aws

Guia do Desenvolvedor

Amazon Timestream

Amazon Timestream: Guia do Desenvolvedor

Table of Contents

Amazon Timestream para LiveAnalytics 1
Timestream para obter LiveAnalytics os principais benefícios 1
Timestream para casos de uso LiveAnalytics 2
Começando a usar o Timestream para LiveAnalytics 2
Como funciona 3
Conceitos 4
Arquitetura 6
Escreve 11
Armazenamento 27
Consultas 28
Consultas programadas 33
Unidade de computação Timestream () TCU 33
Acessando o Timestream para LiveAnalytics 38
..... 38
Usar o console 43
Usando o AWS CLI 49
Usando o API 52
Usando o AWS SDKs 56
Conceitos básicos 61
Tutorial 61
Aplicação de exemplo 63
Exemplos de código 65
Write SDK o cliente 66
SDKCliente de consulta 69
Criar banco de dados 70
Descreva o banco 74
Atualizar um banco de dados 78
Excluir banco de dados 83
Listar bancos de dados 87
Create table 91
Descreva a tabela 100
Atualizar tabela 104
Excluir tabela 108
Listar tabelas 112

Gravar dados ... 117
Executar consulta ... 172
Executar UNLOAD consulta ... 198
Cancelar consulta ... 220
Criar tarefa de carregamento em lote ... 224
Descrever a tarefa de carregamento em lote ... 237
Listar tarefas de carregamento em lote ... 242
Retomar a tarefa de carregamento em lote ... 248
Criar consulta agendada ... 252
Listar consulta agendada ... 267
Descrever a consulta agendada ... 271
Executar consulta agendada ... 275
Atualizar consulta agendada ... 278
Excluir consulta agendada ... 281
Usando o carregamento em lote ... 284
Conceitos ... 285
Pré-requisitos ... 286
Práticas recomendadas ... 287
Preparando um arquivo de dados de carregamento em lote ... 288
Mapeamentos do modelo de dados ... 290
Usando o carregamento em lote com o console ... 294
Usando o carregamento em lote com o CLI ... 299
Usando o carregamento em lote com o SDKs ... 306
Usando relatórios de erro de carregamento em lote ... 306
Usando consultas agendadas ... 307
Benefícios ... 308
Casos de uso ... 309
Exemplo ... 309
Conceitos ... 310
Programar expressões ... 314
Mapeamentos do modelo de dados ... 318
Mensagens de notificação ... 337
Relatórios de erros ... 343
Padrões e exemplos ... 347
Usando UNLOAD ... 450
Benefícios ... 450

Casos de uso 451
Conceitos 451
Pré-requisitos 462
Práticas recomendadas 464
Exemplo de caso de uso 465
Limites 470
Usando insights de consulta 471
Benefícios 471
Otimizando o acesso aos dados 472
Habilitando insights de consulta no Amazon Timestream 476
Otimizar consultas 477
Trabalhando com AWS Backup 482
Como funciona 483
Criar backups 487
Como restaurar backups 488
Copiar backups 490
Excluir backups 490
Cotas e limites 491
Chaves de partição definidas pelo cliente 491
Usando chaves de partição definidas pelo cliente 492
Introdução às chaves de partição definidas pelo cliente 493
Verificando a configuração do esquema de particionamento 497
Atualizando a configuração do esquema de particionamento 502
Vantagens das chaves de partição definidas pelo cliente 506
Limitações das chaves de partição definidas pelo cliente 506
Chaves de partição definidas pelo cliente e dimensões de baixa cardinalidade 506
Criação de chaves de partição para tabelas existentes 507
Cronograma para validação de LiveAnalytics esquema com chaves de partição compostas personalizadas 507
Marcar recursos 510
Restrições de marcação 511
Operações de marcação 511
Segurança 513
Proteção de dados 514
Gerenciamento de identidade e acesso 517
Logging e monitoramento 557

Resiliência 560
Segurança da infraestrutura 561
Análise de configuração e vulnerabilidade 561
Resposta a incidentes 562
VPCendpoints 562
Melhores práticas de segurança 566
Como trabalhar com outros serviços 568
Amazon DynamoDB 569
AWS Lambda 569
AWS IoT Core 571
Amazon Managed Service for Apache Flink 575
Amazon Kinesis 577
Amazon MQ 585
Amazon MSK 586
Amazon QuickSight 589
Amazon SageMaker 593
Amazon SQS 595
DBeaver 596
Grafana 601
SquaredUp 602
Telegraf de código aberto 603
JDBC 608
ODBC 624
VPCendpoints 632
Práticas recomendadas 632
Modelagem de dados 633
Segurança 651
Configurando o Timestream para LiveAnalytics 652
Escreve 653
Consultas 655
Consultas programadas 656
Aplicativos cliente e integrações suportadas 657
Geral 658
Medição e otimização de custos 658
Escreve 658
Armazenamento 661

