortung fallen außerdem weitere Faktoren, wie z. B. die Vertraulichkeit der Daten, die Anforderungen Ihrer Organisation sowie geltende Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung anwenden können, wenn Sie Timestream for InfluxDB verwenden. Die folgenden Themen zeigen Ihnen, wie Sie Timestream für InfluxDB konfigurieren, um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Sie erfahren auch, wie Sie andere AWS Dienste verwenden können, die Ihnen helfen können, Ihre Timestream-Ressourcen für InfluxDB zu überwachen und zu sichern.

Themen

Übersicht

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung anwenden können, wenn Sie Amazon Timestream for InfluxDB verwenden. Die folgenden Themen zeigen Ihnen, wie Sie Amazon Timestream for InfluxDB konfigurieren, um Ihre Sicherheits- und

Compliance-Ziele zu erreichen. Sie erfahren auch, wie Sie andere AWS Dienste nutzen können, mit denen Sie Ihre Amazon Timestream for InfluxDB-Ressourcen überwachen und sichern können.

Sie können den Zugriff auf Ihre Amazon Timestream for InfluxDB-Ressourcen und Ihre Datenbanken auf einer DB-Instance verwalten. Die Methode, mit der Sie den Zugriff verwalten, hängt davon ab, welche Art von Aufgabe der Benutzer mit Amazon Timestream for InfluxDB ausführen muss:

- Führen Sie Ihre DB-Instance in einer Virtual Private Cloud (VPC) aus, die auf dem VPC Amazon-Service für Netzwerkzugriffskontrolle basiert.
- Verwenden Sie Richtlinien für AWS Identity and Access Management (IAM), um Berechtigungen zuzuweisen, die festlegen, wer Amazon Timestream for InfluxDB-Ressourcen verwalten darf. Sie können IAM damit beispielsweise festlegen, wer DB-Instances erstellen, beschreiben, ändern und löschen, Ressourcen taggen oder Sicherheitsgruppen ändern darf.
- Verwenden Sie Sicherheitsgruppen, um zu steuern, welche IP-Adressen oder EC2 Amazon-Instances eine Verbindung zu Ihren Datenbanken auf einer DB-Instance herstellen können. Wenn Sie eine DB-Instance zum ersten Mal erstellen, ist sie nur über Regeln zugänglich, die von einer zugehörigen Sicherheitsgruppe festgelegt wurden.
- Verwenden Sie Secure Socket Layer (SSL) - oder Transport Layer Security (TLS) -Verbindungen mit Ihren DB-Instances.
- Verwenden Sie die Sicherheitsfunktionen Ihrer InfluxDB-Engine, um zu steuern, wer sich bei den Datenbanken einer DB-Instance anmelden kann. Diese Funktionen arbeiten genauso, als würden sich die Datenbanken in Ihrem lokalen Netzwerk befinden. Weitere Informationen finden Sie unter Sicherheit in Timestream für InfluxDB.

> **Note**
>
> Sie müssen die Sicherheit nur für Ihre Anwendungsfälle konfigurieren. Sie müssen keinen Sicherheitszugriff für Prozesse konfigurieren, die Amazon Timestream for InfluxDB verwaltet. Dazu gehört das Erstellen von Backups, das Replizieren von Daten zwischen einer primären DB-Instance und einem Lesereplikat und andere Prozesse.

Themen

- Allgemeine Sicherheit

Allgemeine Sicherheit

Themen

Berechtigungen

InfluxDB-Benutzern sollten Berechtigungen mit den geringsten Rechten gewährt werden. Während der Migration sollten nur Token verwendet werden, die bestimmten Benutzern gewährt wurden, anstelle von Operator-Token.

Timestream for InfluxDB verwendet IAM Berechtigungen, um Benutzerberechtigungen zu kontrollieren. Wir empfehlen, Benutzern Zugriff auf die spezifischen Aktionen und Ressourcen zu gewähren, die sie benötigen. Weitere Informationen finden Sie unter Zugriff mit geringsten Rechten gewähren.

Netzwerkzugriff

Das Influx-Migrationsskript kann lokal funktionieren und Daten zwischen zwei InfluxDB-Instanzen auf demselben System migrieren. Es wird jedoch davon ausgegangen, dass der primäre Anwendungsfall für Migrationen die Migration von Daten über das Netzwerk ist, entweder ein lokales oder ein öffentliches Netzwerk. Damit gehen Sicherheitsüberlegungen einher. Das Influx-Migrationsskript überprüft standardmäßig TLS Zertifikate für Instances mit TLS aktivierter Option: Wir empfehlen Benutzern, die Aktivierung TLS in ihren InfluxDB-Instances vorzunehmen und die `--skip-verify` Option für das Skript nicht zu verwenden.

Wir empfehlen Ihnen, eine Zulassungsliste zu verwenden, um den Netzwerkverkehr auf Quellen zu beschränken, die Sie erwarten. Sie können dies tun, indem Sie den Netzwerkverkehr nur aus bekannten Quellen auf die InfluxDB-Instances beschränken. IPs

Abhängigkeiten

Es sollten die neuesten Hauptversionen aller Abhängigkeiten verwendet werden, einschließlich InfluxCLI, InfluxDB, Python, des Request-Moduls und optionaler Abhängigkeiten wie und. `mountpoint-s3` `rclone`

S3-Buckets

Wenn S3-Buckets als temporärer Speicher für die Migration verwendet werden, empfehlen wir, den öffentlichen Zugriff zu aktivierenTLS, zu versionieren und zu deaktivieren.

Verwendung von S3-Buckets für die Migration

1. Öffnen Sie den AWS Management Console, navigieren Sie zu Amazon Simple Storage Service und wählen Sie dann Buckets aus.
2. Wählen Sie den Bucket aus, den Sie verwenden möchten.
3. Wählen Sie die Registerkarte Berechtigungen.
4. Wählen Sie unter Block public access (bucket settings) (Öffentlichen Zugriff blockieren (Bucket-Einstellungen)), die Option Edit (Bearbeiten).
5. Markieren Sie Alle öffentlichen Zugänge blockieren.
6. Wählen Sie Änderungen speichern.
7. Wählen Sie unter Bucket-Richtlinie Bearbeiten aus.
8. Geben Sie Folgendes ein und <example-bucket>ersetzen Sie es durch Ihren Bucket-Namen, um die Verwendung von TLS Version 1.2 oder höher für Verbindungen zu erzwingen:

```
{
    "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. Wählen Sie Änderungen speichern.
10. Wählen Sie die Registerkarte Eigenschaften aus.
11. Wählen Sie unter Bucket Versioning (Bucket-Versioning) die Option Edit (Bearbeiten).
12. Markieren Sie Aktivieren.
13. Wählen Sie Änderungen speichern.

Informationen zu den bewährten Sicherheitspraktiken für Amazon S3 S3-Buckets finden Sie unter Bewährte Sicherheitsmethoden für Amazon Simple Storage Service.

Datenbankauthentifizierung mit Amazon Timestream für InfluxDB

Amazon Timestream for InfluxDB unterstützt zwei Methoden zur Authentifizierung von Datenbankbenutzern.

Bei der Datenbankauthentifizierung mit Passwort und Zugriffstoken werden unterschiedliche Methoden zur Authentifizierung bei der Datenbank verwendet. Daher kann sich ein bestimmter Benutzer mit nur einer einzigen Authentifizierungsmethode bei einer Datenbank anmelden. In beiden Fällen führt InfluxDB die gesamte Verwaltung von Benutzerkonten und Token durch. API

Passwortauthentifizierung

Während der Erstellung der InfluxDB-DB-Instance haben Sie eine Organisation, einen Benutzer und ein Passwort erstellt. Der Benutzer hat die Berechtigung, alles in Ihrer Timestream for InfluxDB-DB-Instance zu verwalten. Mit dieser Kombination aus Benutzername und Passwort können Sie über die InfluxUI auf LogIn Ihre Instance zugreifen und den Influx CLI auch verwenden, um ein Operator-Token zu generieren.

Ein Operator-Token ist erforderlich, um Benutzer zu erstellen, Buckets, Organisationen usw. zu löschen. Weitere Informationen finden Sie unter Datenbankauthentifizierungsoptionen.

APITokens

APIInfluxDB-Token gewährleisten eine sichere Interaktion zwischen InfluxDB und externen Tools wie Clients oder Anwendungen. Ein API Token gehört einem bestimmten Benutzer und identifiziert InfluxDB-Berechtigungen innerhalb der Organisation des Benutzers.

In InfluxDB gibt es drei Arten von API Token:

- Operator-Token: Gewährt vollen Lese- und Schreibzugriff auf alle Organisationen und alle Organisationsressourcen in OSS InfluxDB 2.x. Für einige Operationen, z. B. das Abrufen der Serverkonfiguration, sind Bedienerberechtigungen erforderlich. Um nach Abschluss des Einrichtungsvorgangs manuell ein Operator-Token mit der InfluxDB-Benutzeroberfläche oder Influx CLI zu erstellen, müssen Sie ein vorhandenes Operator-Token oder Ihren Benutzernamen und Ihr Passwort verwenden. `api/v2` API Informationen zum Erstellen eines neuen Operator-Tokens, ohne ein vorhandenes zu verwenden, finden Sie in der Influxd-Wiederherstellungsauthentifizierung. CLI

  > **Important**
  >
  > Da Operator-Token vollen Lese- und Schreibzugriff auf alle Organisationen in der Datenbank haben, empfehlen wir, für jede Organisation ein All-Access-Token zu erstellen und dieses zur Verwaltung von InfluxDB zu verwenden. Dies trägt dazu bei, versehentliche Interaktionen zwischen Organisationen zu verhindern.

- APIAll-Access-Token: Gewährt vollständigen Lese- und Schreibzugriff auf alle Ressourcen in einer Organisation.
- Tokens mit Lese-/Schreibzugriff: Gewährt Lese-, Schreibzugriff oder beides für bestimmte Bereiche in einer Organisation.

Alle InfluxDb Token sind langlebige Token ohne festgelegtes Ablaufdatum. Es wird daher nicht empfohlen, Ihren Mobilfunkanbieter oder alle Zugriffstoken zu verwenden, um Überwachungsdaten von Ihren Kunden oder Telegraf-Agenten zu senden oder sie in Ihre Dashboard-Anwendungen einzubetten. Erstellen Sie für diese Anwendungen Lese-/Schreib-Tokens mit nur den erforderlichen Berechtigungen, um die Aufgabe zu erledigen. Weitere Informationen zum Erstellen eines InfluxDB-Tokens finden Sie unter Token erstellen.

Secrets

InfluxDB-Operator-Token werden beim Instanz-Setup generiert; andere Arten von Token, wie All-Access- und Lese-/Schreib-Tokens, können mit der Mehrbenutzer-Rotationsfunktion Influx, Influx v2 API oder Timestream CLI for InfluxDB erstellt werden. Informationen zum Generieren, Anzeigen, Zuweisen und API Löschen von Token finden Sie unter Tokens verwalten.

Wir empfehlen, Timestream für InfluxDB-Token zu rotieren, indem Sie häufig Token über AWS Secrets Manager Umgebungsvariablen verwenden und speichern. Informationen zum Rotieren von Timestream für InfluxDB-Benutzer und Das Geheimnis rotierenTokens finden Sie unter Verwenden von Tokens für Tokens in Umgebungsvariablen.

Weitere Informationen finden Sie auch unter:

- Infrastruktursicherheit in Amazon Timestream für InfluxDB
- Bewährte Sicherheitsmethoden für Timestream for InfluxDB

Wie Amazon Timestream for InfluxDB Geheimnisse verwendet

Timestream for InfluxDB unterstützt die Authentifizierung von Benutzernamen und Passwörtern über die Benutzeroberfläche sowie Token-Anmeldeinformationen für Client- und Anwendungsverbindungen mit den geringsten Rechten. Timestream for InfluxDB-Benutzer haben `allAccess` Berechtigungen innerhalb ihrer Organisation, während Tokens beliebige Berechtigungen haben können. Gemäß den bewährten Methoden für die sichere API Tokenverwaltung sollten Benutzer so eingerichtet werden, dass sie Token für den detaillierten Zugriff innerhalb einer Organisation verwalten können. Zusätzliche Informationen zu bewährten Methoden für Administratoren mit Timestream for InfluxDB finden Sie in der Influxdata-Dokumentation.

AWS Secrets Manager ist ein geheimer Speicherdienst, mit dem Sie Datenbankanmeldedaten, API Schlüssel und andere geheime Informationen schützen können. Dann können Sie in Ihrem Code hartcodierte Anmeldeinformationen durch einen API Aufruf von Secrets Manager ersetzen. Auf diese Weise wird sichergestellt, dass das Geheimnis nicht von jemandem missbraucht werden kann, der Ihren Code untersucht, da das Geheimnis nicht vorhanden ist. Einen Überblick über Secrets Manager finden Sie unter Was ist AWS Secrets Manager.

Wenn Sie eine Datenbankinstanz erstellen, erstellt Timestream for InfluxDB automatisch ein Administratorgeheimnis, das Sie mit der Mehrbenutzer-Rotationsfunktion verwenden können. AWS Lambda Um Timestream für InfluxDB-Benutzer und Tokens zu rotieren, müssen Sie für jeden Benutzer oder Token, den Sie rotieren möchten, von Hand ein neues Geheimnis erstellen. Jedes Geheimnis kann mithilfe einer Lambda-Funktion so konfiguriert werden, dass es nach einem Zeitplan rotiert. Der Prozess zur Einrichtung eines neuen rotierenden Geheimnisses besteht aus dem Hochladen des Lambda-Funktionscodes, der Konfiguration der Lambda-Rolle, der Definition des neuen Geheimnisses und der Konfiguration des geheimen Rotationsplans.

Was ist in dem Geheimnis enthalten?

Wenn Sie Timestream for InfluxDB-Benutzeranmeldeinformationen im Secret speichern, verwenden Sie das folgende Format.

Einzelbenutzer:

```
{
  "engine": "<required: must be set to 'timestream-influxdb'>",
  "username": "<required: username>",
  "password": "<required: password>",
  "dbIdentifier": "<required: DB identifier>"
}
```

Wenn Sie eine Timestream for InfluxDB-Instance erstellen, wird automatisch ein Administratorgeheimnis mit Anmeldeinformationen für die Mehrbenutzer-Lambda-Funktion im Secrets Manager gespeichert. Stellen Sie den `Authentication Properties Secret Manager ARN` Wert `adminSecretArn` auf der Übersichtsseite der DB-Instance oder auf den Wert eines Admin-Geheimnisses ein. ARN Um ein neues Administratorgeheimnis zu erstellen, müssen Sie bereits über die zugehörigen Anmeldeinformationen verfügen und die Anmeldeinformationen müssen über Administratorrechte verfügen.

Wenn Sie Timestream for InfluxDB-Token-Anmeldeinformationen im Secret speichern, verwenden Sie das folgende Format.

Mehrbenutzer:

```
{
  "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']>"
}
```

Wenn Sie Timestream for InfluxDB-Administratoranmeldedaten im Secret speichern, verwenden Sie das folgende Format:

Geheimer Admin-Schlüssel:

```
{
  "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>"
}
```

Um die automatische Rotation für das Geheimnis zu aktivieren, muss das Geheimnis die richtige JSON Struktur haben. Informationen Das Geheimnis rotieren zum Rotieren von Timestream für InfluxDB-Secrets finden Sie unter.

Ändern eines Geheimnisses

Die während der Erstellung der Timestream for InfluxDB-Instanz generierten Anmeldeinformationen werden in einem Secrets Manager Manager-Geheimnis in Ihrem Konto gespeichert. Das GetDbInstanceAntwortobjekt enthält ein`influxAuthParametersSecretArn`, die den Amazon-Ressourcennamen (ARN) geheim hält. Das Geheimnis wird erst aufgefüllt, wenn Ihre Timestream for InfluxDB-Instance verfügbar ist. Dies ist eine READONLY Kopie, da sich ein Teil updates/modifications/deletions dieses Geheimnisses nicht auf die erstellte DB-Instance auswirkt. Wenn Sie dieses Geheimnis löschen, bezieht sich die APIAntwort immer noch auf das gelöschte GeheimnisARN.

Um ein neues Token in der Timestream for InfluxDB-Instance zu erstellen, anstatt vorhandene Token-Anmeldeinformationen zu speichern, können Sie Nicht-Operator-Token erstellen, indem Sie den `token` Wert im Secret leer lassen und die Mehrbenutzer-Rotationsfunktion verwenden, wobei die `AUTHENTICATION_CREATION_ENABLED` Lambda-Umgebungsvariable auf gesetzt ist. `true` Wenn Sie ein neues Token erstellen, werden die im Secret definierten Berechtigungen dem Token zugewiesen und können nach der ersten erfolgreichen Rotation nicht mehr geändert werden. Weitere Informationen zur Rotation von Secrets finden Sie unter Rotating AWS Secrets Manager Secrets.

Wenn ein Geheimnis gelöscht wird, wird der zugehörige Benutzer oder das zugehörige Token in der Timestream for InfluxDB-Instance nicht gelöscht.

Das Geheimnis rotieren

Sie verwenden die Lambda-Funktionen Timestream for InfluxDB-Rotation für Einzel- und Mehrbenutzerrotation, um Timestream für InfluxDB-Benutzer- und Token-Anmeldeinformationen zu rotieren. Verwenden Sie die Lambda-Funktion für Einzelbenutzer, um die Benutzeranmeldeinformationen für Ihre Timestream for InfluxDB-Instance zu rotieren, und verwenden Sie die Lambda-Funktion für mehrere Benutzer, um Token-Anmeldeinformationen für Ihre Timestream for InfluxDB-Instance zu rotieren.

Das Rotieren von Benutzern und Tokens mit den Lambda-Funktionen für Einzel- und Mehrbenutzer ist optional. Der Timestream für InfluxDB-Anmeldeinformationen läuft nie ab und alle offengelegten Anmeldeinformationen stellen ein Risiko für böswillige Aktionen gegen Ihre DB-Instance dar. Der Vorteil der Rotation von Timestream für InfluxDB-Anmeldeinformationen mit Secrets Manager ist eine zusätzliche Sicherheitsebene, die den Angriffsvektor offengelegter Anmeldeinformationen auf das Zeitfenster bis zum nächsten Rotationszyklus beschränkt. Wenn für Ihre DB-Instance kein Rotationsmechanismus vorhanden ist, sind alle offengelegten Anmeldeinformationen gültig, bis sie manuell gelöscht werden.

Sie können Secrets Manager so konfigurieren, dass Secrets automatisch nach einem von Ihnen angegebenen Zeitplan für Sie rotiert werden. So können Sie Secrets mit langer Einsatzdauer durch Secrets mit kurzer Einsatzdauer ersetzen und damit das Risiko einer Kompromittierung erheblich verringern. Weitere Informationen zur Rotation von Secrets mit Secrets Manager finden Sie unter Rotation von AWS Secrets Manager Secrets.

Rotierende Benutzer

Wenn Sie Benutzer mit der Lambda-Funktion für Einzelbenutzer rotieren, wird dem Benutzer nach jeder Rotation ein neues zufälliges Passwort zugewiesen. Weitere Informationen zum Aktivieren der automatischen Rotation finden Sie unter Automatische Rotation für AWS Secrets Manager einrichten, die keine Datenbank sind.

Rotierende Administratorgeheimnisse

Um ein Administratorgeheimnis zu rotieren, verwenden Sie die Rotationsfunktion für einzelne Benutzer. Sie müssen dem Secret die `dbIdentifier` Werte `engine` und hinzufügen, da diese Werte bei der DB-Initialisierung nicht automatisch aufgefüllt werden. Die Was ist in dem Geheimnis enthalten? vollständige geheime Vorlage finden Sie unter.

Um ein Administratorgeheimnis für eine Timestream for InfluxDB-Instance zu finden, verwenden Sie das Administratorgeheimnis ARN auf der Übersichtsseite der Timestream for InfluxDB-Instance. Es

wird empfohlen, dass Sie alle Timestream for InfluxDB-Administratorgeheimnisse rotieren, da Admin-Benutzer über erhöhte Berechtigungen für die Timestream for InfluxDB-Instance verfügen.

Lambda-Rotationsfunktion

Sie können einen Timestream für InfluxDB-Benutzer mit der Einzelbenutzer-Rotationsfunktion rotieren, indem Sie das Was ist in dem Geheimnis enthalten? mit einem neuen Geheimnis verwenden und die erforderlichen Felder für Ihren Timestream for InfluxDB-Benutzer hinzufügen. Weitere Informationen zu Lambda-Funktionen mit geheimer Rotation finden Sie unter Rotation nach Lambda-Funktion.

Die Rotationsfunktion für einzelne Benutzer authentifiziert sich bei der Timestream for InfluxDB-DB-Instance unter Verwendung der im Secret definierten Anmeldeinformationen, generiert dann ein neues zufälliges Passwort und legt das neue Passwort für den Benutzer fest. Weitere Informationen zu Lambda-Funktionen mit geheimer Rotation finden Sie unter Rotation nach Lambda-Funktion.

Rollenberechtigungen für die Ausführung von Lambda-Funktionen

Verwenden Sie die folgende IAM Richtlinie als Rolle für die Lambda-Funktion für Einzelbenutzer. Die Richtlinie gibt der Lambda-Funktion die erforderlichen Berechtigungen, um eine geheime Rotation für Timestream für InfluxDB-Benutzer durchzuführen.

Ersetzen Sie alle unten in der IAM Richtlinie aufgeführten Elemente durch Werte aus Ihrem Konto: AWS

- {rotating_secret_arn} — Die Informationen ARN für das Geheimnis, das rotiert wird, finden Sie in den geheimen Details des Secrets Secrets Manager.
- {db_instance_arn} — Den Timestream für die InfluxDB-Instance finden Sie auf der Übersichtsseite der Timestream for InfluxDB-Instance. ARN

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

Rotierende Token

Sie können ein Timestream for InfluxDB-Token mit der Mehrbenutzer-Rotationsfunktion rotieren, indem Sie das Was ist in dem Geheimnis enthalten? mit einem neuen Geheimnis verwenden und die erforderlichen Felder für Ihr Timestream for InfluxDB-Token hinzufügen. Weitere Informationen zu Lambda-Funktionen mit geheimer Rotation finden Sie unter Rotation nach Lambda-Funktion.

Sie können ein Timestream for InfluxDB-Token rotieren, indem Sie die Mehrbenutzer-Lambda-Funktion Timestream for InfluxDB verwenden. Setzen Sie die `AUTHENTICATION_CREATION_ENABLED` Umgebungsvariable `true` in der Lambda-Konfiguration auf, um die Token-Erstellung zu aktivieren. Um ein neues Token zu erstellen, verwenden Sie den Was ist in dem Geheimnis enthalten? für Ihren geheimen Wert. Lassen Sie das `token` Schlüssel-Wert-Paar im neuen Geheimnis weg und legen Sie den Wert `type` auf fest`allAccess`, oder definieren Sie die spezifischen Berechtigungen und legen Sie den Typ auf fest. `custom` Die Rotationsfunktion erstellt während des ersten Rotationszyklus ein neues Token. Sie können die Token-Berechtigungen nicht ändern, indem Sie das Geheimnis nach der Rotation bearbeiten. Bei allen nachfolgenden Rotationen werden die in der DB-Instance festgelegten Berechtigungen verwendet.

Lambda-Rotationsfunktion

Die Rotationsfunktion für mehrere Benutzer rotiert die Token-Anmeldeinformationen, indem ein neues, berechtigungsidentisches Token mit den Administratoranmeldedaten im geheimen Administratorschlüssel erstellt wird. Die Lambda-Funktion validiert den Tokenwert im Secret, bevor sie das Ersatztoken erstellt, den neuen Token-Wert im Secret speichert und das alte Token löscht. Wenn die Lambda-Funktion ein neues Token erstellt, überprüft sie zunächst, ob die `AUTHENTICATION_CREATION_ENABLED` Umgebungsvariable auf gesetzt ist`true`, dass das Geheimnis keinen Tokenwert enthält und dass der Tokentyp kein Typoperator ist.

Rollenberechtigungen für die Ausführung von Lambda-Funktionen

Verwenden Sie die folgende IAM Richtlinie als Rolle für die Mehrbenutzer-Lambda-Funktion. Die Richtlinie gibt der Lambda-Funktion die erforderlichen Berechtigungen, um eine geheime Rotation für Timestream for InfluxDB-Token durchzuführen.

Ersetzen Sie alle unten in der IAM Richtlinie aufgeführten Elemente durch Werte aus Ihrem Konto: AWS

- {rotating_secret_arn} — Die Informationen ARN für das Geheimnis, das rotiert wird, finden Sie in den geheimen Details des Secrets Secrets Manager.
- {authentication_properties_admin_secret_arn} — Das Timestream for InfluxDB-Administratorgeheimnis finden Sie auf der Übersichtsseite der Timestream for InfluxDB-Instance. ARN
- {db_instance_arn} — Den Timestream für die InfluxDB-Instance finden Sie auf der Übersichtsseite der Timestream for InfluxDB-Instance. ARN

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

Datenschutz in Timestream for InfluxDB

Das Modell der AWS gemeinsamen Verantwortung gilt für den Datenschutz in Amazon Timestream for InfluxDB. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der alle Systeme laufen. AWS Cloud Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie im Abschnitt Datenschutz FAQ. Informationen zum Datenschutz in Europa finden Sie im AWS Shared Responsibility Model und im GDPR Blogbeitrag auf dem AWS Security Blog.

Aus Datenschutzgründen empfehlen wir, dass Sie Ihre AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto eine Multi-Faktor-Authentifizierung (MFA).

- Verwenden SieSSL/TLS, um mit AWS Ressourcen zu kommunizieren. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Einrichtung API und Protokollierung von Benutzeraktivitäten mit AWS CloudTrail. Informationen zur Verwendung von CloudTrail Pfaden zur Erfassung von AWS Aktivitäten finden Sie unter Arbeiten mit CloudTrail Pfaden im AWS CloudTrail Benutzerhandbuch.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie FIPS 140-3 validierte kryptografische Module für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine benötigenAPI, verwenden Sie einen Endpunkt. FIPS Weitere Informationen zu den verfügbaren FIPS Endpunkten finden Sie unter Federal Information Processing Standard () 140-3. FIPS

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies schließt ein, wenn Sie mit Timestream for InfluxDB oder anderen AWS-Services über die Konsole arbeiten,, oder. API AWS CLI AWS SDKs Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie einem externen Server eine URL zur Verfügung stellen, empfehlen wir dringend, dass Sie keine Anmeldeinformationen in die aufnehmen, um Ihre Anfrage an diesen Server URL zu validieren.

Für detailliertere Informationen zu Datenschutzthemen von Timestream for InfluxDB wie Verschlüsselung im Ruhezustand und Schlüsselverwaltung wählen Sie eines der verfügbaren Themen unten aus.

Themen

- Verschlüsselung im Ruhezustand
- Verschlüsselung während der Übertragung

Verschlüsselung im Ruhezustand

Timestream für die InfluxDB-Verschlüsselung im Ruhezustand bietet verbesserte Sicherheit, indem alle Ihre Daten im Ruhezustand mithilfe von Verschlüsselungsschlüsseln verschlüsselt werden, die in () gespeichert sind.AWS Key Management ServiceAWS KMS Diese Funktionalität trägt zur Verringerung des Betriebsaufwands und der Komplexität bei, die mit dem Schutz sensibler Daten

einhergeht. Mit der Verschlüsselung ruhender Daten können Sie sicherheitsrelevante Anwendungen erstellen, die eine strenge Einhaltung der Verschlüsselungsvorschriften und der gesetzlichen Bestimmungen erfordern.

- Die Verschlüsselung ist auf Ihrer Timestream for InfluxDB-DB-Instance standardmäßig aktiviert und kann nicht ausgeschaltet werden. Der Industriestandard-Verschlüsselungsalgorithmus AES -256 ist der verwendete Standardverschlüsselungsalgorithmus.
- AWS KMS wird für die Verschlüsselung im Ruhezustand in Timestream für InfluxDB verwendet.
- Sie müssen Ihre DB-Instance-Client-Anwendungen nicht ändern, um Verschlüsselung zu verwenden.

Verschlüsselung während der Übertragung

Alle Ihre Timestream for InfluxDB-Daten werden bei der Übertragung verschlüsselt. Standardmäßig ist die gesamte Kommunikation zu und von Timestream for InfluxDB durch die Verwendung der Transport Layer Security () -Verschlüsselung geschützt. TLS

Der Datenverkehr zu und von Amazon Timestream for InfluxDB wird mit den unterstützten TLS Versionen 1.2 oder 1.3 gesichert.

Identity and Access Management für Amazon Timestream für InfluxDB

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf Ressourcen sicher zu kontrollieren. AWS IAMAdministratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), Timestream für InfluxDB-Ressourcen zu verwenden. IAMist eine AWS-Service , die Sie ohne zusätzliche Kosten nutzen können.

Themen

- Verwenden von serviceverknüpften Rollen für Amazon Timestream for InfluxDB
- AWS verwaltete Richtlinien für Amazon Timestream for InfluxDB
- Über einen Endpunkt eine Verbindung zu Timestream for InfluxDB herstellen VPC

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen als IAM Benutzer authentifiziert (angemeldet AWS) sein oder eine IAM Rolle übernehmen. Root-Benutzer des AWS-Kontos

Sie können sich AWS als föderierte Identität anmelden, indem Sie Anmeldeinformationen verwenden, die über eine Identitätsquelle bereitgestellt wurden. AWS IAM Identity Center (IAMIdentity Center-) Nutzer, die Single-Sign-On-Authentifizierung Ihres Unternehmens und Ihre Google- oder Facebook-Anmeldeinformationen sind Beispiele für föderierte Identitäten. Wenn Sie sich als föderierte Identität anmelden, hat Ihr Administrator zuvor einen Identitätsverbund mithilfe von Rollen eingerichtet. IAM Wenn Sie AWS mithilfe eines Verbunds darauf zugreifen, übernehmen Sie indirekt eine Rolle.

Je nachdem, welcher Benutzertyp Sie sind, können Sie sich beim AWS Management Console oder beim AWS Zugangsportal anmelden. Weitere Informationen zur Anmeldung finden Sie AWS unter So melden Sie sich bei Ihrem an AWS-Konto im AWS-Anmeldung Benutzerhandbuch.

Wenn Sie AWS programmgesteuert darauf zugreifen, AWS stellt es ein Software Development Kit (SDK) und eine Befehlszeilenschnittstelle (CLI) bereit, mit der Sie Ihre Anfragen mithilfe Ihrer Anmeldeinformationen kryptografisch signieren können. Wenn Sie keine AWS Tools verwenden, müssen Sie Anfragen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode, um Anfragen selbst zu signieren, finden Sie im IAMBenutzerhandbuch unter AWS Signature Version 4 für API Anfragen.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen angeben. AWS Empfiehlt beispielsweise, die Multi-Faktor-Authentifizierung (MFA) zu verwenden, um die Sicherheit Ihres Kontos zu erhöhen. Weitere Informationen finden Sie unter Multi-Faktor-Authentifizierung im AWS IAM Identity Center Benutzerhandbuch und AWS Multi-Faktor-Authentifizierung IAM im IAM Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAMBenutzer ist eine Identität innerhalb von Ihnen AWS-Konto , die über spezifische Berechtigungen für eine einzelne Person oder Anwendung verfügt. Wir empfehlen, sich nach Möglichkeit auf temporäre Anmeldeinformationen zu verlassen, anstatt IAM Benutzer mit langfristigen

Anmeldeinformationen wie Passwörtern und Zugriffsschlüsseln zu erstellen. Wenn Sie jedoch spezielle Anwendungsfälle haben, für die langfristige Anmeldeinformationen von IAM Benutzern erforderlich sind, empfehlen wir, die Zugriffsschlüssel abwechselnd zu verwenden. Weitere Informationen finden Sie im Benutzerhandbuch unter Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, für die IAM langfristige Anmeldeinformationen erforderlich sind.

Eine IAMGruppe ist eine Identität, die eine Sammlung von IAM Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise eine Gruppe benennen IAMAdminsund dieser Gruppe Berechtigungen zur Verwaltung von IAM Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie im Benutzerhandbuch unter Anwendungsfälle für IAM IAM Benutzer.

IAMRollen

Eine IAMRolle ist eine Identität innerhalb von Ihnen AWS-Konto , für die bestimmte Berechtigungen gelten. Sie ähnelt einem IAM Benutzer, ist jedoch keiner bestimmten Person zugeordnet. Um vorübergehend eine IAM Rolle in der zu übernehmen AWS Management Console, können Sie von einem Benutzer zu einer IAM Rolle (Konsole) wechseln. Sie können eine Rolle übernehmen, indem Sie eine AWS CLI AWS API OR-Operation aufrufen oder eine benutzerdefinierte Operation verwendenURL. Weitere Informationen zu Methoden zur Verwendung von Rollen finden Sie unter Methoden zur Übernahme einer Rolle im IAMBenutzerhandbuch.

IAMRollen mit temporären Anmeldeinformationen sind in den folgenden Situationen nützlich:

- Verbundbenutzerzugriff – Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie im IAMBenutzerhandbuch unter Erstellen einer Rolle für einen externen Identitätsanbieter. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Um zu kontrollieren, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in. IAM Informationen zu Berechtigungssätzen finden Sie unter Berechtigungssätze im AWS IAM Identity Center -Benutzerhandbuch.

- Temporäre IAM Benutzerberechtigungen — Ein IAM Benutzer oder eine Rolle kann eine IAM Rolle übernehmen, um vorübergehend verschiedene Berechtigungen für eine bestimmte Aufgabe zu übernehmen.
- Kontoübergreifender Zugriff — Sie können eine IAM Rolle verwenden, um jemandem (einem vertrauenswürdigen Principal) in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. Bei einigen können Sie AWS-Services jedoch eine Richtlinie direkt an eine Ressource anhängen (anstatt eine Rolle als Proxy zu verwenden). Informationen zum Unterschied zwischen Rollen und ressourcenbasierten Richtlinien für den kontenübergreifenden Zugriff finden Sie IAMim Benutzerhandbuch unter Kontoübergreifender Ressourcenzugriff. IAM
- Serviceübergreifender Zugriff — Einige AWS-Services verwenden Funktionen in anderen. AWS-Services Wenn Sie beispielsweise in einem Service einen Anruf tätigen, ist es üblich, dass dieser Service Anwendungen in Amazon ausführt EC2 oder Objekte in Amazon S3 speichert. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
  - Zugriffssitzungen weiterleiten (FAS) — Wenn Sie einen IAM Benutzer oder eine Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FASverwendet die Berechtigungen des Prinzipals, der an aufruft AWS-Service, kombiniert mit der Anforderung, Anfragen AWS-Service an nachgelagerte Dienste zu stellen. FASAnfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren Abschluss Interaktionen mit anderen AWS-Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien beim Stellen von FAS Anfragen finden Sie unter Zugriffssitzungen weiterleiten.
  - Servicerolle — Eine Servicerolle ist eine IAMRolle, die ein Dienst übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM Administrator kann eine Servicerolle von innen heraus erstellen, ändern und löschenIAM. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter Erstellen einer Rolle zum Delegieren von Berechtigungen AWS-Service an eine.
  - Dienstbezogene Rolle — Eine dienstverknüpfte Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Servicebezogene Rollen erscheinen in Ihrem Dienst AWS-Konto und gehören dem Dienst. Ein IAM Administrator kann die Berechtigungen für dienstbezogene Rollen anzeigen, aber nicht bearbeiten.
- Auf Amazon ausgeführte Anwendungen EC2 — Sie können eine IAM Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2 Instance

ausgeführt werden und AWS API Anfragen stellen AWS CLI . Dies ist dem Speichern von Zugriffsschlüsseln innerhalb der EC2 Instance vorzuziehen. Um einer EC2 Instanz eine AWS Rolle zuzuweisen und sie allen ihren Anwendungen zur Verfügung zu stellen, erstellen Sie ein Instanzprofil, das an die Instanz angehängt ist. Ein Instanzprofil enthält die Rolle und ermöglicht Programmen, die auf der EC2 Instanz ausgeführt werden, temporäre Anmeldeinformationen abzurufen. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter Verwenden einer IAM Rolle zur Erteilung von Berechtigungen für Anwendungen, die auf EC2 Amazon-Instances ausgeführt werden.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese mit AWS Identitäten oder Ressourcen verknüpfen. Eine Richtlinie ist ein Objekt, AWS das, wenn es einer Identität oder Ressource zugeordnet ist, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anfrage stellt. Berechtigungen in den Richtlinien bestimmen, ob die Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden in AWS Form von JSON Dokumenten gespeichert. Weitere Informationen zur Struktur und zum Inhalt von JSON Richtliniendokumenten finden Sie im IAMBenutzerhandbuch unter Überblick über JSON Richtlinien.

Administratoren können mithilfe von AWS JSON Richtlinien festlegen, wer Zugriff auf was hat. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Um Benutzern die Erlaubnis zu erteilen, Aktionen mit den Ressourcen durchzuführen, die sie benötigen, kann ein IAM Administrator IAM Richtlinien erstellen. Der Administrator kann dann die IAM Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen übernehmen.

IAMRichtlinien definieren Berechtigungen für eine Aktion, unabhängig von der Methode, mit der Sie den Vorgang ausführen. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Rolleninformationen aus dem AWS Management Console AWS CLI, dem oder dem abrufen AWS API.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind Dokumente mit JSON Berechtigungsrichtlinien, die Sie an eine Identität anhängen können, z. B. an einen IAM Benutzer, eine Benutzergruppe oder eine Rolle. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und

unter welchen Bedingungen ausführen können. Informationen zum Erstellen einer identitätsbasierten Richtlinie finden Sie im Benutzerhandbuch unter Definieren benutzerdefinierter IAM Berechtigungen mit vom Kunden verwalteten Richtlinien. IAM

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem System zuordnen können. AWS-Konto Zu den verwalteten Richtlinien gehören AWS verwaltete Richtlinien und vom Kunden verwaltete Richtlinien. Informationen zur Auswahl zwischen einer verwalteten Richtlinie oder einer Inline-Richtlinie finden Sie im IAMBenutzerhandbuch unter Wählen Sie zwischen verwalteten Richtlinien und Inline-Richtlinien.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON Richtliniendokumente, die Sie an eine Ressource anhängen. Beispiele für ressourcenbasierte Richtlinien sind IAM Rollenvertrauensrichtlinien und Amazon S3 S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien nicht IAM in einer ressourcenbasierten Richtlinie verwenden.

Zugriffskontrolllisten () ACLs

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLsähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON Richtliniendokumentformat.

Amazon S3 und AWS WAF Amazon VPC sind Beispiele für Dienste, die Unterstützung bietenACLs. Weitere Informationen finden Sie unter Übersicht über ACLs die Zugriffskontrollliste (ACL) im Amazon Simple Storage Service Developer Guide.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger verbreitete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen — Eine Berechtigungsgrenze ist eine erweiterte Funktion, mit der Sie die maximalen Berechtigungen festlegen, die eine identitätsbasierte Richtlinie einer IAM Entität (IAMBenutzer oder Rolle) gewähren kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen zu Berechtigungsgrenzen finden Sie im IAMBenutzerhandbuch unter Berechtigungsgrenzen für IAM Entitäten.
- Dienststeuerungsrichtlinien (SCPs) — SCPs sind JSON Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in festlegen AWS Organizations. AWS Organizations ist ein Dienst zur Gruppierung und zentralen Verwaltung mehrerer AWS-Konten Unternehmenseigentümer. Wenn Sie alle Funktionen in einer Organisation aktivieren, können Sie Richtlinien zur Servicesteuerung (SCPs) auf einige oder alle Ihre Konten anwenden. Das SCP schränkt die Berechtigungen für Entitäten in Mitgliedskonten ein, einschließlich der einzelnen Root-Benutzer des AWS-Kontos. Weitere Informationen zu Organizations und SCPs finden Sie unter Richtlinien zur Servicesteuerung im AWS Organizations Benutzerhandbuch.
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter Sitzungsrichtlinien.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie im IAMBenutzerhandbuch unter Bewertungslogik für Richtlinien.

So funktioniert Amazon Timestream for InfluxDB mit IAM

IAMFunktionen, die Sie mit Amazon Timestream für InfluxDB verwenden können

IAMFunktion	Timestream für InfluxDB-Unterstützung
Identitätsbasierte Richtlinien	Ja
Ressourcenbasierte Richtlinien	Nein
Richtlinienaktionen	Ja
Richtlinienressourcen	Ja
Bedingungsschlüssel für die Richtlinie	Nein
ACLs	Nein
ABAC(Tags in Richtlinien)	Ja
Temporäre Anmeldeinformationen	Ja
Hauptberechtigungen	Ja
Servicerollen	Nein
Serviceverknüpfte Rollen	Ja

Einen allgemeinen Überblick darüber, wie Timestream for InfluxDB und andere AWS Dienste mit den meisten IAM Funktionen funktionieren, finden Sie IAM im Benutzerhandbuch unter AWS Dienste, die IAM mit funktionieren.

Identitätsbasierte Richtlinien für Timestream for InfluxDB

Unterstützt Richtlinien auf Identitätsbasis: Ja

Identitätsbasierte Richtlinien sind Dokumente mit JSON Berechtigungsrichtlinien, die Sie an eine Identität anhängen können, z. B. an einen Benutzer, eine IAM Benutzergruppe oder eine Rolle. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen einer identitätsbasierten Richtlinie finden Sie im Benutzerhandbuch unter Definieren benutzerdefinierter IAM Berechtigungen mit vom Kunden verwalteten Richtlinien. IAM

Mit IAM identitätsbasierten Richtlinien können Sie zulässige oder verweigerte Aktionen und Ressourcen sowie die Bedingungen angeben, unter denen Aktionen zulässig oder verweigert werden. Sie können den Prinzipal nicht in einer identitätsbasierten Richtlinie angeben, da er für den Benutzer oder die Rolle gilt, dem er zugeordnet ist. Weitere Informationen zu allen Elementen, die Sie in einer JSON Richtlinie verwenden können, finden Sie im IAMBenutzerhandbuch unter Referenz zu IAM JSON Richtlinienelementen.

Beispiele für identitätsbasierte Richtlinien für Timestream for InfluxDB

Beispiele für identitätsbasierte Richtlinien von Timestream für InfluxDB finden Sie unter.. Beispiele für identitätsbasierte Richtlinien für Amazon Timestream for InfluxDB

Ressourcenbasierte Richtlinien in Timestream für InfluxDB

Unterstützt ressourcenbasierte Richtlinien: Nein

Ressourcenbasierte Richtlinien sind JSON Richtliniendokumente, die Sie an eine Ressource anhängen. Beispiele für ressourcenbasierte Richtlinien sind IAM Rollenvertrauensrichtlinien und Amazon S3 S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Um den kontoübergreifenden Zugriff zu ermöglichen, können Sie in einer ressourcenbasierten Richtlinie ein ganzes Konto oder IAM Entitäten in einem anderen Konto als Prinzipal angeben. Durch das Hinzufügen eines kontoübergreifenden Auftraggebers zu einer ressourcenbasierten Richtlinie ist nur die halbe Vertrauensbeziehung eingerichtet. Wenn sich der Prinzipal und die Ressource unterscheiden AWS-Konten, muss ein IAM Administrator des vertrauenswürdigen Kontos auch der Prinzipalentität (Benutzer oder Rolle) die Berechtigung zum Zugriff auf die Ressource gewähren. Sie erteilen Berechtigungen, indem Sie der juristischen Stelle eine identitätsbasierte Richtlinie anfügen. Wenn jedoch eine ressourcenbasierte Richtlinie Zugriff auf einen Prinzipal in demselben Konto gewährt, ist keine zusätzliche identitätsbasierte Richtlinie erforderlich. Weitere Informationen finden Sie IAMim IAMBenutzerhandbuch unter Kontoübergreifender Ressourcenzugriff.

Richtlinienaktionen für Timestream for InfluxDB

Unterstützt Richtlinienaktionen: Ja

Administratoren können mithilfe von AWS JSON Richtlinien angeben, wer Zugriff auf was hat. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das `Action` Element einer JSON Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Richtlinienaktionen haben normalerweise denselben Namen wie der zugehörige AWS API Vorgang. Es gibt einige Ausnahmen, z. B. Aktionen, für die nur eine Genehmigung erforderlich ist und für die es keinen entsprechenden Vorgang gibt. API Es gibt auch einige Operationen, die mehrere Aktionen in einer Richtlinie erfordern. Diese zusätzlichen Aktionen werden als abhängige Aktionen bezeichnet.

Schließen Sie Aktionen in eine Richtlinie ein, um Berechtigungen zur Durchführung der zugeordneten Operation zu erteilen.

Eine Liste der Timestream for InfluxDB-Aktionen finden Sie unter Von Amazon Timestream für InfluxDB definierte Aktionen in der Service Authorization Reference.

Richtlinienaktionen in Timestream for InfluxDB verwenden vor der Aktion das folgende Präfix:

```
timestream-influxdb
```

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie mit Kommata:

```
"Action": [
      "timestream-influxdb:action1",
      "timestream-influxdb:action2"
         ]
```

Sie können auch Platzhalter verwenden, um mehrere Aktionen anzugeben. Beispielsweise können Sie alle Aktionen festlegen, die mit dem Wort `Describe` beginnen, einschließlich der folgenden Aktion:

```
"Action": "timestream-influxdb:Describe*"
```

Richtlinienressourcen für Timestream for InfluxDB

Unterstützt Richtlinienressourcen: Ja

Administratoren können mithilfe von AWS JSON Richtlinien angeben, wer Zugriff auf was hat. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das `Resource` JSON Richtlinienelement gibt das Objekt oder die Objekte an, für die die Aktion gilt. Anweisungen müssen entweder ein – `Resource`oder ein `NotResource`-Element enthalten. Es hat sich bewährt, eine Ressource mit ihrem Amazon-Ressourcennamen (ARN) anzugeben. Sie können dies für Aktionen tun, die einen bestimmten Ressourcentyp unterstützen, der als Berechtigungen auf Ressourcenebene bezeichnet wird.

Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, z. B. Auflistungsoperationen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

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

Eine Liste der Timestream for InfluxDB-Ressourcentypen und ihrer ARNs Typen finden Sie unter Von Amazon Timestream für InfluxDB definierte Ressourcen in der Service Authorization Reference. Informationen dazu, mit welchen Aktionen Sie die ARN jeder Ressource angeben können, finden Sie unter Von Amazon Timestream für InfluxDB definierte Aktionen.

Schlüssel für Richtlinienbedingungen für Timestream for InfluxDB

Unterstützt dienstspezifische Richtlinien-Bedingungsschlüssel: Nein

Administratoren können mithilfe von AWS JSON Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Condition` (oder `Condition` block) ermöglicht Ihnen die Angabe der Bedingungen, unter denen eine Anweisung wirksam ist. Das Element `Condition` ist optional. Sie können bedingte Ausdrücke erstellen, die Bedingungsoperatoren verwenden, z. B. ist gleich oder kleiner als, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt.

Wenn Sie mehrere `Condition`-Elemente in einer Anweisung oder mehrere Schlüssel in einem einzelnen `Condition`-Element angeben, wertet AWS diese mittels einer logischen `AND`-Operation aus. Wenn Sie mehrere Werte für einen einzelnen Bedingungsschlüssel angeben, AWS wertet die

Bedingung mithilfe einer logischen `OR` Operation aus. Alle Bedingungen müssen erfüllt werden, bevor die Berechtigungen der Anweisung gewährt werden.

Sie können auch Platzhaltervariablen verwenden, wenn Sie Bedingungen angeben. Sie können einem IAM Benutzer beispielsweise nur dann Zugriff auf eine Ressource gewähren, wenn sie mit seinem IAM Benutzernamen gekennzeichnet ist. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter IAMRichtlinienelemente: Variablen und Tags.

AWS unterstützt globale Bedingungsschlüssel und dienstspezifische Bedingungsschlüssel. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter Kontext-Schlüssel für AWS globale Bedingungen im IAMBenutzerhandbuch.

Zugriffskontrolllisten (ACLs) in Timestream für InfluxDB

Unterstützt: Nein ACLs

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLsähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON Richtliniendokumentformat.

Attributbasierte Zugriffskontrolle () ABAC mit Timestream für InfluxDB

Unterstützt ABAC (Tags in Richtlinien): Ja

Die attributbasierte Zugriffskontrolle (ABAC) ist eine Autorisierungsstrategie, die Berechtigungen auf der Grundlage von Attributen definiert. In werden AWS diese Attribute als Tags bezeichnet. Sie können Tags an IAM Entitäten (Benutzer oder Rollen) und an viele AWS Ressourcen anhängen. Das Markieren von Entitäten und Ressourcen ist der erste Schritt vonABAC. Anschließend entwerfen Sie ABAC Richtlinien, die Operationen zulassen, wenn das Tag des Prinzipals mit dem Tag auf der Ressource übereinstimmt, auf die er zugreifen möchte.

ABACist hilfreich in Umgebungen, die schnell wachsen, und hilft in Situationen, in denen die Richtlinienverwaltung umständlich wird.

Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungselement einer Richtlinie Tag-Informationen an, indem Sie die Schlüssel `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, oder Bedingung `aws:TagKeys` verwenden.

Wenn ein Service alle drei Bedingungsschlüssel für jeden Ressourcentyp unterstützt, lautet der Wert für den Service Ja. Wenn ein Service alle drei Bedingungsschlüssel für nur einige Ressourcentypen unterstützt, lautet der Wert Teilweise.

Weitere Informationen dazu finden Sie ABAC unter Definieren von Berechtigungen mit ABAC Autorisierung im IAMBenutzerhandbuch. Ein Tutorial mit Schritten zur Einrichtung finden Sie im ABAC Benutzerhandbuch unter Verwenden der attributebasierten Zugriffskontrolle (ABAC). IAM

Temporäre Anmeldeinformationen mit Timestream für InfluxDB verwenden

Unterstützt temporäre Anmeldeinformationen: Ja

Einige funktionieren AWS-Services nicht, wenn Sie sich mit temporären Anmeldeinformationen anmelden. Weitere Informationen, einschließlich Informationen darüber, AWS-Services wie Sie mit temporären Anmeldeinformationen arbeiten können AWS-Services , finden Sie IAM im IAMBenutzerhandbuch.

Sie verwenden temporäre Anmeldeinformationen, wenn Sie sich mit einer anderen AWS Management Console Methode als einem Benutzernamen und einem Kennwort anmelden. Wenn Sie beispielsweise AWS über den Single Sign-On-Link (SSO) Ihres Unternehmens darauf zugreifen, werden bei diesem Vorgang automatisch temporäre Anmeldeinformationen erstellt. Sie erstellen auch automatisch temporäre Anmeldeinformationen, wenn Sie sich als Benutzer bei der Konsole anmelden und dann die Rollen wechseln. Weitere Informationen zum Rollenwechsel finden Sie im Benutzerhandbuch unter Von einem Benutzer zu einer IAM Rolle (Konsole) wechseln. IAM

Mit dem AWS CLI oder können Sie manuell temporäre Anmeldeinformationen erstellen AWS API. Sie können diese temporären Anmeldeinformationen dann für den Zugriff verwenden AWS. AWS empfiehlt, temporäre Anmeldeinformationen dynamisch zu generieren, anstatt langfristige Zugriffsschlüssel zu verwenden. Weitere Informationen finden Sie unter Temporäre Sicherheitsanmeldeinformationen unter IAM.

Serviceübergreifende Prinzipalberechtigungen für Timestream for InfluxDB

Unterstützt Forward-Access-Sitzungen (): Ja FAS

Wenn Sie einen IAM Benutzer oder eine Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FASverwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, kombiniert mit der Anforderung, Anfragen AWS-Service an nachgelagerte Dienste zu stellen. FASAnfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren Abschluss Interaktionen mit anderen AWS-Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien beim Stellen von FAS Anfragen finden Sie unter Zugriffssitzungen weiterleiten.

Servicerollen für Timestream for InfluxDB

Unterstützt Servicerollen: Nein

Eine Servicerolle ist eine IAMRolle, die ein Dienst übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM Administrator kann eine Servicerolle von innen heraus erstellen, ändern und löschenIAM. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter Erstellen einer Rolle zum Delegieren von Berechtigungen AWS-Service an eine.

> **Warning**
>
> Das Ändern der Berechtigungen für eine Servicerolle kann die Funktionalität von Timestream for InfluxDB beeinträchtigen. Bearbeiten Sie Servicerollen nur, wenn Timestream for InfluxDB dazu eine Anleitung bietet.

Mit Diensten verknüpfte Rollen für Timestream for InfluxDB

Unterstützt serviceverknüpfte Rollen: Ja

Eine serviceverknüpfte Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Dienstbezogene Rollen werden in Ihrem Dienst angezeigt AWS-Konto und gehören dem Dienst. Ein IAM Administrator kann die Berechtigungen für dienstbezogene Rollen anzeigen, aber nicht bearbeiten.

Einzelheiten zum Erstellen oder Verwalten von dienstbezogenen Rollen finden Sie unter AWS Dienste, die mit funktionieren. IAM Suchen Sie in der Tabelle nach einem Service mit einem `Yes` in der Spalte Service-linked role (Serviceverknüpfte Rolle). Wählen Sie den Link Yes (Ja) aus, um die Dokumentation für die serviceverknüpfte Rolle für diesen Service anzuzeigen.

Beispiele für identitätsbasierte Richtlinien für Amazon Timestream for InfluxDB

Standardmäßig sind Benutzer und Rollen nicht berechtigt, Timestream für InfluxDB-Ressourcen zu erstellen oder zu ändern. Sie können auch keine Aufgaben mit AWS Management Console, AWS Command Line Interface (AWS CLI) oder ausführen. AWS API Um Benutzern die Berechtigung zu erteilen, Aktionen mit den Ressourcen durchzuführen, die sie benötigen, kann ein IAM Administrator IAM Richtlinien erstellen. Der Administrator kann dann die IAM Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen übernehmen.

Informationen zum Erstellen einer IAM identitätsbasierten Richtlinie mithilfe dieser Beispieldokumente zu JSON Richtlinien finden Sie im IAMBenutzerhandbuch unter IAM Richtlinien erstellen (Konsole).

Einzelheiten zu den von Timestream für InfluxDB definierten Aktionen und Ressourcentypen, einschließlich des Formats ARNs für jeden der Ressourcentypen, finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon Timestream for InfluxDB in der Service Authorization Reference.

Themen

- Bewährte Methoden für Richtlinien
- Verwenden der Timestream for InfluxDB-Konsole
- Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer
- Zugreifen auf einen Amazon-S3-Bucket
- Alle Operationen zulassen
- Eine DB-Instance erstellen, beschreiben, löschen und aktualisieren

Bewährte Methoden für Richtlinien

Identitätsbasierte Richtlinien legen fest, ob jemand Timestream for InfluxDB-Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder diese löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Befolgen Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Anleitungen und Empfehlungen:

- Beginnen Sie mit AWS verwalteten Richtlinien und wechseln Sie zu Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um damit zu beginnen, Ihren Benutzern und Workloads Berechtigungen zu gewähren. Sie sind in Ihrem verfügbar. AWS-Konto Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie AWS im IAMBenutzerhandbuch unter AWS Verwaltete Richtlinien oder Verwaltete Richtlinien für Jobfunktionen.
- Berechtigungen mit den geringsten Rechten anwenden — Wenn Sie Berechtigungen mit IAM Richtlinien festlegen, gewähren Sie nur die Berechtigungen, die für die Ausführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten

Berechtigungen. Weitere Informationen zur Verwendung IAM zum Anwenden von Berechtigungen finden Sie IAMim Benutzerhandbuch unter Richtlinien und Berechtigungen. IAM

- Verwenden Sie Bedingungen in IAM Richtlinien, um den Zugriff weiter einzuschränken — Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen einzuschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um anzugeben, dass alle Anfragen über gesendet werden müssenSSL. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese über einen bestimmten Zweck verwendet werden AWS-Service, z. AWS CloudFormation B. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter IAMJSONRichtlinienelemente: Bedingung.
- Verwenden Sie IAM Access Analyzer, um Ihre IAM Richtlinien zu validieren, um sichere und funktionale Berechtigungen zu gewährleisten. IAM Access Analyzer validiert neue und bestehende Richtlinien, sodass die Richtlinien der IAM Richtliniensprache (JSON) und den IAM bewährten Methoden entsprechen. IAMAccess Analyzer bietet mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen, um Sie bei der Erstellung sicherer und funktionaler Richtlinien zu unterstützen. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter Überprüfen von Richtlinien mit IAM Access Analyzer.
- Multi-Faktor-Authentifizierung erforderlich (MFA) — Wenn Sie ein Szenario haben, in dem IAM Benutzer oder ein Root-Benutzer erforderlich sind AWS-Konto, aktivieren Sie die Option MFA für zusätzliche Sicherheit. Um festzulegen, MFA wann API Operationen aufgerufen werden, fügen Sie MFA Bedingungen zu Ihren Richtlinien hinzu. Weitere Informationen finden Sie unter Sicherer API Zugriff mit MFA im IAMBenutzerhandbuch.

Weitere Informationen zu bewährten Methoden finden Sie unter Bewährte Sicherheitsmethoden IAM im IAM Benutzerhandbuch. IAM

Verwenden der Timestream for InfluxDB-Konsole

Um auf die Amazon Timestream for InfluxDB-Konsole zugreifen zu können, benötigen Sie einen Mindestsatz an Berechtigungen. Diese Berechtigungen müssen es Ihnen ermöglichen, Details zu den Timestream for InfluxDB-Ressourcen in Ihrem aufzulisten und anzuzeigen. AWS-Konto Wenn Sie eine identitätsbasierte Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Entitäten (Benutzer oder Rollen) mit dieser Richtlinie.

Sie müssen Benutzern, die nur Anrufe an oder am tätigen, keine Mindestberechtigungen für die AWS CLI Konsole gewähren. AWS API Erlauben Sie stattdessen nur den Zugriff auf die Aktionen, die dem API Vorgang entsprechen, den sie ausführen möchten.

Um sicherzustellen, dass Benutzer und Rollen die Timestream for InfluxDB-Konsole weiterhin verwenden können, fügen Sie den Entitäten auch den Timestream für InfluxDB `ConsoleAccess` oder `ReadOnly` AWS die verwaltete Richtlinie hinzu. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen für einen Benutzer im Benutzerhandbuch. IAM

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die es IAM Benutzern ermöglicht, die internen und verwalteten Richtlinien einzusehen, die mit ihrer Benutzeridentität verknüpft sind. Diese Richtlinie umfasst Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder programmgesteuert mithilfe von oder. 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": "*"
        }
    ]
```

```
}
```