Consultas ... 662
Otimização de custo ... 662
Monitoramento com a Amazon CloudWatch ... 663
Solução de problemas ... 679
Manipulação de WriteRecords aceleradores ... 679
Lidando com registros rejeitados ... 680
Solução de problemas UNLOAD ... 680
Timestream para códigos de erro LiveAnalytics específicos ... 682
Cotas ... 684
Cotas padrão ... 684
Limites do serviço ... 686
Tipos de dados compatíveis ... 689
Carregamento em lote ... 689
Restrições de nomenclatura ... 690
Palavras-chave reservadas ... 692
Identificadores do sistema ... 695
UNLOAD ... 695
Referência da linguagem de consulta ... 695
Tipos de dados compatíveis ... 696
Funcionalidade de série temporal integrada ... 700
SQLapoio ... 715
Operadores lógicos ... 724
Operadores de comparação ... 725
Funções de comparação ... 726
Expressões condicionais ... 729
Funções de conversão ... 731
Operadores matemáticos ... 732
Funções matemáticas ... 732
Operadores de string ... 735
Funções de string ... 735
Operadores de matriz ... 740
Funções de array ... 740
Funções bitwise ... 748
Funções de expressões regulares ... 750
Operadores de data/hora ... 755
Funções de data/hora ... 757

Funções agregadas 775
Funções de janela 791
Consultas de exemplo 796
APIreferência 810
Ações 811
Tipos de dados 951
Erros comuns 1060
Parâmetros gerais 1061
Histórico do documentos 1064
Amazon Timestream para InfluxDB 1070
Instâncias de banco de dados 1070
Classes da instância de banco de dados 1072
Tipos de classe de instância de banco de dados 1072
Especificações de hardware 1072
Armazenamento de instâncias 1074
Tipos de armazenamento do InfluxDB 1074
Dimensionamento de instâncias 1075
Regiões e zonas de disponibilidade 1076
Disponibilidade das regiões 1077
Design de regiões 1078
Zonas de disponibilidade 1078
Faturamento 1078
Configuração 1079
Inscreva-se para AWS 1079
Configuração 1080
Determinar requisitos 1082
VPCacesso 1085
Conceitos básicos 1086
Criando e conectando-se a uma instância Timestream for InfluxDB 1087
Criando um novo token de operador para sua instância do InfluxDB 1100
Migração de dados do InfluxDB autogerenciado para o Timestream for InfluxDB 1101
Preparação 1101
Como usar o script 1103
Visão geral da migração 1105
Configurar uma instância de banco de dados 1109
Criar uma instância de banco de dados 1110

Configurações para instâncias de banco de dados 1113
Conectando-se a uma instância de banco de dados Amazon Timestream para InfluxDB ... 1117
Gerenciar instâncias de banco de dados 1155
Atualizando instâncias de banco de 1156
Manutenção de uma instância de banco de dados 1158
Excluir uma instância de banco de dados 1158
Implantações de instâncias de banco de dados multi-AZ 1160
Configuração para visualizar registros do InfluxDB em instâncias do Timestream Influxdb . 1164
Marcar recursos 1166
Restrições de marcação 1166
Melhores práticas para Timestream for InfluxDB 1167
Otimize gravações no InfluxDB 1167
Design para desempenho 1169
Solução de problemas 1171
Aviso de que a versão “dev” não é reconhecida 1171
A migração falhou durante a fase de restauração 1172
Diretrizes operacionais básicas do Amazon Timestream para InfluxDB 1172
RAMRecomendações de instâncias de banco 1173
Segurança 1173
Visão geral 1174
Autenticação de banco de dados com Amazon Timestream para InfluxDB 1178
Como o Timestream for InfluxDB usa segredos 1180
Proteção de dados 1187
Identity and Access Management 1189
Logging e monitoramento 1228
Validação de conformidade 1231
Resiliência 1233
Segurança da infraestrutura 1233
Configuração e análise de vulnerabilidade no Timestream for InfluxDB 1234
Resposta a incidentes 1235
Amazon Timestream para API InfluxDB e endpoints de interface () VPC AWS PrivateLink 1235
Melhores práticas de segurança 1238
APIreferência 1240
Histórico do documentos 1240
..... mccxlvii

Para que serve o Amazon LiveAnalytics Timestream?

O Amazon Timestream LiveAnalytics for é um banco de dados de séries temporais rápido, escalável, totalmente gerenciado e criado especificamente para facilitar o armazenamento e a análise de trilhões de pontos de dados de séries temporais por dia. O Timestream for LiveAnalytics economiza tempo e custos no gerenciamento do ciclo de vida dos dados de séries temporais, mantendo os dados recentes na memória e movendo os dados históricos para um nível de armazenamento com custo otimizado com base nas políticas definidas pelo usuário. O mecanismo de consulta específico LiveAnalytics do Timestream for permite que você acesse e analise dados recentes e históricos juntos, sem precisar especificar sua localização. O Amazon Timestream LiveAnalytics for tem funções integradas de análise de séries temporais, ajudando você a identificar tendências e padrões em seus dados quase em tempo real. O Timestream for LiveAnalytics é sem servidor e aumenta ou diminui automaticamente para ajustar a capacidade e o desempenho. Como você não precisa gerenciar a infraestrutura subjacente, você pode se concentrar em otimizar e criar seus aplicativos.

O Timestream for LiveAnalytics também se integra aos serviços comumente usados para coleta de dados, visualização e aprendizado de máquina. Você pode enviar dados para o Amazon Timestream LiveAnalytics para usar o Amazon Kinesis, a AWS IoT Core Amazon e o Telegraf de código MSK aberto. Você pode visualizar dados usando Amazon QuickSight, Grafana e ferramentas de business intelligence por meio de. JDBC Você também pode usar a Amazon SageMaker com o Timestream LiveAnalytics para aprendizado de máquina.

Timestream para obter LiveAnalytics os principais benefícios

Os principais benefícios do Amazon LiveAnalytics Timestream para são:

- Sem servidor com auto-scaling — Com o Amazon Timestream for, não há servidores para gerenciar nem capacidade LiveAnalytics para provisionar. Conforme as necessidades de seu aplicativo mudam, o Timestream for escalável LiveAnalytics automaticamente para ajustar a capacidade.
- Gerenciamento do ciclo de vida dos dados — O Amazon Timestream LiveAnalytics for simplifica o processo complexo de gerenciamento do ciclo de vida dos dados. Ele oferece armazenamento em camadas, com um armazenamento de memória para dados recentes e um armazenamento magnético para dados históricos. O Amazon Timestream automatiza a transferência de dados do armazenamento de memória para o armazenamento magnético com base em políticas configuráveis pelo usuário.

- Acesso simplificado aos dados — Com o Amazon Timestream LiveAnalytics for, você não precisa mais usar ferramentas diferentes para acessar dados recentes e históricos. O mecanismo de consulta criado especificamente pelo Amazon Timestream LiveAnalytics for acessa e combina dados de forma transparente em todos os níveis de armazenamento sem que você precise especificar a localização dos dados.
- Criado especificamente para séries temporais - Você pode analisar rapidamente dados de séries temporais usando funções de séries temporais integradas para suavizaçãoSQL, aproximação e interpolação. O Timestream for LiveAnalytics também suporta agregados avançados, funções de janela e tipos de dados complexos, como matrizes e linhas.
- Sempre criptografado — o Amazon Timestream LiveAnalytics for garante que seus dados de séries temporais sejam sempre criptografados, estejam eles em repouso ou em trânsito. O Amazon Timestream LiveAnalytics for também permite que você especifique AWS KMS uma chave gerenciada pelo cliente CMK () para criptografar dados no armazenamento magnético.
- Alta disponibilidade — O Amazon Timestream garante a alta disponibilidade de suas solicitações de gravação e leitura ao replicar dados automaticamente e alocar recursos em pelo menos três zonas de disponibilidade diferentes em uma única região. AWS Para obter mais informações, consulte o Acordo de Nível de Serviço Timestream.
- Durabilidade — O Amazon Timestream garante a durabilidade de seus dados ao replicar automaticamente seus dados de memória e armazenamento magnético em diferentes zonas de disponibilidade em uma única região. AWS Todos os seus dados são gravados em disco antes de confirmar que sua solicitação de gravação foi concluída.

Timestream para casos de uso LiveAnalytics

Exemplos de uma lista crescente de casos de uso do Timestream para LiveAnalytics incluem:

- Métricas de monitoramento para melhorar o desempenho e a disponibilidade de seus aplicativos.
- Armazenamento e análise de telemetria industrial para agilizar o gerenciamento e a manutenção de equipamentos.
- Rastreamento da interação do usuário com um aplicativo ao longo do tempo.
- Armazenamento e análise de dados de sensores de IoT.

Começando a usar o Timestream para LiveAnalytics

Recomendamos que você comece lendo as seguintes seções:

- Tutorial- Para criar um banco de dados preenchido com conjuntos de dados de amostra e executar consultas de amostra.
- Amazon LiveAnalytics Timestream para conceitos- Aprender o Timestream essencial para LiveAnalytics conceitos.
- Acessando o Timestream para LiveAnalytics- Para saber como acessar o Timestream para LiveAnalytics usar o console AWS CLI, ou. API
- Cotas- Para saber mais sobre cotas no número de Timestream para LiveAnalytics componentes que você pode provisionar.

Para saber como começar rapidamente a desenvolver aplicativos para o Timestream for LiveAnalytics, veja o seguinte:

- Usando o AWS SDKs
- Referência da linguagem de consulta

Como funciona

As seções a seguir fornecem uma visão geral dos componentes do serviço Amazon Timestream for Live Analytics e de como eles interagem.

Depois de ler esta introdução, consulte as Acessando o Timestream para LiveAnalytics seções para saber como acessar o Timestream for Live Analytics usando o console, AWS CLI, ou. SDKs

Tópicos

- Amazon LiveAnalytics Timestream para conceitos
- Arquitetura
- Escreve
- Armazenamento
- Consultas
- Consultas programadas
- Unidade de computação Timestream () TCU

Amazon LiveAnalytics Timestream para conceitos

Os dados da série temporal são uma sequência de pontos de dados registrados em um intervalo de tempo. Esse tipo de dado é usado para medir eventos que mudam com o tempo. Os exemplos incluem.