Zugreifen auf einen Amazon-S3-Bucket

In diesem Beispiel möchten Sie einem IAM Benutzer in Ihrem AWS Konto Zugriff auf einen Ihrer Amazon S3 S3-Buckets gewähren. `examplebucket` Sie möchten dem Benutzer außerdem Berechtigungen zum Hinzufügen, Aktualisieren und Löschen von Objekten gewähren.

Zusätzlich zum Erteilen der Berechtigungen `s3:PutObject`, `s3:GetObject` und `s3:DeleteObject` für den Benutzer, gewährt die Richtlinie die Berechtigungen `s3:ListAllMyBuckets`, `s3:GetBucketLocation` und `s3:ListBucket`. Dies sind die zusätzlichen Berechtigungen, die von der Konsole benötigt werden. Außerdem sind die Aktionen `s3:PutObjectAcl` und `s3:GetObjectAcl` erforderlich, um Objekte in der Konsole kopieren, ausschneiden und einfügen zu können. Ein Beispiel für eine exemplarische Vorgehensweise, bei der Benutzern Berechtigungen erteilt und diese mithilfe der Konsole getestet werden, finden Sie unter Eine exemplarische Vorgehensweise: Verwenden von Benutzerrichtlinien zur Steuerung des Zugriffs auf Ihren Bucket.

```
{
   "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/*"
      }
   ]
}
```

Alle Operationen zulassen

Im Folgenden finden Sie eine Beispielrichtlinie, die alle Operationen in Timestream for InfluxDB zulässt.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream-influxdb:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Eine DB-Instance erstellen, beschreiben, löschen und aktualisieren

Die folgende Beispielrichtlinie ermöglicht es einem Benutzer, eine DB-Instance zu erstellen, zu beschreiben, zu löschen und zu aktualisieren`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"
        }
    ]
}
```

Fehlerbehebung bei Amazon Timestream für InfluxDB-Identität und Zugriff

Verwenden Sie die folgenden Informationen, um häufig auftretende Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit Timestream for InfluxDB und auftreten können. IAM

Themen

- Ich bin nicht berechtigt, eine Aktion in Timestream for InfluxDB durchzuführen
- Ich möchte Personen außerhalb meines AWS Kontos den Zugriff auf meinen Timestream für InfluxDB-Ressourcen ermöglichen

Ich bin nicht berechtigt, eine Aktion in Timestream for InfluxDB durchzuführen

Wenn Ihnen AWS Management Console mitgeteilt wird, dass Sie nicht berechtigt sind, eine Aktion auszuführen, müssen Sie sich an Ihren Administrator wenden, um Unterstützung zu erhalten. Ihr Administrator ist die Person, die Ihnen Ihren Benutzernamen und Ihr Passwort bereitgestellt hat.

Der folgende Beispielfehler tritt auf, wenn der `mateojackson`-Benutzer versucht, die Konsole zum Anzeigen von Details zu einer fiktiven *`my-example-widget`*-Ressource zu verwenden, jedoch nicht über `timestream-influxdb:`*`GetWidget`*-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 timestream-influxdb:GetWidget on resource: my-example-widget
```

In diesem Fall bittet Mateo seinen Administrator um die Aktualisierung seiner Richtlinien, um unter Verwendung der Aktion *`my-example-widget`* auf die Ressource `timestream-influxdb:`*`GetWidget`* zugreifen zu können.

Ich möchte Personen außerhalb meines AWS Kontos den Zugriff auf meinen Timestream für InfluxDB-Ressourcen ermöglichen

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Für Dienste, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (ACLs) unterstützen, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Steuern des Zugriffs auf eine DB-Instance in einem VPC
- Informationen darüber, ob Timestream for InfluxDB diese Funktionen unterstützt, finden Sie unter So funktioniert Amazon Timestream for InfluxDB mit. IAM
- Informationen dazu, wie Sie den Zugriff auf Ihre Ressourcen über AWS Konten hinweg gewähren können, die Sie besitzen, finden Sie im Benutzerhandbuch unter Gewähren des Zugriffs für einen IAM Benutzer in einem anderen AWS Konto, das Sie besitzen. IAM
- Informationen dazu, wie Sie AWS Konten von Drittanbietern Zugriff auf Ihre Ressourcen gewähren können, finden Sie im IAMBenutzerhandbuch unter Gewähren des Zugriffs auf AWS Konten Dritter.
- Informationen dazu, wie Sie Zugriff über einen Identitätsverbund gewähren, finden Sie im Benutzerhandbuch unter Zugriff für extern authentifizierte Benutzer (Identitätsverbund). IAM
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie im Benutzerhandbuch unter Unterschiede zwischen IAM Rollen und ressourcenbasierten Richtlinien. IAM

Steuern des Zugriffs auf eine DB-Instance in einem VPC

Mit Amazon Virtual Private Cloud (AmazonVPC) können Sie AWS Ressourcen wie Amazon Timestream für InfluxDB-DB-Instances in einer virtuellen privaten Cloud () starten. VPC Wenn Sie Amazon verwendenVPC, haben Sie die Kontrolle über Ihre virtuelle Netzwerkumgebung. Sie können Ihren eigenen IP-Adressbereich auswählen, Subnetze erstellen sowie Routing-Tabellen und Zugriffskontrolllisten konfigurieren.

Eine VPC Sicherheitsgruppe kontrolliert den Zugriff auf DB-Instances innerhalb einerVPC. Jede VPC Sicherheitsgruppenregel ermöglicht einer bestimmten Quelle den Zugriff auf eine DB-Instance in einerVPC, die dieser VPC Sicherheitsgruppe zugeordnet ist. Die Quelle kann ein Adressbereich (z. B. 203.0.113.0/24) oder eine andere Sicherheitsgruppe sein. VPC Wenn Sie

eine VPC Sicherheitsgruppe als Quelle angeben, lassen Sie eingehenden Datenverkehr von allen Instanzen (normalerweise Anwendungsservern) zu, die die Quellsicherheitsgruppe verwenden. VPC Bevor Sie versuchen, eine Verbindung zu Ihrer DB-Instance herzustellen, konfigurieren Sie Ihre VPC für Ihren Anwendungsfall. Im Folgenden finden Sie gängige Szenarien für den Zugriff auf eine DB-Instance in einemVPC:

Eine DB-Instance in einer, auf die von einer EC2 Amazon-Instance VPC zugegriffen wird, in derselben VPC

Eine übliche Verwendung einer DB-Instance in einer VPC ist die gemeinsame Nutzung von Daten mit einem Anwendungsserver, der in einer EC2 Instance derselben ausgeführt wirdVPC. Die EC2 Instance kann einen Webserver mit einer Anwendung ausführen, die mit der DB-Instance interagiert.

Eine DB-Instance in einer, auf die von einer EC2 Instance in einer anderen VPC zugegriffen wird VPC

In einigen Fällen befindet sich Ihre DB-Instance in einer anderen VPC EC2 Instanz als die Instance, mit der Sie darauf zugreifen. Wenn ja, können Sie VPC Peering verwenden, um auf die DB-Instance zuzugreifen.

Eine DB-Instance in einer Anwendung, VPC auf die ein Client über das Internet zugreift

Um über das Internet auf eine VPC DB-Instance in einer Client-Anwendung zuzugreifen, konfigurieren Sie eine VPC mit einem einzigen öffentlichen Subnetz und verwenden die öffentlichen Subnetze, um die DB-Instance zu erstellen. Sie konfigurieren in der auch ein Internet-GatewayVPC, um die Kommunikation über das Internet zu ermöglichen. Um von außerhalb eine Verbindung zu einer DB-Instance herzustellenVPC, muss die DB-Instance öffentlich zugänglich sein. Außerdem muss der Zugriff unter Verwendung der eingehenden Regeln der Sicherheitsgruppe der DB-Instance gewährt werden, und andere Anforderungen müssen erfüllt sein.

Weitere Informationen zu VPC Sicherheitsgruppen finden Sie unter Sicherheitsgruppen im Amazon Virtual Private Cloud Cloud-Benutzerhandbuch.

Einzelheiten zum Herstellen einer Verbindung zu einer Timestream for InfluxDB-DB-Instance finden Sie unter. Verbindung zu einer Amazon Timestream for InfluxDB-DB-Instance herstellen

Sicherheitsgruppenszenario

Eine übliche Verwendung einer DB-Instance in einer VPC ist die gemeinsame Nutzung von Daten mit einem Anwendungsserver, der in einer EC2 Amazon-Instance in derselben ausgeführt wirdVPC, auf die von einer Client-Anwendung außerhalb von zugegriffen wirdVPC. Für dieses Szenario verwenden Sie den Timestream für InfluxDB und VPC Seiten auf der AWS Management Console oder den Timestream für InfluxDB und EC2 API Operationen, um die erforderlichen Instances und Sicherheitsgruppen zu erstellen:

1. Erstellen Sie eine VPC Sicherheitsgruppe (zum Beispiel`sg-0123ec2example`) und definieren Sie Regeln für eingehenden Datenverkehr, die die IP-Adressen der Client-Anwendung als Quelle verwenden. Diese Sicherheitsgruppe ermöglicht es Ihrer Client-Anwendung, eine Verbindung zu EC2 Instances in einer Instanz herzustellenVPC, die diese Sicherheitsgruppe verwendet.
2. Erstellen Sie eine EC2 Instanz für die Anwendung und fügen Sie die EC2 Instanz der VPC Sicherheitsgruppe (`sg-0123ec2example`) hinzu, die Sie im vorherigen Schritt erstellt haben.
3. Erstellen Sie eine zweite VPC Sicherheitsgruppe (z. B.`sg-6789rdsexample`) und erstellen Sie eine neue Regel, indem Sie die VPC Sicherheitsgruppe, die Sie in Schritt 1 (`sg-0123ec2example`) erstellt haben, als Quelle angeben.
4. Erstellen Sie eine neue DB-Instance und fügen Sie die DB-Instance der VPC Sicherheitsgruppe (`sg-6789rdsexample`) hinzu, die Sie im vorherigen Schritt erstellt haben. Verwenden Sie beim Erstellen der DB dieselbe Portnummer wie die, die Sie für die VPC Sicherheitsgruppenregel (`sg-6789rdsexample`) angegeben haben, die Sie in Schritt 3 erstellt haben.

Eine VPC Sicherheitsgruppe erstellen

Sie können mithilfe der VPC Konsole eine VPC Sicherheitsgruppe für eine DB-Instance erstellen. Informationen zum Erstellen einer Sicherheitsgruppe finden Sie unter Sicherheitsgruppen im Amazon Virtual Private Cloud Cloud-Benutzerhandbuch.

Verknüpfen einer Sicherheitsgruppe mit einer DB-Instance

Sie können einer DB-Instance eine Sicherheitsgruppe zuordnen, indem Sie Update auf der Timestream for InfluxDB-Konsole, `UpdateDBInstance` Timestream for API InfluxDB oder den Befehl verwenden. `update-db-instance` AWS CLI

Das folgende CLI Beispiel ordnet eine bestimmte VPC Sicherheitsgruppe zu und entfernt DB-Sicherheitsgruppen aus der DB-Instance

```
aws timestream-influxdb update-db-instance --identifier dbName --vpc-security-group-
ids sg-ID
```

Informationen zum Ändern einer DB-Instance finden Sie unter Aktualisieren von DB-Instances.

Verwenden von serviceverknüpften Rollen für Amazon Timestream for InfluxDB

Amazon Timestream for InfluxDB verwendet AWS Identity and Access Management (IAM) service-verknüpfte Rollen. Eine serviceverknüpfte Rolle ist ein einzigartiger IAM Rollentyp, der direkt mit einem AWS Service verknüpft ist, z. B. Amazon Timestream für InfluxDB. Servicebezogene Rollen von Amazon Timestream for InfluxDB sind von Amazon Timestream for InfluxDB vordefiniert. Sie beinhalten alle Berechtigungen, die der Service benötigt, um Dienste im Namen Ihrer DB-Instances aufzurufen AWS .

Eine serviceverknüpfte Rolle erleichtert die Einrichtung von Amazon Timestream für InfluxDB, da Sie die erforderlichen Berechtigungen nicht manuell hinzufügen müssen. Die Rollen sind bereits in Ihrem AWS Konto vorhanden, sind jedoch mit Amazon Timestream für InfluxDB-Anwendungsfälle verknüpft und haben vordefinierte Berechtigungen. Nur Amazon Timestream for InfluxDB kann diese Rollen übernehmen, und nur diese Rollen können die vordefinierte Berechtigungsrichtlinie verwenden. Sie können die Rollen nur nach dem Löschen der zugehörigen Ressourcen löschen. Dies schützt Ihre Amazon Timestream for InfluxDB-Ressourcen, da Sie nicht versehentlich die erforderlichen Berechtigungen für den Zugriff auf die Ressourcen entfernen können.

Informationen zu anderen Diensten, die serviceverknüpfte Rollen unterstützen, finden Sie unter AWS Services That Work with IAM und suchen Sie in der Spalte Service-Linked Role nach den Diensten, für die Ja steht. Wählen Sie über einen Link Ja aus, um die Dokumentation zu einer serviceverknüpften Rolle für diesen Service anzuzeigen.

Inhalt

- Löschen einer dienstverknüpften Rolle (IAMKonsole)
- Löschen einer dienstverknüpften Rolle () IAM CLI
- Löschen einer serviceverknüpften Rolle () IAM API
- Unterstützte Regionen für Amazon Timestream for InfluxDB Service-Linked Roles

Servicebezogene Rollenberechtigungen für Amazon Timestream for InfluxDB

Amazon Timestream for InfluxDB verwendet die serviceverknüpfte Rolle mit dem Namen AmazonTimestreamInfluxDBServiceRolePolicy— Diese Richtlinie ermöglicht es Timestream for InfluxDB, AWS Ressourcen in Ihrem Namen zu verwalten, soweit dies für die Verwaltung Ihrer Cluster erforderlich ist.

Die Richtlinie für AmazonTimestreamInflux DBServiceRolePolicy servicebezogene Rollenberechtigungen ermöglicht es Amazon Timestream for InfluxDB, die folgenden Aktionen für die angegebenen Ressourcen durchzuführen:

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

Um es einer IAM Entität zu ermöglichen, servicebezogene Rollen zu erstellen AmazonTimestreamInflux DBServiceRolePolicy

Fügen Sie den Berechtigungen für diese IAM Entität die folgende Richtlinienerklärung hinzu:

```
{
```

```
    "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"}}
}
```

Um einer IAM Entität das Löschen von AmazonTimestreamInflux DBServiceRolePolicy dienstbezogenen Rollen zu ermöglichen

Fügen Sie den Berechtigungen für diese IAM Entität die folgende Richtlinienerklärung hinzu:

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

Alternativ können Sie eine AWS verwaltete Richtlinie verwenden, um vollen Zugriff auf Amazon Timestream for InfluxDB zu gewähren.

Eine serviceverknüpfte Rolle erstellen () IAM

Sie müssen eine serviceverknüpfte Rolle nicht manuell erstellen. Wenn Sie eine DB-Instance erstellen, erstellt Amazon Timestream for InfluxDB die serviceverknüpfte Rolle für Sie.

Wenn Sie diese serviceverknüpfte Rolle löschen und sie dann erneut erstellen müssen, können Sie dasselbe Verfahren anwenden, um die Rolle in Ihrem Konto neu anzulegen. Wenn Sie eine DB-Instance erstellen, erstellt Amazon Timestream for InfluxDB die serviceverknüpfte Rolle erneut für Sie.

Bearbeiten der Beschreibung einer serviceverknüpften Rolle für Amazon Timestream for InfluxDB

Amazon Timestream for InfluxDB erlaubt Ihnen nicht, die serviceverknüpfte Rolle zu bearbeiten. AmazonTimestreamInflux DBServiceRolePolicy Da möglicherweise verschiedene Entitäten auf die Rolle verweisen, kann der Rollenname nach dem Erstellen einer serviceverknüpften Rolle nicht mehr geändert werden. Sie können die Beschreibung der Rolle jedoch mithilfe von bearbeiten. IAM

Bearbeiten einer servicebezogenen Rollenbeschreibung (IAMKonsole)

Sie können die IAM Konsole verwenden, um eine Beschreibung einer dienstbezogenen Rolle zu bearbeiten.

So bearbeiten Sie die Beschreibung einer serviceverknüpften Rolle (Konsole)

1. Wählen Sie im linken Navigationsbereich der IAM Konsole die Option Rollen aus.
2. Wählen Sie den Namen der zu ändernden Rolle.
3. Wählen Sie neben Role description ganz rechts Edit.
4. Geben Sie eine neue Beschreibung im Dialogfeld ein und klicken Sie auf Save (Speichern).

Eine Beschreibung einer serviceverknüpften Rolle bearbeiten () IAM CLI

Sie können IAM Operationen von verwenden AWS Command Line Interface , um eine servicebezogene Rollenbeschreibung zu bearbeiten.

Um die Beschreibung einer dienstbezogenen Rolle zu ändern () CLI

1. (Optional) Um die aktuelle Beschreibung einer Rolle anzuzeigen, verwenden Sie den IAM Vorgang AWS CLI `get-role` for.

   Example

   ```
   $ aws iam get-role --role-name AmazonTimestreamInfluxDBServiceRolePolicy
   ```

   Verwenden Sie den Rollennamen, nicht denARN, um auf Rollen mit den CLI Operationen zu verweisen. Wenn eine Rolle beispielsweise Folgendes hatARN:`arn:aws:iam::123456789012:role/myrole`, bezeichnen Sie die Rolle als**`myrole`**.
2. Um die Beschreibung einer serviceverknüpften Rolle zu aktualisieren, verwenden Sie den IAM Vorgang AWS CLI `update-role-description` for.

Linux und macOS

```
$ aws iam update-role-description \
    --role-name AmazonTimestreamInfluxDBServiceRolePolicy \
    --description "new description"
```

Windows

```
$ aws iam update-role-description ^
    --role-name AmazonTimestreamInfluxDBServiceRolePolicy ^
    --description "new description"
```

Bearbeiten einer mit einem Dienst verknüpften Rollenbeschreibung () IAM API

Sie können die verwenden IAMAPI, um eine mit einem Dienst verknüpfte Rollenbeschreibung zu bearbeiten.

Um die Beschreibung einer dienstbezogenen Rolle zu ändern () API

1. (Optional) Um die aktuelle Beschreibung einer Rolle anzuzeigen, verwenden Sie den Vorgang IAM API GetRole.

   Example

   ```
   https://iam.amazonaws.com/
      ?Action=GetRole
      &RoleName=AmazonTimestreamInfluxDBServiceRolePolicy
      &Version=2010-05-08
      &AUTHPARAMS
   ```

2. Verwenden Sie den IAM API Vorgang, um die Beschreibung einer Rolle zu aktualisieren UpdateRoleDescription.

   Example

   ```
   https://iam.amazonaws.com/
      ?Action=UpdateRoleDescription
      &RoleName=AmazonTimestreamInfluxDBServiceRolePolicy
      &Version=2010-05-08
   ```

```
    &Description="New description"
```

Löschen einer serviceverknüpften Rolle für Amazon Timestream for InfluxDB

Wenn Sie ein Feature oder einen Dienst, die bzw. der eine serviceverknüpften Rolle erfordert, nicht mehr benötigen, sollten Sie diese Rolle löschen. Auf diese Weise haben Sie keine ungenutzte juristische Stelle, die nicht aktiv überwacht oder verwaltet wird. Sie müssen jedoch Ihre serviceverknüpfte Rolle zunächst bereinigen, bevor Sie sie löschen können.

Amazon Timestream for InfluxDB löscht die serviceverknüpfte Rolle nicht für Sie.

Bereinigen einer serviceverknüpften Rolle

Bevor Sie eine serviceverknüpfte Rolle löschen können, stellen Sie zunächst sicher, dass der Rolle keine Ressourcen (Cluster) zugeordnet sind. IAM

Um zu überprüfen, ob die dienstverknüpfte Rolle über eine aktive Sitzung in der Konsole verfügt IAM

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im linken Navigationsbereich der IAM Konsole Rollen aus. Wählen Sie dann den Namen (nicht das Kontrollkästchen) der AmazonTimestreamInflux DBServiceRolePolicy Rolle aus.
3. Wählen Sie auf der Seite Summary für die ausgewählte Rolle die Registerkarte Access Advisor.
4. Überprüfen Sie auf der Registerkarte Access Advisor die jüngsten Aktivitäten für die serviceverknüpfte Rolle.

Löschen einer dienstverknüpften Rolle (IAMKonsole)

Sie können die IAM Konsole verwenden, um eine dienstverknüpfte Rolle zu löschen.

So löschen Sie eine serviceverknüpfte Rolle (Konsole)

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im linken Navigationsbereich der IAM Konsole Rollen aus. Aktivieren Sie dann das Kontrollkästchen neben dem Rollennamen, den Sie löschen möchten, nicht den Namen oder die Zeile selbst.

3. Wählen Sie für Role actions oben auf der Seite Delete role aus.
4. Überprüfen Sie auf der Bestätigungsseite die Daten, auf die der Dienst zuletzt zugegriffen hat. Aus diesen Daten geht hervor, wann jede der ausgewählten Rollen zuletzt auf einen AWS Dienst zugegriffen hat. Auf diese Weise können Sie leichter bestätigen, ob die Rolle derzeit aktiv ist. Wenn Sie fortfahren möchten, wählen Sie Yes, Delete aus, um die serviceverknüpfte Rolle zur Löschung zu übermitteln.
5. Sehen Sie sich die IAM Konsolenbenachrichtigungen an, um den Fortschritt beim Löschen der dienstbezogenen Rolle zu verfolgen. Da das Löschen der IAM dienstbezogenen Rolle asynchron erfolgt, kann die Löschaufgabe erfolgreich sein oder fehlschlagen, nachdem Sie die Rolle zum Löschen eingereicht haben. Wenn der Vorgang fehlschlägt, können Sie in den Benachrichtigungen View details oder View Resources auswählen, um zu erfahren, warum die Löschung fehlgeschlagen ist.

Löschen einer dienstverknüpften Rolle () IAM CLI

Sie können IAM Operationen von verwenden, AWS Command Line Interface um eine dienstverknüpfte Rolle zu löschen.

Um eine dienstverknüpfte Rolle zu löschen () CLI

1. Wenn Sie den Namen der serviceverknüpften Rolle, die Sie löschen möchten, nicht kennen, geben Sie den folgenden Befehl ein. Dieser Befehl listet die Rollen und ihre Amazon-Ressourcennamen (ARNs) in Ihrem Konto auf.

   ```
   $ aws iam get-role --role-name role-name
   ```

   Verwenden Sie den Rollennamen, nicht denARN, um auf Rollen mit den CLI Vorgängen zu verweisen. Wenn eine Rolle beispielsweise den hat ARN`arn:aws:iam::123456789012:role/myrole`, bezeichnen Sie die Rolle als**`myrole`**.
2. Da eine serviceverknüpfte Rolle nicht gelöscht werden kann, wenn sie verwendet wird oder ihr Ressourcen zugeordnet sind, müssen Sie eine Löschungsanforderung übermitteln. Diese Anforderung kann verweigert werden, wenn diese Bedingungen nicht erfüllt sind. Sie benötigen die `deletion-task-id` aus der Antwort, um den Status der Löschaufgabe zu überprüfen. Geben Sie Folgendes ein, um eine Anforderung zum Löschen einer serviceverknüpften Rolle abzusenden.

```
$ aws iam delete-service-linked-role --role-name role-name
```

3. Geben Sie Folgendes ein, um den Status der Löschaufgabe zu überprüfen.

```
$ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
```

Der Status der Löschaufgabe kann `NOT_STARTED`, `IN_PROGRESS`, `SUCCEEDED` oder `FAILED` lauten. Wenn die Löschung fehlschlägt, gibt der Aufruf den Grund zurück, sodass Sie das Problem beheben können.

Löschen einer serviceverknüpften Rolle () IAM API

Sie können die verwenden IAMAPI, um eine dienstverknüpfte Rolle zu löschen.

Um eine dienstverknüpfte Rolle zu löschen () API

1. Um einen Löschantrag für eine dienstbezogene Rolle einzureichen, rufen Sie DeleteServiceLinkedRole. Geben Sie in der Anfrage einen Rollennamen an.

   Da eine serviceverknüpfte Rolle nicht gelöscht werden kann, wenn sie verwendet wird oder ihr Ressourcen zugeordnet sind, müssen Sie eine Löschungsanforderung übermitteln. Diese Anforderung kann verweigert werden, wenn diese Bedingungen nicht erfüllt sind. Sie benötigen die `DeletionTaskId` aus der Antwort, um den Status der Löschaufgabe zu überprüfen.
2. Um den Status des Löschvorgangs zu überprüfen, rufen Sie GetServiceLinkedRoleDeletionStatus. Geben Sie in der Anfrage die an`DeletionTaskId`.

   Der Status der Löschaufgabe kann `NOT_STARTED`, `IN_PROGRESS`, `SUCCEEDED` oder `FAILED` lauten. Wenn die Löschung fehlschlägt, gibt der Aufruf den Grund zurück, sodass Sie das Problem beheben können.

Unterstützte Regionen für Amazon Timestream for InfluxDB Service-Linked Roles

Amazon Timestream for InfluxDB unterstützt die Verwendung von serviceverknüpften Rollen in allen Regionen, in denen der Service verfügbar ist. Weitere Informationen finden Sie unter AWS -Service-Endpunkte.

AWS verwaltete Richtlinien für Amazon Timestream for InfluxDB

Um Benutzern, Gruppen und Rollen Berechtigungen hinzuzufügen, ist es einfacher, AWS verwaltete Richtlinien zu verwenden, als Richtlinien selbst zu schreiben. Es erfordert Zeit und Fachwissen, um vom IAMKunden verwaltete Richtlinien zu erstellen, die Ihrem Team nur die Berechtigungen gewähren, die es benötigt. Um schnell loszulegen, können Sie unsere AWS verwalteten Richtlinien verwenden. Diese Richtlinien decken allgemeine Anwendungsfälle ab und sind in Ihrem AWS Konto verfügbar. Weitere Informationen zu AWS verwalteten Richtlinien finden Sie im IAMBenutzerhandbuch unter AWS Verwaltete Richtlinien.

AWS Dienste verwalten und aktualisieren AWS verwaltete Richtlinien. Sie können die Berechtigungen in AWS verwalteten Richtlinien nicht ändern. Services fügen einer von AWS verwalteten Richtlinien gelegentlich zusätzliche Berechtigungen hinzu, um neue Features zu unterstützen. Diese Art von Update betrifft alle Identitäten (Benutzer, Gruppen und Rollen), an welche die Richtlinie angehängt ist. Services aktualisieren eine von AWS verwaltete Richtlinie am ehesten, ein neues Feature gestartet wird oder neue Vorgänge verfügbar werden. Dienste entfernen keine Berechtigungen aus einer AWS verwalteten Richtlinie, sodass durch Richtlinienaktualisierungen Ihre bestehenden Berechtigungen nicht beeinträchtigt werden.

AWS Unterstützt außerdem verwaltete Richtlinien für Jobfunktionen, die sich über mehrere Dienste erstrecken. Die ReadOnlyAccess AWS verwaltete Richtlinie bietet beispielsweise schreibgeschützten Zugriff auf alle AWS Dienste und Ressourcen. Wenn ein Dienst eine neue Funktion startet, werden nur Leseberechtigungen für neue Operationen und Ressourcen AWS hinzugefügt. Eine Liste und eine Beschreibung der Richtlinien für Jobfunktionen finden Sie im IAMBenutzerhandbuch unter AWS Verwaltete Richtlinien für Jobfunktionen.

AWS verwaltete Richtlinie: AmazonTimestreamInflux DBServiceRolePolicy

Sie können die AmazonTimestreamInflux DBServiceRolePolicy AWS verwaltete Richtlinie nicht mit Identitäten in Ihrem Konto verknüpfen. Diese Richtlinie ist Teil der Rolle, die mit dem AWS TimestreamforInflux DB-Dienst verknüpft ist. Diese Rolle ermöglicht es dem Dienst, Netzwerkschnittstellen und Sicherheitsgruppen in Ihrem Konto zu verwalten.