- Preços das ações ao longo do tempo
- Medições de temperatura ao longo do tempo
- CPUutilização de uma EC2 instância ao longo do tempo

Com dados de séries temporais, cada ponto de dados consiste em um carimbo de data/hora, um ou mais atributos e o evento que muda com o tempo. Esses dados podem ser usados para obter insights sobre o desempenho e a integridade de um aplicativo, detectar anomalias e identificar oportunidades de otimização. Por exemplo, DevOps os engenheiros podem querer visualizar dados que meçam as mudanças nas métricas de desempenho da infraestrutura. Talvez os fabricantes queiram rastrear os dados do sensor de IoT que medem as mudanças nos equipamentos em uma instalação. Os profissionais de marketing on-line podem querer analisar dados de fluxo de cliques que capturem como um usuário navega em um site ao longo do tempo. Como os dados de séries temporais são gerados de várias fontes em volumes extremamente altos, eles precisam ser coletados de forma econômica quase em tempo real e, portanto, exigem um armazenamento eficiente que ajude a organizar e analisar os dados.

A seguir estão os principais conceitos do Timestream for. LiveAnalytics

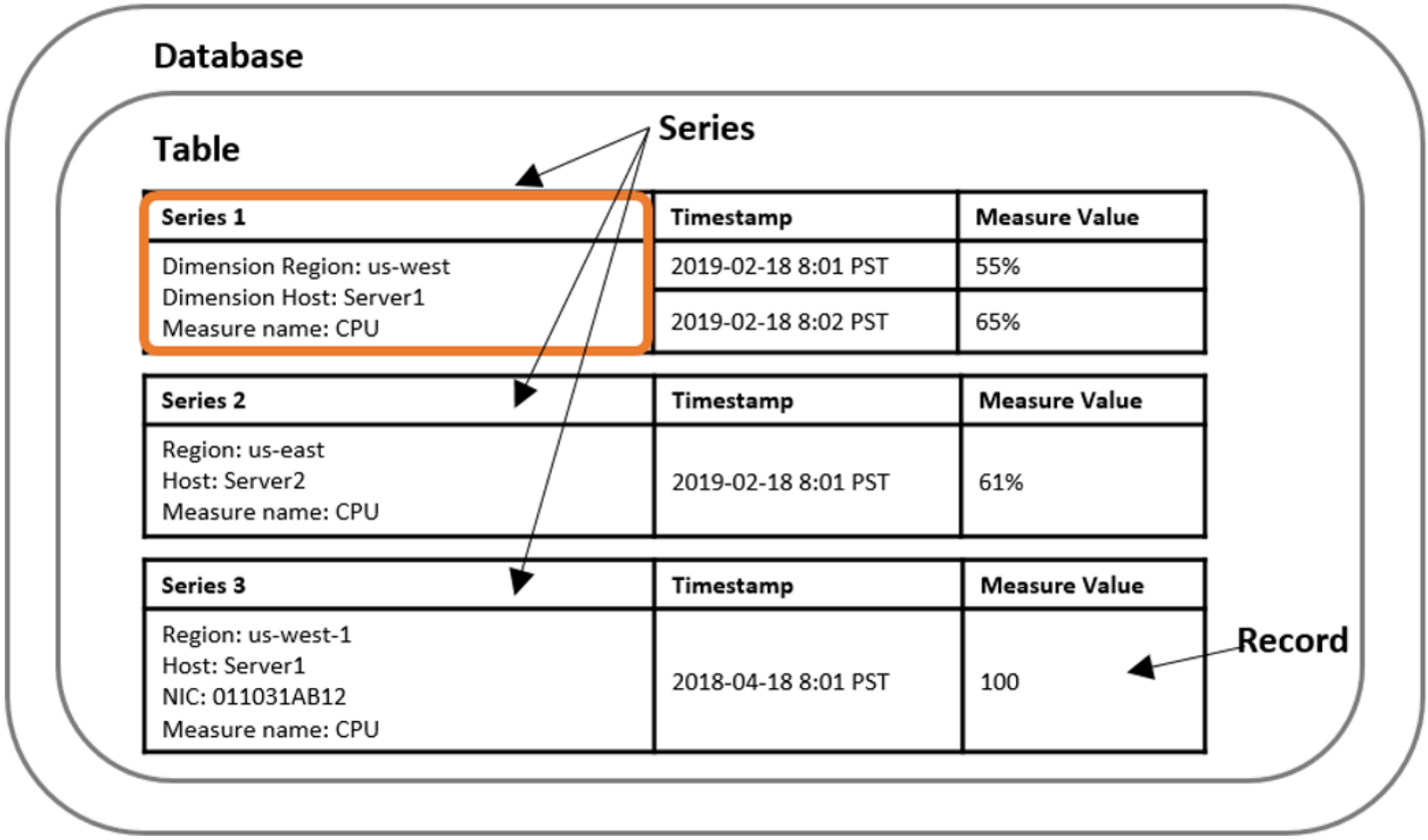
- Séries temporais - Uma sequência de um ou mais pontos de dados (ou registros) gravados em um intervalo de tempo. Exemplos são o preço de uma ação ao longo do tempo, a utilização da CPU memória de uma EC2 instância ao longo do tempo e a leitura de temperatura/pressão de um sensor de IoT ao longo do tempo.
- Registro - Um único ponto de dados em uma série temporal.
- Dimensão - Um atributo que descreve os metadados de uma série temporal. Uma dimensão consiste em um nome e um valor de dimensão. Considere os seguintes exemplos:
  - Ao considerar uma bolsa de valores como uma dimensão, o nome da dimensão é “bolsa de valores” e o valor da dimensão é "NYSE”
  - Ao considerar uma AWS região como uma dimensão, o nome da dimensão é “região” e o valor da dimensão é “us-east-1"

  - Para um sensor de IoT, o nome da dimensão é “ID do dispositivo” e o valor da dimensão é “12345"
- Medida - O valor real que está sendo medido pelo registro. Exemplos são o preço das ações, a CPU utilização da memória e a leitura de temperatura ou umidade. As medidas consistem em nomes e valores de medidas. Considere os seguintes exemplos:
  - Para o preço de uma ação, o nome da medida é “preço da ação” e o valor da medida é o preço real da ação em um determinado momento.
  - Para CPU utilização, o nome da medida é “CPUutilização” e o valor da medida é a utilização realCPU.