Timestream for InfluxDB verwendet die Berechtigungen in dieser Richtlinie, um EC2 Sicherheitsgruppen und Netzwerkschnittstellen zu verwalten. Dies ist erforderlich, um Timestream für InfluxDB-DB-Instances zu verwalten.

Informationen zu dieser Richtlinie im JSON Format finden Sie unter. AmazonTimestreamInfluxDBServiceRolePolicy

AWS-verwaltete Richtlinien für Amazon Timestream for InfluxDB

AWS adressiert viele gängige Anwendungsfälle durch die Bereitstellung eigenständiger IAM Richtlinien, die von erstellt und verwaltet werden. AWS Die verwalteten Richtlinien erteilen die erforderlichen Berechtigungen für viele häufige Anwendungsfälle, sodass Sie nicht mühsam ermitteln müssen, welche Berechtigungen erforderlich sind. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter AWS Verwaltete Richtlinien.

Die folgenden AWS verwalteten Richtlinien, die Sie Benutzern in Ihrem Konto zuordnen können, sind spezifisch für Timestream for InfluxDB:

AmazonTimestreamInfluxDBFullAccess

Sie können die `AmazonTimestreamInfluxDBFullAccess` Richtlinie an Ihre Identitäten anhängen. IAM Diese Richtlinie gewährt Administratorberechtigungen, die vollen Zugriff auf alle Timestream for InfluxDB-Ressourcen ermöglichen.

Sie können auch Ihre eigenen benutzerdefinierten IAM Richtlinien erstellen, um Berechtigungen für Amazon Timestream für API InfluxDB-Aktionen zuzulassen. Sie können diese benutzerdefinierten Richtlinien an die IAM Benutzer oder Gruppen anhängen, die diese Berechtigungen benötigen.

Informationen zur JSON Formatierung dieser Richtlinie finden Sie unter AmazonTimestreamInfluxDBFullAccess.

Timestream für InfluxDB-Aktualisierungen verwalteter Richtlinien AWS

Sehen Sie sich Details zu Aktualisierungen der AWS verwalteten Richtlinien für Timestream for InfluxDB an, seit dieser Dienst begonnen hat, diese Änderungen zu verfolgen. Abonnieren Sie den RSS Feed auf der Seite Timestream for InfluxDB-Dokumentenverlauf, um automatische Benachrichtigungen über Änderungen an dieser Seite zu erhalten.

Änderung	Beschreibung	Datum
AmazonTimestreamInfluxDBFullAccess – Aktualisierung auf eine bestehende Richtlinie	Die `ec2:DescribeRouteTables` Aktion wurde der vorhandenen `AmazonTimestreamInfluxDBFullAccess` verwalteten Richtlinie hinzugefügt. Diese Aktion wird zur Beschreibung Ihrer Routing-Tabellen verwendet	08.10.2024
AWS verwaltete Richtlinie: AmazonTimestreamInfluxDBServiceRolePolicy – Neue Richtlinie.	Amazon Timestream for InfluxDB hat eine neue Richtlinie hinzugefügt, die es dem Service ermöglicht, Netzwerkschnittstellen und Sicherheitsgruppen in Ihrem Konto zu verwalten.	14.03.2024
AmazonTimestreamInfluxDBFullAccess – Neue Richtlinie.	Amazon Timestream for InfluxDB hat eine neue Richtlinie hinzugefügt, die vollen Administratorzugriff zum Erstellen, Aktualisieren, Löschen und Auflisten von Amazon Timestream InfluxDB-Instances sowie zum Erstellen und Auflisten von Parametergruppen bietet.	14.03.2024

Über einen Endpunkt eine Verbindung zu Timestream for InfluxDB herstellen VPC

Sie können über einen privaten Schnittstellenendpunkt in Ihrer virtuellen privaten Cloud () eine direkte Verbindung zu Timestream for InfluxDB herstellen. VPC Wenn Sie einen VPC Schnittstellenendpunkt

verwenden, erfolgt die Kommunikation zwischen Ihnen VPC und Timestream for InfluxDB ausschließlich innerhalb des Netzwerks. AWS

Timestream for InfluxDB unterstützt Amazon Virtual Private Cloud (AmazonVPC) -Endpunkte, die von bereitgestellt werden. AWS PrivateLink Jeder VPC Endpunkt wird durch eine oder mehrere Elastic Network Interfaces (ENIs) mit privaten IP-Adressen in Ihren Subnetzen repräsentiert. VPC

Der VPC Schnittstellen-Endpunkt verbindet Sie VPC direkt mit Timestream for InfluxDB ohne Internet-Gateway, NAT Gerät, VPN Verbindung oder Verbindung. AWS Direct Connect Die Instanzen in Ihrem System benötigen VPC keine öffentlichen IP-Adressen, um mit Timestream for InfluxDB zu kommunizieren.

Regionen

Timestream for InfluxDB unterstützt VPC Endpunkte und VPC Endpunktrichtlinien, in denen Timestream for InfluxDB unterstützt AWS-Regionen wird.

Themen

- Überlegungen zu Timestream für InfluxDB-Endpunkte VPC
- Einen VPC Endpunkt für Timestream for InfluxDB erstellen
- Verbindung zu einem Timestream for InfluxDB-Endpunkt herstellen VPC
- Den Zugriff auf einen VPC Endpunkt kontrollieren
- Verwenden eines VPC Endpunkts in einer Richtlinienerklärung
- Protokollieren Sie Ihren VPC Endpunkt

Überlegungen zu Timestream für InfluxDB-Endpunkte VPC

Bevor Sie einen VPC Schnittstellenendpunkt für Timestream for InfluxDB einrichten, lesen Sie das Thema Eigenschaften und Einschränkungen der Schnittstellenendpunkte im Handbuch.AWS PrivateLink

Die Unterstützung von Timestream for InfluxDB für einen Endpunkt umfasst Folgendes. VPC

- Sie können Ihren VPC Endpunkt verwenden, um alle Timestream for API InfluxDB-Operationen von Ihrem aus aufzurufen. VPC
- Sie können AWS CloudTrail Protokolle verwenden, um Ihre Nutzung von Timestream für InfluxDB-Ressourcen über den Endpunkt zu überprüfen. VPC Details hierzu finden Sie unter Protokollieren Sie Ihren VPC Endpunkt.

Einen VPC Endpunkt für Timestream for InfluxDB erstellen

Sie können einen VPC Endpunkt für Timestream for InfluxDB erstellen, indem Sie die VPC Amazon-Konsole oder Amazon verwenden. VPC API Weitere Informationen finden Sie unter Erstellen eines Schnittstellenendpunkts im AWS PrivateLink -Leitfaden.

- Verwenden Sie den folgenden Dienstnamen, um einen VPC Endpunkt für Timestream for InfluxDB zu erstellen:

  ```
  com.amazonaws.region.timestream-influxdb
  ```

  In der Region USA West (Oregon) (`us-west-2`) würde der Servicename wie folgt lauten:

  ```
  com.amazonaws.us-west-2.timestream-influxdb
  ```

Um die Verwendung des VPC Endpunkts zu vereinfachen, können Sie einen privaten DNS Namen für Ihren Endpunkt aktivieren. VPC Wenn Sie die Option Enable DNS Name auswählen, wird der Standard-Timestream für DNS InfluxDB-Hostname zu Ihrem Endpunkt aufgelöst. VPC `https://timestream-influxdb.us-west-2.amazonaws.com`Würde beispielsweise zu einem VPC Endpunkt aufgelöst, der mit dem Dienstnamen verbunden ist. `com.amazonaws.us-west-2.timestream-influxdb`

Diese Option erleichtert die Verwendung des VPC Endpunkts. Der AWS SDKs und AWS CLI verwendet standardmäßig den Standard-Timestream for DNS InfluxDB-Hostnamen, sodass Sie den VPC Endpunkt nicht URL in Anwendungen und Befehlen angeben müssen.

Weitere Informationen finden Sie unter Zugriff auf einen Service über einen Schnittstellenendpunkt im AWS PrivateLink -Leitfaden.

Verbindung zu einem Timestream for InfluxDB-Endpunkt herstellen VPC

Sie können über den VPC Endpunkt eine Verbindung zu Timestream for InfluxDB herstellen, indem Sie ein AWS SDK, das oder verwenden. AWS CLI AWS Tools for PowerShell Um den VPC Endpunkt anzugeben, verwenden Sie seinen Namen. DNS

Wenn Sie bei der Erstellung Ihres VPC Endpunkts private Hostnamen aktiviert haben, müssen Sie den VPC Endpunkt nicht URL in Ihren CLI Befehlen oder der Anwendungskonfiguration angeben. Der Standard-Timestream für DNS InfluxDB-Hostname wird zu Ihrem Endpunkt aufgelöst. VPC

Der AWS CLI und SDKs verwendet standardmäßig diesen Hostnamen, sodass Sie damit beginnen können, den Endpunkt zu verwenden, um eine Verbindung zu einem regionalen Timestream for VPC InfluxDB-Endpunkt herzustellen, ohne etwas an Ihren Skripten und Anwendungen zu ändern.

Um private Hostnamen zu verwenden, müssen die `enableDnsSupport` Attribute `enableDnsHostnames` und von Ihnen VPC auf gesetzt sein. `true` Verwenden Sie die ModifyVpcAttributeOperation, um diese Attribute festzulegen. Einzelheiten finden Sie unter DNSAttribute für Sie anzeigen und aktualisieren VPC im VPCAmazon-Benutzerhandbuch.

Den Zugriff auf einen VPC Endpunkt kontrollieren

Um den Zugriff auf Ihren VPC Endpunkt für Timestream for InfluxDB zu kontrollieren, fügen Sie Ihrem VPCEndpunkt eine Endpunktrichtlinie hinzu. VPC Die Endpunktrichtlinie bestimmt, ob Prinzipale den VPC Endpunkt verwenden können, um Timestream für InfluxDB-Operationen auf Timestream for InfluxDB-Ressourcen aufzurufen.

Sie können eine VPC Endpunktrichtlinie erstellen, wenn Sie Ihren Endpunkt erstellen, und Sie können die Endpunktrichtlinie jederzeit ändern. VPC Verwenden Sie die VPC Verwaltungskonsole oder die ModifyVpcEndpointOperationen CreateVpcEndpointoder. Sie können eine VPC Endpunktrichtlinie auch mithilfe einer AWS CloudFormation Vorlage erstellen und ändern. Hilfe zur Verwendung der VPC Managementkonsole finden Sie unter Erstellen eines Schnittstellenendpunkts und Ändern eines Schnittstellenendpunkts im AWS PrivateLink Handbuch.

> **Note**
>
> Timestream for InfluxDB unterstützt VPC Endpunktrichtlinien ab Juli 2020. VPCEndpunkte für Timestream for InfluxDB, die vor diesem Datum erstellt wurden, haben die VPCStandard-Endpunktrichtlinie, die Sie jedoch jederzeit ändern können.

Themen

- Über Endpunktrichtlinien VPC
- VPCStandard-Endpunktrichtlinie
- Eine Endpunktrichtlinie VPC erstellen
- Eine VPC Endpunktrichtlinie anzeigen

Über Endpunktrichtlinien VPC

Damit eine Timestream for InfluxDB-Anfrage, die einen VPC Endpunkt verwendet, erfolgreich ist, benötigt der Principal Berechtigungen aus zwei Quellen:

- Eine IAMRichtlinie muss dem Principal die Erlaubnis erteilen, den Vorgang auf der Ressource aufzurufen.
- Eine VPC Endpunktrichtlinie muss dem Prinzipal die Erlaubnis erteilen, den Endpunkt für die Anforderung zu verwenden.

VPCStandard-Endpunktrichtlinie

Jeder VPC Endpunkt hat eine VPC Endpunktrichtlinie, aber Sie müssen die Richtlinie nicht angeben. Wenn Sie keine Richtlinie angeben, erlaubt die standardmäßige Endpunktrichtlinie alle Operationen aller Prinzipale auf allen Ressourcen über den Endpunkt.

Für Timestream for InfluxDB-Ressourcen muss der Principal jedoch auch die Berechtigung haben, den Vorgang über eine IAMRichtlinie aufzurufen. In der Praxis besagt die Standardrichtlinie daher, dass ein Principal, wenn er berechtigt ist, einen Vorgang für eine Ressource aufzurufen, ihn auch mithilfe des Endpunkts aufrufen kann.

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Principal": "*",
      "Resource": "*"
    }
  ]
}
```

Um es Prinzipalen zu ermöglichen, den VPC Endpunkt nur für eine Teilmenge ihrer erlaubten Operationen zu verwenden, erstellen oder aktualisieren Sie die Endpunktrichtlinie. VPC

Eine Endpunktrichtlinie VPC erstellen

Eine VPC Endpunktrichtlinie bestimmt, ob ein Principal berechtigt ist, den VPC Endpunkt zur Ausführung von Vorgängen auf einer Ressource zu verwenden. Für Timestream for InfluxDB-

Ressourcen muss der Principal auch die Erlaubnis haben, die Operationen anhand einer Richtlinie auszuführen. IAM

Jede VPC Endpunkt-Richtlinienerklärung erfordert die folgenden Elemente:

- Der Prinzipal, der die Aktionen ausführen kann
- Aktionen, die ausgeführt werden können
- Ressourcen, für die Aktionen ausgeführt werden können

In der Richtlinienerklärung wird der VPC Endpunkt nicht angegeben. Stattdessen gilt sie für jeden VPC Endpunkt, an den die Richtlinie angehängt ist. Weitere Informationen finden Sie unter Steuern des Zugriffs auf Dienste mit VPC Endpunkten im VPCAmazon-Benutzerhandbuch.

AWS CloudTrail protokolliert alle Operationen, die den VPC Endpunkt verwenden.

Eine VPC Endpunktrichtlinie anzeigen

Um die VPC Endpunktrichtlinie für einen Endpunkt anzuzeigen, verwenden Sie die VPCVerwaltungskonsole oder den DescribeVpcEndpointsVorgang.

Mit dem folgenden AWS CLI Befehl wird die Richtlinie für den Endpunkt mit der angegebenen VPC Endpunkt-ID abgerufen.

Bevor Sie diesen Befehl ausführen, ersetzen Sie die Beispiel-Endpunkt-ID durch eine gültige aus Ihrem Konto.

```
$ aws ec2 describe-vpc-endpoints \
--query 'VpcEndpoints[?VpcEndpointId==`vpc-endpoint-id`].[PolicyDocument]'
--output text
```

Verwenden eines VPC Endpunkts in einer Richtlinienerklärung

Sie können den Zugriff auf Timestream für InfluxDB-Ressourcen und -Operationen steuern, wenn die Anfrage von einem Endpunkt kommt VPC oder diesen verwendet. VPC Verwenden Sie dazu einen der folgenden globalen Bedingungsschlüssel in einer Richtlinie. IAM

- Verwenden Sie den `aws:sourceVpce` Bedingungsschlüssel, um den Zugriff je nach VPC Endpunkt zu gewähren oder einzuschränken.
- Verwenden Sie den `aws:sourceVpc` Bedingungsschlüssel, um den Zugriff auf der Grundlage des Hostings des VPC privaten Endpunkts zu gewähren oder einzuschränken.

> **Note**
>
> Seien Sie vorsichtig, wenn Sie wichtige Richtlinien und IAM Richtlinien auf der Grundlage Ihres VPC Endpunkts erstellen. Wenn eine Richtlinienerklärung vorschreibt, dass Anfragen von einem bestimmten VPC Endpunkt VPC oder einem bestimmten Endpunkt kommen, schlagen Anfragen von integrierten AWS Diensten, die in Ihrem Namen eine Timestream for InfluxDB-Ressource verwenden, möglicherweise fehl.
> Außerdem ist der `aws:sourceIP` Bedingungsschlüssel nicht wirksam, wenn die Anfrage von einem VPCAmazon-Endpunkt kommt. Um Anfragen auf einen VPC Endpunkt zu beschränken, verwenden Sie die `aws:sourceVpc` Bedingungsschlüssel `aws:sourceVpce` oder. Weitere Informationen finden Sie im AWS PrivateLink Handbuch unter Identitäts- und Zugriffsmanagement für VPC VPC Endgeräte und Endpunktdienste.

Sie können diese globalen Bedingungsschlüssel verwenden, um den Zugriff auf solche Operationen zu kontrollieren CreateDbInstance, die nicht von einer bestimmten Ressource abhängen.

Protokollieren Sie Ihren VPC Endpunkt

AWS CloudTrail protokolliert alle Operationen, die den VPC Endpunkt verwenden. Wenn eine Anfrage an Timestream for InfluxDB einen VPC Endpunkt verwendet, erscheint die VPC Endpunkt-ID im AWS CloudTrail Protokolleintrag, der die Anfrage aufzeichnet. Sie können die Endpunkt-ID verwenden, um die Verwendung Ihres Timestream for InfluxDB-Endpunkts zu überprüfen. VPC

Ihre CloudTrail Protokolle enthalten jedoch keine Operationen, die von Principals in anderen Konten angefordert wurden, oder Anfragen nach Timestream für InfluxDB-Operationen auf Timestream für InfluxDB-Ressourcen und Aliase in anderen Konten. Zu Ihrem Schutz werden AnfragenVPC, die durch eine VPCEndpunktrichtlinie abgelehnt wurden, aber andernfalls zugelassen worden wären, nicht aufgezeichnet. AWS CloudTrail

Protokollierung und Überwachung in Timestream für InfluxDB

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit und Leistung von Timestream for InfluxDB und Ihren Lösungen. AWS Sie sollten Überwachungsdaten aus allen Teilen Ihrer AWS Lösung sammeln, damit Sie einen Fehler an mehreren Punkten leichter debuggen können, falls einer auftritt. Bevor Sie jedoch mit der Überwachung von Timestream for InfluxDB beginnen, sollten Sie einen Überwachungsplan erstellen, der Antworten auf die folgenden Fragen enthält:

- Was sind Ihre Ziele bei der Überwachung?
- Welche Ressourcen werden überwacht?
- Wie oft werden diese Ressourcen überwacht?
- Welche Überwachungstools werden verwendet?
- Wer soll die Überwachungsaufgaben ausführen?
- Wer soll benachrichtigt werden, wenn Fehler auftreten?

Der nächste Schritt besteht darin, eine Ausgangsbasis für die normale Leistung von Timestream für InfluxDB in Ihrer Umgebung festzulegen, indem Sie die Leistung zu verschiedenen Zeiten und unter verschiedenen Lastbedingungen messen. Speichern Sie bei der Überwachung von Timestream für InfluxDB historische Überwachungsdaten, damit Sie sie mit aktuellen Leistungsdaten vergleichen, normale Leistungsmuster und Leistungsanomalien identifizieren und Methoden zur Behebung von Problemen entwickeln können.

Zur Festlegung eines Grundwertes sollten Sie mindestens die folgenden Elemente überwachen:

- Systemfehler, sodass Sie feststellen können, ob Anfragen zu einem Fehler geführt haben.

Themen

Überwachungstools

AWS bietet verschiedene Tools, mit denen Sie Timestream für InfluxDB überwachen können. Sie können einige dieser Tools so konfigurieren, dass diese die Überwachung für Sie übernehmen, während bei anderen Tools ein manuelles Eingreifen nötig ist. Wir empfehlen, dass Sie die Überwachungsaufgaben möglichst automatisieren.

Themen

Automatisierte Überwachungstools

Sie können die folgenden automatisierten Überwachungstools verwenden, um Timestream for InfluxDB zu beobachten und zu melden, wenn etwas nicht stimmt:

- Amazon CloudWatch Alarms — Überwachen Sie eine einzelne Metrik über einen von Ihnen angegebenen Zeitraum und führen Sie eine oder mehrere Aktionen aus, die auf dem Wert der Metrik im Verhältnis zu einem bestimmten Schwellenwert über mehrere Zeiträume basieren. Die Aktion ist eine Benachrichtigung, die an ein Amazon Simple Notification Service (AmazonSNS) -Thema oder eine Amazon EC2 Auto Scaling Scaling-Richtlinie gesendet wird. CloudWatch Alarme lösen keine Aktionen aus, nur weil sie sich in einem bestimmten Zustand befinden. Der Status muss sich geändert haben und für eine bestimmte Anzahl von Zeiträumen beibehalten worden sein. Weitere Informationen finden Sie unter Überwachung mit Amazon CloudWatch.

Manuelle Überwachungstools

Ein weiterer wichtiger Teil der Überwachung von Timestream for InfluxDB besteht darin, die Elemente manuell zu überwachen, die von den CloudWatch Alarmen nicht abgedeckt werden. Der Timestream für InfluxDB, CloudWatch Trusted Advisor, und andere AWS Management Console Dashboards bieten einen at-a-glance Überblick über den Zustand Ihrer Umgebung. AWS

- Die CloudWatch Startseite zeigt Folgendes:
  - Aktuelle Alarme und Status
  - Diagramme mit Alarmen und Ressourcen
  - Servicestatus

  Darüber hinaus können CloudWatch Sie Folgendes verwenden:
  - Erstellen angepasster Dashboards zur Überwachung der gewünschten Services.
  - Aufzeichnen von Metrikdaten, um Probleme zu beheben und Trends zu erkennen
  - Suchen und durchsuchen Sie alle Ihre AWS Ressourcenmetriken
  - Erstellen und Bearbeiten von Alarmen, um über Probleme benachrichtigt zu werden

Timestream für API InfluxDB-Aufrufe protokollieren mit AWS CloudTrail

Timestream for InfluxDB ist in einen Dienst integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen eines Benutzers, einer Rolle oder eines AWS Dienstes in Timestream for InfluxDB

bereitstellt. CloudTrail erfasst API Aufrufe der Data Definition Language (DDL) für Timestream for InfluxDB als Ereignisse. Zu den aufgenommenen Aufrufen gehören Aufrufe von der Timestream for InfluxDB-Konsole und Code-Aufrufe an den Timestream für InfluxDB-Operationen. API Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Bereitstellung von CloudTrail Ereignissen an einen Amazon Simple Storage Service (Amazon S3) -Bucket aktivieren, einschließlich Ereignissen für Timestream for InfluxDB. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse trotzdem auf der CloudTrail Konsole im Ereignisverlauf anzeigen. Anhand der von gesammelten Informationen können Sie die Anfrage CloudTrail, die an Timestream for InfluxDB gestellt wurde, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Details ermitteln.

Weitere Informationen CloudTrail dazu finden Sie im AWS CloudTrail Benutzerhandbuch.

Timestream für InfluxDB-Informationen in CloudTrail

CloudTrail ist in Ihrem AWS Konto aktiviert, wenn Sie das Konto erstellen. Wenn in Timestream for InfluxDB Aktivitäten auftreten, wird diese Aktivität zusammen mit anderen AWS Serviceereignissen in der CloudTrail Ereignishistorie in einem Ereignis aufgezeichnet. Sie können die neusten Ereignisse in Ihr AWS -Konto herunterladen und dort suchen und anzeigen. Weitere Informationen finden Sie unter Ereignisse mit CloudTrail Ereignisverlauf anzeigen.

Für eine fortlaufende Aufzeichnung von Ereignissen in Ihrem AWS Konto, einschließlich Ereignissen für Timestream for InfluxDB, erstellen Sie einen Trail. Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Wenn Sie einen Trail in der Konsole erstellen, gilt der Trail standardmäßig für alle AWS Regionen. Der Trail protokolliert Ereignisse aus allen Regionen der AWS Partition und übermittelt die Protokolldateien an den von Ihnen angegebenen Amazon S3 S3-Bucket. Darüber hinaus können Sie andere AWS Dienste konfigurieren, um die in den CloudTrail Protokollen gesammelten Ereignisdaten weiter zu analysieren und darauf zu reagieren.

Weitere Informationen finden Sie in folgenden Themen im AWS CloudTrail -Benutzerhandbuch:

- Übersicht zum Erstellen eines Trails
- CloudTrail Unterstützte Dienste und Integrationen
- Konfiguration von SNS Amazon-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail Protokolldateien aus mehreren Regionen
- Empfangen von CloudTrail Protokolldateien von mehreren Konten
- Protokollierung von Datenereignissen

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anfrage mit Root- oder AWS Identity and Access Management (IAM) Benutzeranmeldedaten gestellt wurde
- Ob die Anfrage mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen föderierten Benutzer ausgeführt wurde
- Ob die Anfrage von einem anderen AWS Dienst gestellt wurde

Weitere Informationen finden Sie im CloudTrail userIdentityElement.

Compliance-Validierung für Amazon Timestream for InfluxDB

Externe Prüfer bewerten die Sicherheit und Konformität von Amazon Timestream for InfluxDB im Rahmen mehrerer AWS Compliance-Programme. Diese umfassen u. a. folgende:

- GDPR
- HIPAA
- PCI
- SOC

Um zu erfahren, ob AWS-Service ein in den Geltungsbereich bestimmter Compliance-Programme fällt, finden Sie unter AWS-Services Umfang nach Compliance-Programm AWS-Services unter aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter AWS Compliance-Programme AWS .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Berichte herunterladen unter .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. AWS stellt die folgenden Ressourcen zur Verfügung, die Sie bei der Einhaltung der Vorschriften unterstützen:

- Schnellstartanleitungen zu Sicherheit und Compliance — In diesen Bereitstellungsleitfäden werden architektonische Überlegungen erörtert und Schritte für die Bereitstellung von Basisumgebungen beschrieben AWS , bei denen Sicherheit und Compliance im Mittelpunkt stehen.

- Architecting for HIPAA Security and Compliance on Amazon Web Services — In diesem Whitepaper wird beschrieben, wie Unternehmen Anwendungen erstellen HIPAA können, die AWS für sie in Frage kommen.

  > **Note**
  >
  > Nicht alle sind berechtigt AWS-Services . HIPAA Weitere Informationen finden Sie in der Referenz für HIPAA qualifizierte Dienste.

- AWS Ressourcen zur AWS von Vorschriften — Diese Sammlung von Arbeitsmappen und Leitfäden kann auf Ihre Branche und Ihren Standort zutreffen.
- AWS Leitfäden zur Einhaltung von Vorschriften für Kunden — Verstehen Sie das Modell der gemeinsamen Verantwortung aus dem Blickwinkel der Einhaltung von Vorschriften. In den Leitfäden werden die bewährten Verfahren zur Sicherung zusammengefasst AWS-Services und die Leitlinien für Sicherheitskontrollen in verschiedenen Frameworks (einschließlich des National Institute of Standards and Technology (NIST), des Payment Card Industry Security Standards Council (PCI) und der International Organization for Standardization (ISO)) zusammengefasst.
- Evaluierung von Ressourcen anhand von Regeln im AWS Config Entwicklerhandbuch — Der AWS Config Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- AWS Security Hub— Dies AWS-Service bietet einen umfassenden Überblick über Ihren internen Sicherheitsstatus. AWS Security Hub verwendet Sicherheitskontrollen, um Ihre AWS -Ressourcen zu bewerten und Ihre Einhaltung von Sicherheitsstandards und bewährten Methoden zu überprüfen. Eine Liste der unterstützten Services und Kontrollen finden Sie in der Security-Hub-Steuerungsreferenz.
- Amazon GuardDuty — Dies AWS-Service erkennt potenzielle Bedrohungen für Ihre Workloads AWS-Konten, Container und Daten, indem es Ihre Umgebung auf verdächtige und böswillige Aktivitäten überwacht. GuardDuty kann Ihnen helfen, verschiedene Compliance-Anforderungen zu erfüllen PCIDSS, z. B. durch die Erfüllung der Anforderungen zur Erkennung von Eindringlingen, die in bestimmten Compliance-Frameworks vorgeschrieben sind.
- AWS Audit Manager— Auf diese AWS-Service Weise können Sie Ihre AWS Nutzung kontinuierlich überprüfen, um das Risikomanagement und die Einhaltung von Vorschriften und Industriestandards zu vereinfachen.

Resilienz in Amazon Timestream für InfluxDB

Die AWS globale Infrastruktur basiert AWS auf Regionen und Availability Zones. AWS Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Amazon Timestream for InfluxDB erstellt regelmäßig interne Backups und bewahrt sie 24 Stunden lang auf, um Verfügbarkeit und Haltbarkeit zu gewährleisten. Snapshots werden bei Löschungen erstellt und zur Unterstützung von Wiederherstellungen 30 Tage lang aufbewahrt. Um auf diese zuzugreifen oder sie zu verwenden, reichen Sie beim AWS Support ein Ticket ein.

Sie können Ihre Instance mit Multi-AZ-Wiederherstellungsfunktionen erstellen. Weitere Informationen finden Sie unter Multi-AZ-DB-Instance-Bereitstellungen.

Infrastruktursicherheit in Amazon Timestream für InfluxDB

Als verwalteter Service ist Amazon Timestream for InfluxDB durch die AWS globalen Netzwerksicherheitsverfahren geschützt, die im Whitepaper Amazon Web Services: Sicherheitsprozesse im Überblick beschrieben sind.

Sie verwenden AWS veröffentlichte API Aufrufe der Kontrollebene, um über das Netzwerk auf Timestream for InfluxDB zuzugreifen. Weitere Informationen finden Sie unter Kontrollebenen und Datenebenen. Clients müssen Transport Layer Security (TLS) 1.2 oder höher unterstützen. Wir empfehlen TLS 1.2 oder 1.3. Kunden müssen außerdem Cipher Suites mit Perfect Forward Secrecy (PFS) wie Ephemeral Diffie-Hellman () oder Elliptic Curve Ephemeral Diffie-Hellman (DHE) unterstützen. ECDHE Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Darüber hinaus müssen Anfragen mithilfe einer Zugriffsschlüssel-ID und eines geheimen Zugriffsschlüssels, der einem Prinzipal zugeordnet ist, signiert werden. IAM Alternativ können Sie mit AWS Security Token Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Timestream for InfluxDB ist so konzipiert, dass Ihr Datenverkehr auf die spezifische AWS Region beschränkt ist, in der sich Ihre Timestream for InfluxDB-Instance befindet.

Sicherheitsgruppen

Sicherheitsgruppen kontrollieren die Zugriffsaktivitäten von eingehendem und ausgehendem Datenverkehr in einer DB-Instance. Standardmäßig ist der Netzwerkzugriff auf eine DB-Instance deaktiviert. Sie können Regeln in einer Sicherheitsgruppe angeben, die den Zugriff aus einem IP-Adressbereich, über einen Port oder für eine Sicherheitsgruppe zulassen. Nach der Konfiguration von ingress-Regeln gelten diese für alle DB-Instances, die dieser Sicherheitsgruppe zugeordnet sind.

Weitere Informationen finden Sie unter Steuern des Zugriffs auf eine DB-Instance in einem VPC.

Konfiguration und Schwachstellenanalyse in Timestream for InfluxDB

Konfiguration und IT-Kontrollen liegen in der gemeinsamen Verantwortung von Ihnen, AWS unserem Kunden. Weitere Informationen finden Sie im Modell der AWS gemeinsamen Verantwortung. Zusätzlich zum Modell der gemeinsamen Verantwortung sollten Timestream for InfluxDB-Benutzer Folgendes beachten:

- Es obliegt dem Kunden, seine Client-Anwendungen mit den relevanten kundenseitigen Abhängigkeiten zu patchen.
- Kunden sollten gegebenenfalls Penetrationstests in Betracht ziehen (siehe https://aws.amazon.com/security/ Penetrationstesting/.)

Reaktion auf Vorfälle in Timestream für InfluxDB

Servicevorfälle mit Amazon Timestream for InfluxDB werden im Personal Health Dashboard gemeldet. Sie können mehr über das Dashboard und hier erfahren. AWS Health

Timestream for InfluxDB unterstützt die Berichterstattung mithilfe von. AWS CloudTrail Weitere Informationen finden Sie unter Timestream für API InfluxDB-Aufrufe protokollieren mit AWS CloudTrail.

Amazon Timestream für InfluxDB API und VPC Schnittstellenendpunkte ()AWS PrivateLink

Sie können eine private Verbindung zwischen Ihren VPC und den Endpunkten der Amazon Amazon Timestream for InfluxDB-Steuerebene herstellen, indem Sie einen API Schnittstellenendpunkt

erstellen. VPC Schnittstellen-Endpunkte werden betrieben von. AWS PrivateLink AWS PrivateLink ermöglicht Ihnen den privaten Zugriff auf Amazon Timestream für API InfluxDB-Operationen ohne Internet-Gateway, NAT Gerät, VPN Verbindung oder AWS Direct Connect-Verbindung.

Ihre Instances benötigen VPC keine öffentlichen IP-Adressen, um mit Amazon Timestream für API InfluxDB-Endpunkte zu kommunizieren. Ihre Instances benötigen auch keine öffentlichen IP-Adressen, um einen der verfügbaren Timestreams für InfluxDB-Operationen zu verwenden. API Der Datenverkehr zwischen Ihnen VPC und Amazon Timestream for InfluxDB verlässt das Amazon-Netzwerk nicht. Jeder Schnittstellenendpunkt wird durch eine oder mehrere Elastic Network-Schnittstellen in Ihren Subnetzen dargestellt. Weitere Informationen zu Elastic Network Interfaces finden Sie unter Elastic Network Interfaces im EC2Amazon-Benutzerhandbuch.

- Weitere Informationen zu VPC Endpunkten finden Sie unter Interface VPC endpoints (AWS PrivateLink) im VPCAmazon-Benutzerhandbuch.
- Weitere Informationen zu Timestream für InfluxDB-Operationen finden Sie unter Timestream für API InfluxDB-Operationen. API

Nachdem Sie einen VPC Schnittstellenendpunkt erstellt haben und private DNS Hostnamen für den Endpunkt aktivieren, der Standard-Timestream for InfluxDB-Endpunkt (https://timestream-influxb).*Region*.amazonaws.com) wird zu Ihrem Endpunkt aufgelöst. VPC Wenn Sie private DNS Hostnamen nicht aktivieren, VPC stellt Amazon einen DNS Endpunktnamen bereit, den Sie im folgenden Format verwenden können:

```
VPC_Endpoint_ID.timestream-influxb.Region.vpce.amazonaws.com
```

Weitere Informationen finden Sie unter Interface VPC Endpoints (AWS PrivateLink) im VPCAmazon-Benutzerhandbuch. Timestream for InfluxDB unterstützt Aufrufe aller seiner API Aktionen in Ihrem. VPC

> **Note**
>
> Private DNS Hostnamen können nur für einen VPC Endpunkt in der aktiviert werden. VPC Wenn Sie einen zusätzlichen VPC Endpunkt erstellen möchten, sollte der private DNS Hostname dafür deaktiviert werden.

Überlegungen zu Endpunkten VPC

Bevor Sie einen VPC Schnittstellenendpunkt für Amazon Timestream für API InfluxDB-Endpunkte einrichten, stellen Sie sicher, dass Sie die Eigenschaften und Einschränkungen der Schnittstellenendpunkte im Amazon-Benutzerhandbuch lesen. VPC Alle Timestream for API InfluxDB-Operationen, die für die Verwaltung von Amazon Timestream for InfluxDB-Ressourcen relevant sind, stehen Ihnen zur Verfügung. VPC AWS PrivateLink VPCEndpunktrichtlinien werden für Timestream für InfluxDB-Endpunkte unterstützt. API Standardmäßig ist der volle Zugriff auf Timestream für API InfluxDB-Operationen über den Endpunkt erlaubt. Weitere Informationen finden Sie unter Steuern des Zugriffs auf Dienste mit VPC Endpunkten im VPCAmazon-Benutzerhandbuch.

Einen VPC Schnittstellenendpunkt für den Timestream für InfluxDB erstellen API

Sie können einen VPC Endpunkt für Amazon Timestream for InfluxDB entweder API mit der VPC Amazon-Konsole oder der erstellen. AWS CLI Weitere Informationen finden Sie unter Erstellen eines Schnittstellenendpunkts im VPCAmazon-Benutzerhandbuch.

Nachdem Sie einen VPC Schnittstellenendpunkt erstellt haben, können Sie private DNS Hostnamen für den Endpunkt aktivieren. Wenn Sie dies tun, der standardmäßige Amazon Timestream for InfluxDB-Endpunkt (https://timestream-influxb.*Region*.amazonaws.com) wird zu Ihrem Endpunkt aufgelöst. VPC Weitere Informationen finden Sie unter Zugreifen auf einen Service über einen Schnittstellenendpunkt im VPCAmazon-Benutzerhandbuch.

Erstellen einer VPC Endpunktrichtlinie für Amazon Timestream for InfluxDB API

Sie können Ihrem VPC Endpunkt eine Endpunktrichtlinie hinzufügen, die den Zugriff auf den Timestream für InfluxDB steuert. API Die Richtlinie legt Folgendes fest:

- Prinzipal, der die Aktionen ausführen kann.
- Aktionen, die ausgeführt werden können
- Die Ressourcen, für die Aktionen ausgeführt werden können.

Weitere Informationen finden Sie unter Steuern des Zugriffs auf Dienste mit VPC Endpunkten im VPCAmazon-Benutzerhandbuch.

Example VPCEndpunktrichtlinie für Timestream für InfluxDB-Aktionen API

Das Folgende ist ein Beispiel für eine Endpunktrichtlinie für den Timestream für InfluxDB. API Wenn diese Richtlinie an einen Endpunkt angehängt ist, gewährt sie allen Prinzipalen auf allen Ressourcen Zugriff auf die aufgelisteten Timestream for API InfluxDB-Aktionen.