  As medidas podem ser modeladas no Timestream LiveAnalytics como registros de várias medidas ou medidas únicas. Para obter mais informações, consulte Registros de várias medidas versus registros de medida única.
- Timestamp - Indica quando uma medida foi coletada para um determinado registro. Timestream for LiveAnalytics suporta registros de data e hora com granularidade de nanossegundos.
- Tabela - Um contêiner para um conjunto de séries temporais relacionadas.
- Banco de dados - Um contêiner de nível superior para tabelas.

Um resumo do Timestream para conceitos LiveAnalytics

Um banco de dados contém 0 ou mais tabelas. Cada tabela contém 0 ou mais séries temporais. Cada série temporal consiste em uma sequência de registros em um determinado intervalo de tempo em uma granularidade especificada. Cada série temporal pode ser descrita usando seus metadados ou dimensões, seus dados ou medidas e seus registros de data e hora.

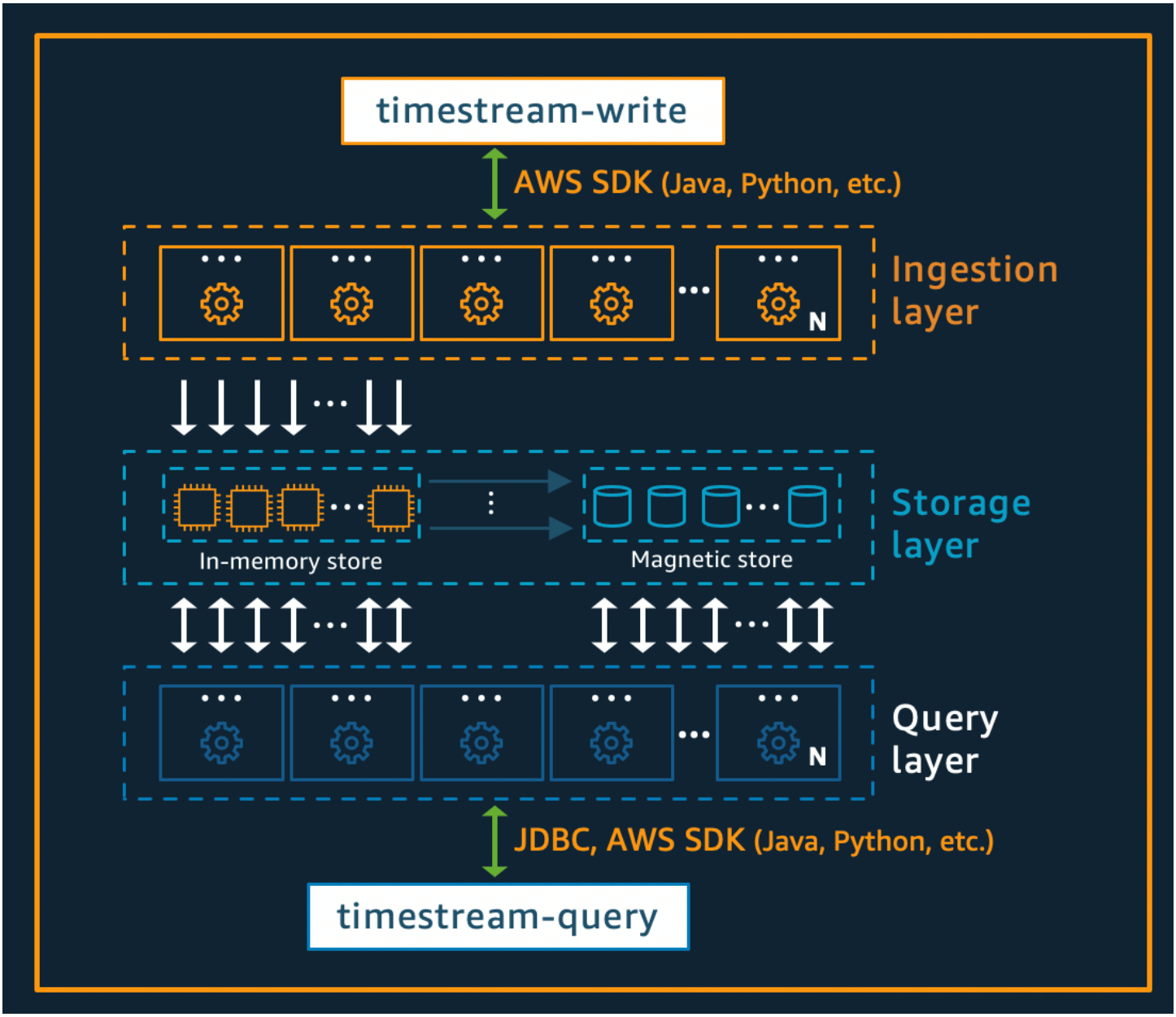


Arquitetura

O Amazon Timestream for Live Analytics foi projetado desde o início para coletar, armazenar e processar dados de séries temporais em grande escala. Sua arquitetura sem servidor oferece suporte a sistemas totalmente desacoplados de ingestão, armazenamento e processamento de consultas de dados que podem ser escalados de forma independente. Esse design simplifica cada subsistema, facilitando a obtenção de confiabilidade inabalável, eliminando gargalos de escala e reduzindo as chances de falhas correlacionadas do sistema. Cada um desses fatores se torna mais importante à medida que o sistema se expande.

Tópicos

- Arquitetura de escrita
- Arquitetura de armazenamento
- Arquitetura de consulta
- Arquitetura celular



Arquitetura de escrita

Ao gravar dados de séries temporais, o Amazon Timestream for Live Analytics encaminha gravações para uma tabela, partição, para uma instância de armazenamento de memória tolerante a falhas que processa gravações de dados de alto rendimento. O armazenamento de memória, por sua vez, obtém durabilidade em um sistema de armazenamento separado que replica os dados em três zonas de disponibilidade ()AZs. A replicação é baseada em quórum, de forma que a perda de nós, ou de uma AZ inteira, não interrompa a disponibilidade de gravação. Quase em tempo real, outros nós de armazenamento na memória são sincronizados com os dados para atender às consultas. Os nós de réplica do leitor AZs também se estendem, para garantir a alta disponibilidade de leitura.

O Timestream for Live Analytics suporta a gravação de dados diretamente no armazenamento magnético, para aplicativos que geram menor taxa de transferência de dados que chegam tardiamente. Os dados de chegada tardia são dados com um registro de data e hora anterior à hora atual. Semelhante às gravações de alto rendimento no armazenamento de memória, os dados gravados no armazenamento magnético são replicados em três AZs e a replicação é baseada em quórum.

Independentemente de os dados serem gravados na memória ou no armazenamento magnético, o Timestream for Live Analytics indexa e particiona automaticamente os dados antes de gravá-los no armazenamento. Uma única tabela Timestream for Live Analytics pode ter centenas, milhares ou até milhões de partições. Partições individuais não se comunicam diretamente entre si e não compartilham nenhum dado (arquitetura sem compartilhamento). Em vez disso, o particionamento de uma tabela é rastreado por meio de um serviço de indexação e rastreamento de partições altamente disponível. Isso fornece outra separação de preocupações projetada especificamente para minimizar o efeito das falhas no sistema e tornar as falhas correlacionadas muito menos prováveis.

Arquitetura de armazenamento

Quando os dados são armazenados no Timestream for Live Analytics, os dados são organizados em ordem temporal e ao longo do tempo, com base nos atributos de contexto gravados com os dados. Ter um esquema de particionamento que divide o “espaço” além do tempo é importante para escalar massivamente um sistema de séries temporais. Isso ocorre porque a maioria dos dados de séries temporais é gravada na hora atual ou em torno dela. Como resultado, o particionamento apenas por tempo não faz um bom trabalho ao distribuir o tráfego de gravação ou permitir a remoção eficaz dos dados no momento da consulta. Isso é importante para o processamento de séries temporais em escala extrema e permitiu que o Timestream for Live Analytics escalasse ordens de magnitude mais altas do que os outros sistemas líderes do mercado atualmente, sem servidor. As partições resultantes são chamadas de “blocos” porque representam divisões de um espaço bidimensional (projetadas para serem de tamanho semelhante). As tabelas do Timestream for Live Analytics começam como uma única partição (bloco) e depois são divididas na dimensão espacial conforme a taxa de transferência exigida. Quando os blocos atingem um determinado tamanho, eles se dividem na dimensão do tempo para obter um melhor paralelismo de leitura à medida que o tamanho dos dados aumenta.

O Timestream for Live Analytics foi projetado para gerenciar automaticamente o ciclo de vida dos dados de séries temporais. O Timestream for Live Analytics oferece dois armazenamentos de dados: um armazenamento na memória e um armazenamento magnético econômico. Ele também oferece suporte à configuração de políticas em nível de tabela para transferir dados automaticamente entre

lojas. As gravações de dados de alta taxa de transferência recebidas chegam ao armazenamento de memória, onde os dados são otimizados para gravações, bem como as leituras realizadas no horário atual para alimentar consultas do tipo painel e alertas. Quando o prazo principal para as necessidades de gravação, alertas e painéis tiver passado, permitindo que os dados fluam automaticamente do armazenamento de memória para o armazenamento magnético para otimizar os custos. O Timestream for Live Analytics permite definir uma política de retenção de dados no armazenamento de memória para essa finalidade. As gravações de dados para dados que chegam tarde são gravadas diretamente no armazenamento magnético.