```
{
 "Statement": [{
  "Principal": "*",
  "Effect": "Allow",
  "Action": [
   "timestream-influxb:CreateDbInstance",
   "timestream-influxb:UpdateDbInstance"
  ],
  "Resource": "*"
 }]
}
```

Example VPCEndpunktrichtlinie, die jeglichen Zugriff von einem bestimmten Konto aus verweigert AWS

Die folgende VPC Endpunktrichtlinie verweigert AWS das Konto *123456789012* der gesamte Zugriff auf Ressourcen über den Endpunkt. Die Richtlinie erlaubt alle Aktionen von anderen Konten.

```
{
 "Statement": [{
   "Action": "*",
   "Effect": "Allow",
   "Resource": "*",
   "Principal": "*"
  },
  {
   "Action": "*",
   "Effect": "Deny",
   "Resource": "*",
   "Principal": {
    "AWS": [
     "123456789012"
    ]
   }
  }
 ]
}
```

Bewährte Sicherheitsmethoden für Timestream for InfluxDB

Amazon Timestream for InfluxDB bietet eine Reihe von Sicherheitsfunktionen, die Sie bei der Entwicklung und Implementierung Ihrer eigenen Sicherheitsrichtlinien berücksichtigen sollten. Die

folgenden bewährten Methoden sind allgemeine Richtlinien und keine vollständige Sicherheitslösung. Da diese bewährten Methoden für Ihre Umgebung möglicherweise nicht angemessen oder ausreichend sind, sollten Sie sie als hilfreiche Überlegungen und nicht als bindend ansehen.

Implementieren des Zugriffs mit geringsten Berechtigungen

Bei der Erteilung von Berechtigungen entscheiden Sie, wer welche Berechtigungen für welche Timestream-Ressourcen für InfluxDB-Ressourcen erhält. Sie aktivieren die spezifischen Aktionen, die daraufhin für die betreffenden Ressourcen erlaubt sein sollen. Aus diesem Grund sollten Sie nur Berechtigungen gewähren, die zum Ausführen einer Aufgabe erforderlich sind. Die Implementierung der geringstmöglichen Zugriffsrechte ist eine grundlegende Voraussetzung zum Reduzieren des Sicherheitsrisikos und der Auswirkungen, die aufgrund von Fehlern oder böswilligen Absichten entstehen könnten.

Verwenden Sie Rollen IAM

Produzenten- und Client-Anwendungen müssen über gültige Anmeldeinformationen verfügen, um auf Timestream für InfluxDB-DB-Instances zugreifen zu können. Sie sollten AWS Anmeldeinformationen nicht direkt in einer Client-Anwendung oder in einem Amazon S3 S3-Bucket speichern. Dabei handelt es sich um langfristige Anmeldeinformationen, die nicht automatisch rotiert werden und bedeutende geschäftliche Auswirkungen haben könnten, wenn sie kompromittiert werden.

Stattdessen sollten Sie eine IAM Rolle verwenden, um temporäre Anmeldeinformationen für Ihre Produzenten- und Client-Anwendungen für den Zugriff auf Timestream für InfluxDB-DB-Instances zu verwalten. Wenn Sie eine Rolle verwenden, müssen Sie keine langfristigen Anmeldeinformationen (z. B. Benutzername und Passwort oder Zugriffsschlüssel) für den Zugriff auf andere Ressourcen verwenden.

Weitere Informationen finden Sie in den folgenden Themen im IAM Benutzerhandbuch:

- IAMRollen
- Gängige Szenarien für Rollen: Benutzer, Anwendungen und Services

Verwenden Sie AWS Identity and Access Management (IAM) -Konten, um den Zugriff auf Amazon Timestream für API InfluxDB-Operationen zu steuern, insbesondere Operationen, die Amazon Timestream for InfluxDB-Ressourcen erstellen, ändern oder löschen. Zu diesen Ressourcen gehören DB-Instances, Sicherheitsgruppen und Parametergruppen.

- Erstellen Sie einen individuellen Benutzer für jede Person, die Amazon Timestream for InfluxDB-Ressourcen verwaltet, einschließlich Ihnen selbst. Verwenden Sie keine AWS Root-Anmeldeinformationen, um Amazon Timestream for InfluxDB-Ressourcen zu verwalten.
- Gewähren Sie jedem Benutzer nur den Mindestsatz an Berechtigungen, die für die Ausführung seiner Aufgaben erforderlich sind.
- Verwenden Sie IAM Gruppen, um Berechtigungen für mehrere Benutzer effektiv zu verwalten.
- Wechseln Sie regelmäßig die IAM-Anmeldeinformationen.
- Konfigurieren Sie AWS Secrets Manager so, dass die Secrets für Amazon Timestream for InfluxDB automatisch rotiert werden. Weitere Informationen finden Sie unter Rotation Ihrer AWS Secrets Manager Manager-Geheimnisse im AWS Secrets Manager Manager-Benutzerhandbuch. Sie können die Anmeldeinformationen auch programmgesteuert von AWS Secrets Manager abrufen. Weitere Informationen finden Sie unter Abrufen des geheimen Werts im AWS Secrets Manager Manager-Benutzerhandbuch.
- Sichern Sie Ihren Timestream für API InfluxDB-Influx-Token, indem Sie die verwenden. APITokens

Implementieren einer serverseitigen Verschlüsselung in abhängigen Ressourcen

Daten im Ruhezustand und Daten während der Übertragung können in Timestream for InfluxDB verschlüsselt werden. Weitere Informationen finden Sie unter Verschlüsselung während der Übertragung.

Wird zur Überwachung von Anrufen verwendet CloudTrail API

Timestream for InfluxDB ist in einen Dienst integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen eines Benutzers, einer Rolle oder eines AWS Dienstes in Timestream for InfluxDB bereitstellt.

Anhand der von gesammelten Informationen können Sie die Anfrage CloudTrail, die an Timestream for InfluxDB gestellt wurde, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Details ermitteln.

Weitere Informationen finden Sie unter the section called “Protokollierung des Timestreams für LiveAnalytics API Anrufe mit AWS CloudTrail”.

Amazon Timestream for InfluxDB unterstützt CloudTrail Ereignisse auf der Kontrollebene, aber keine Datenebene. Weitere Informationen finden Sie unter Kontrollebenen und Datenebenen.

Öffentliche Zugänglichkeit

Wenn Sie eine DB-Instance in einer Virtual Private Cloud (VPC) starten, die auf dem VPC Amazon-Service basiert, können Sie den öffentlichen Zugriff für diese DB-Instance ein- oder ausschalten. Um festzulegen, ob die von Ihnen erstellte DB-Instance einen DNS Namen hat, der in eine öffentliche IP-Adresse aufgelöst wird, verwenden Sie den Parameter Public Accessibility. Mithilfe dieses Parameters können Sie festlegen, ob ein öffentlicher Zugriff auf die DB-Instance besteht

Wenn sich Ihre DB-Instance in einer befindet, VPC aber nicht öffentlich zugänglich ist, können Sie auch eine AWS Site-to-Site VPN Verbindung oder eine AWS Direct Connect-Verbindung verwenden, um von einem privaten Netzwerk aus darauf zuzugreifen.

Wenn Ihre DB-Instance öffentlich zugänglich ist, sollten Sie unbedingt Maßnahmen ergreifen, um Denial-of-Service-Bedrohungen zu verhindern oder zu mindern. Weitere Informationen finden Sie unter Einführung in Denial-of-Service-Angriffe und Schutz von Netzwerken.

APIReferenz

Eine vollständige Liste und Einzelheiten zu Amazon Timestream for InfluxDB finden Sie unter Amazon Timestream for InfluxDBAPIs. APIs

Fehlercodes, die allen AWS Diensten gemeinsam sind, finden Sie im Abschnitt AWS Support.

Dokumentverlauf

Änderung	Beschreibung	Datum
Update nur für die Dokumentation	Das Thema Kontingente wurde aktualisiert, um die Standardkontingente und Systemlimits voneinander zu trennen.	22. Oktober 2024
Amazon Timestream unterstützt jetzt Query-Insights	Timestream bietet jetzt Unterstützung für die Query Insights-Funktion, mit der Sie Ihre Abfragen optimieren,	22. Oktober 2024

	deren Leistung verbessern und Kosten senken können.	
Amazon Timestream for InfluxDB aktualisiert eine bestehende Richtlinie.	Amazon Timestream for InfluxDB hat die `ec2:DescribeRouteTables` Aktion zur bestehenden `AmazonTimestreamInfluxDBFullAccess` verwalteten Richtlinie zur Beschreibung Ihrer Routing-Tabellen hinzugefügt	8. Oktober 2024
`AmazonTimestreamReadOnlyAccess` — Aktualisierung einer bestehenden Richtlinie	Timestream for LiveAnalytics hat der `AmazonTimestreamReadOnlyAccess` verwalteten Richtlinie die `DescribeAccountSettings` Berechtigung zur Beschreibung von AWS-Konto Einstellungen hinzugefügt.	3. Juni 2024
Amazon Timestream unterstützt LiveAnalytics derzeit Timestream Compute Units () TCUs	Amazon Timestream bietet LiveAnalytics derzeit Unterstützung für Timestream Compute Units (TCUs) zur Messung der Rechenkapazität, die für Ihre Abfrageanforderungen zugewiesen wurde.	29. April 2024

Neue Richtlinien wurden hinzugefügt	Amazon Timestream for InfluxDB hat zwei neue Richtlinien hinzugefügt: Eine, die es dem Service ermöglicht, Netzwerkschnittstellen und Sicherheitsgruppen in Ihrem Konto zu verwalten. Weitere Informationen finden Sie unter. AmazonTimestreamInfluxDBServiceRolePolicy Eine weitere, die vollen Administratorzugriff zum Erstellen, Aktualisieren, Löschen und Auflisten von Amazon Timestream InfluxDB-Instances sowie zum Erstellen und Auflisten von Parametergruppen bietet. Weitere Informationen finden Sie unter. AmazonTimestreamInfluxDBFullAccess	14. März 2024
Amazon Timestream für InfluxDB ist jetzt allgemein verfügbar.	Diese Dokumentation behandelt die erste Version von Amazon Timestream für InfluxDB.	14. März 2024

Amazon Timestream for LiveAnalytics Query-Ereignisse sind verfügbar in AWS CloudTrail	Amazon Timestream veröffentlicht LiveAnalytics vorerst API Abfragedatenereignisse für. AWS CloudTrail Kunden können alle API Query-Anfragen, die in ihren AWS Konten gestellt wurden, prüfen und sich Informationen darüber anzeigen lassen, welcher IAM Benutzer/welche Rolle die Anfrage gestellt hat, wann die Anfrage gestellt wurde, welche Datenbanken und Tabellen abgefragt wurden und welche Abfrage-ID der Anfrage lautet.	12. September 2023
Amazon Timestream für LiveAnalytics UNLOAD	Amazon Timestream unterstützt LiveAnalytics derzeit UNLOAD den Export von Abfrageergebnissen nach S3.	12. Mai 2023
Amazon Timestream für die LiveAnalytics Aktualisierung einer bestehenden Richtlinie.	Batchladeberechtigungen wurden zu einer verwalteten Richtlinie hinzugefügt.	24. Februar 2023
Amazon Timestream für LiveAnalytics Batch-Load.	Amazon Timestream unterstützt LiveAnalytics derzeit Batch-Load-Funktionen.	24. Februar 2023
Amazon Timestream unterstützt LiveAnalytics AWS Backup derzeit.	Amazon Timestream unterstützt LiveAnalytics AWS Backup derzeit.	14. Dezember 2022

Amazon Timestream für LiveAnalytics Aktualisierungen AWS verwalteter Richtlinien	Neue Informationen zu AWS verwalteten Richtlinien und Amazon Timestream für LiveAnalytics, einschließlich Aktualisierungen vorhandener verwalteter Richtlinien.	29. November 2021
Amazon Timestream for LiveAnalytics unterstützt geplante Abfragen	Amazon Timestream unterstützt LiveAnalytics derzeit die Ausführung einer Abfrage in Ihrem Namen, basierend auf einem Zeitplan.	29. November 2021
Amazon Timestream for LiveAnalytics unterstützt Magnetic Store.	Amazon Timestream unterstützt LiveAnalytics derzeit die Verwendung von Magnetspeicher für Ihre Tabellenschreibvorgänge.	29. November 2021
Amazon Timestream für Datensätze LiveAnalytics mit mehreren Messwerten.	Amazon Timestream unterstützt LiveAnalytics derzeit ein kompakteres Format zum Speichern Ihrer Zeitreihendaten.	29. November 2021
Amazon Timestream für LiveAnalytics Aktualisierungen AWS verwalteter Richtlinien	Neue Informationen zu AWS verwalteten Richtlinien und Amazon Timestream für LiveAnalytics, einschließlich Aktualisierungen vorhandener verwalteter Richtlinien.	24. Mai 2021
Amazon Timestream for LiveAnalytics ist jetzt in der Region Europa (Frankfurt) verfügbar.	Amazon Timestream for LiveAnalytics ist jetzt in der Region Europa (Frankfurt) allgemein verfügbar (`eu-central-1`).	23. April 2021

Amazon Timestream for LiveAnalytics unterstützt jetzt VPC Endpoints ()AWS PrivateLink.	Amazon Timestream unterstützt LiveAnalytics derzeit die Verwendung von VPC Endpunkten ()AWS PrivateLink.	23. März 2021
Amazon Timestream unterstützt jetzt tabellenübergreifende Abfragen.	Sie können Amazon Timestream verwenden LiveAnalytics , um tabellenübergreifende Abfragen auszuführen.	10. Februar 2021
Amazon Timestream unterstützt LiveAnalytics derzeit erweiterte Statistiken zur Abfrageausführung.	Amazon Timestream unterstützt LiveAnalytics derzeit erweiterte Statistiken zur Abfrageausführung, wie z. B. die Menge der gescannten Daten.	10. Februar 2021
Amazon Timestream unterstützt LiveAnalytics derzeit erweiterte Zeitreihenfunktionen.	Sie können Amazon Timestream für verwenden LiveAnalytics , um SQL Abfragen mit erweiterten Zeitreihenfunktionen wie Ableitungen, Integralen und Korrelationen auszuführen.	10. Februar 2021
Amazon Timestream for LiveAnalytics ist jetzt HIPAA und ISO PCI konform.	Sie können Amazon Timestream jetzt für Workloads verwenden, LiveAnalytics für die eine HIPAAISO, und PCI -konforme Infrastruktur erforderlich ist.	27. Januar 2021

Amazon Timestream unterstützt LiveAnalytics derzeit Open-Source-Telegraf und Grafana.	Sie können jetzt Telegraf, den Plug-in-gesteuerten Open-Source-Serveragenten für die Erfassung und Berichterstattung von Metriken, und Grafana, die Open-Source-Analyse- und Überwachungsplattform für Datenbanken, mit Amazon Timestream für verwenden. LiveAnalytics	25. November 2020
Amazon Timestream for LiveAnalytics ist jetzt allgemein verfügbar.	Diese Dokumentation behandelt die erste Version von Amazon Timestream für LiveAnalytics.	30. September 2020

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.