Quando os dados estão disponíveis no armazenamento magnético (devido à expiração do período de retenção do armazenamento de memória ou devido a gravações diretas no armazenamento magnético), eles são reorganizados em um formato altamente otimizado para leituras de dados de grande volume. O armazenamento magnético também tem uma política de retenção de dados que pode ser configurada se houver um limite de tempo em que os dados percam sua utilidade. Quando os dados excedem o intervalo de tempo definido para a política de retenção do armazenamento magnético, eles são removidos automaticamente. Portanto, com o Timestream for Live Analytics, além de algumas configurações, o gerenciamento do ciclo de vida dos dados ocorre perfeitamente nos bastidores.

Arquitetura de consulta

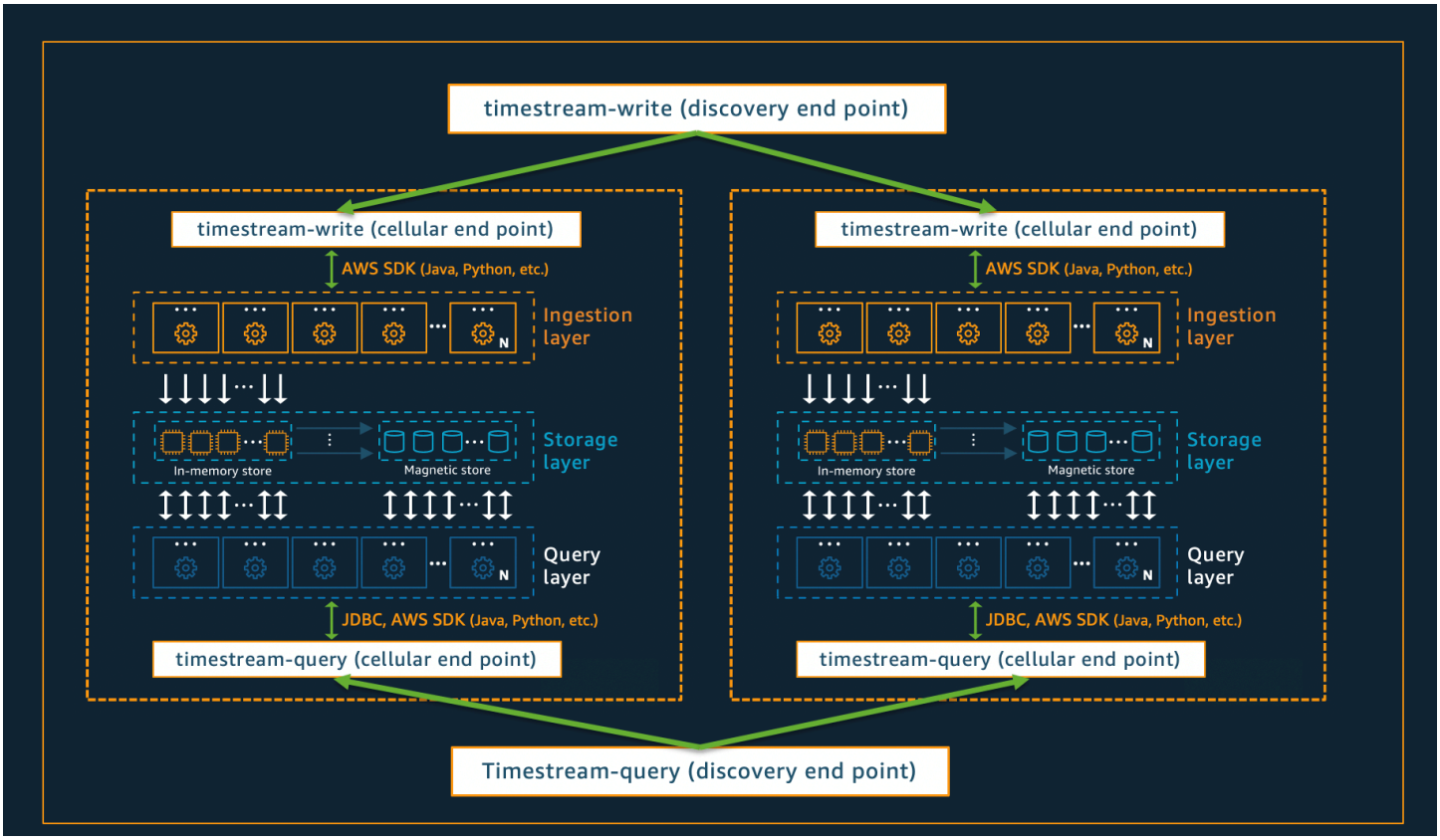
As consultas do Timestream for Live Analytics são expressas em uma SQL gramática que tem extensões para suporte específico de séries temporais (tipos e funções de dados específicos de séries temporais), portanto, a curva de aprendizado é fácil para desenvolvedores que já estão familiarizados. SQL Em seguida, as consultas são processadas por um mecanismo de consulta distribuído e adaptável que usa metadados do serviço de rastreamento e indexação de blocos para acessar e combinar facilmente os dados entre os armazenamentos de dados no momento em que a consulta é emitida. Isso proporciona uma experiência que ressoa bem com os clientes, pois reúne muitas das complexidades de Rube Goldberg em uma abstração de banco de dados simples e familiar.

As consultas são executadas por uma frota dedicada de trabalhadores, em que o número de trabalhadores alistados para executar uma determinada consulta é determinado pela complexidade da consulta e pelo tamanho dos dados. O desempenho de consultas complexas em grandes conjuntos de dados é obtido por meio de um paralelismo massivo, tanto na frota de tempo de execução de consultas quanto nas frotas de armazenamento do sistema. A capacidade de analisar grandes quantidades de dados com rapidez e eficiência é um dos maiores pontos fortes do

Timestream for Live Analytics. Uma única consulta que executa mais de terabytes ou mesmo petabytes de dados pode ter milhares de máquinas trabalhando em tudo ao mesmo tempo.

Arquitetura celular

Para garantir que o Timestream for Live Analytics possa oferecer uma escala praticamente infinita para seus aplicativos e, ao mesmo tempo, garantir 99,99% de disponibilidade, o sistema também foi projetado usando uma arquitetura celular. Em vez de escalar o sistema como um todo, o Timestream for Live Analytics segmenta em várias cópias menores de si mesmo, chamadas de células. Isso permite que as células sejam testadas em grande escala e evita que um problema no sistema em uma célula afete a atividade em qualquer outra célula em uma determinada região. Embora o Timestream for Live Analytics tenha sido projetado para oferecer suporte a várias células por região, considere o seguinte cenário fictício, no qual há 2 células em uma região.



No cenário descrito acima, a ingestão de dados e as solicitações de consulta são processadas primeiro pelo endpoint de descoberta para ingestão e consulta de dados, respectivamente. Em seguida, o endpoint de descoberta identifica a célula que contém os dados do cliente e direciona a solicitação para o endpoint de ingestão ou consulta apropriado para essa célula. Ao usar oSDKs, essas tarefas de gerenciamento de endpoints são tratadas de forma transparente para você.

> **Note**
>
> Ao usar VPC endpoints com o Timestream for Live Analytics ou acessar diretamente REST API as operações do Timestream for Live Analytics, você precisará interagir diretamente com os endpoints celulares. Para obter orientação sobre como fazer isso, consulte VPCEndpoints para obter instruções sobre como configurar VPC endpoints e Endpoint Discovery Pattern para obter instruções sobre a invocação direta das operações. REST API

Escreve

Você pode coletar dados de séries temporais de dispositivos conectados, sistemas de TI e equipamentos industriais e gravá-los no Timestream for Live Analytics. O Timestream for Live Analytics permite que você grave pontos de dados de uma única série temporal e/ou pontos de dados de várias séries em uma única solicitação de gravação quando a série temporal pertence à mesma tabela. Para sua conveniência, o Timestream for Live Analytics oferece um esquema flexível que detecta automaticamente os nomes das colunas e os tipos de dados das tabelas do Timestream for Live Analytics com base nos nomes das dimensões e nos tipos de dados dos valores de medida que você especifica ao invocar gravações no banco de dados. Você também pode gravar lotes de dados no Timestream for Live Analytics.

> **Note**
>
> O Timestream for Live Analytics oferece suporte à semântica de consistência eventual para leituras. Isso significa que quando você consulta dados imediatamente após gravar um lote de dados no Timestream for Live Analytics, os resultados da consulta podem não refletir os resultados de uma operação de gravação concluída recentemente. Os resultados também podem incluir alguns dados obsoletos. Da mesma forma, ao gravar dados de séries temporais com uma ou mais novas dimensões, uma consulta pode retornar um subconjunto parcial de colunas por um curto período de tempo. Se você repetir essas solicitações de consulta após um curto período de tempo, os resultados deverão retornar os dados mais recentes.

Você pode gravar dados usando o AWS SDKs, AWS CLI, ou por meio de AWS Lambda AWS IoT CoreAmazon Managed Service for Apache Flink,Amazon Kinesis,Amazon MSK,, Telegraf de código aberto e.

Tópicos

- Tipos de dados
- Nenhuma definição de esquema inicial
- Gravando dados (inserções e acréscimos)
- Consistência eventual para leituras
- Gravações em lote com WriteRecords API
- Carregamento em lote
- Escolhendo entre a WriteRecords API operação e a carga em lote

Tipos de dados

O Timestream for Live Analytics oferece suporte aos seguintes tipos de dados para gravações.

Tipo de dados	Descrição
BIGINT	Representa um inteiro assinado de 64 bits.
BOOLEAN	Representa os dois valores verdadeiros da lógica, a saber, verdadeiro e falso.
DOUBLE	Precisão variável de 64 bits implementando o IEEE Padrão 754 para Aritmética Binária de Ponto Flutuante. **Note** Existem funções de linguagem de consulta `Infinity` e valores `NaN` duplos que podem ser usados em consultas. Mas você não pode gravar esses valores no Timestream.
VARCHAR	Dados de caracteres de comprimento variável com um comprimento máximo opcional. O limite máximo é de 2 KB.
MULTI	Tipo de dados para registros de várias medidas. Esse tipo de dados inclui uma ou mais medidas do tipo `BIGINT` `BOOLEANDOUBLE`,`VARCHAR`,, `TIMESTAMP` e.

Tipo de dados	Descrição
TIMESTAMP	Representa uma instância no tempo usando o tempo de entrada de precisão de nanossegundosUTC, rastreando o tempo desde a hora do Unix. Atualmente, esse tipo de dados é suportado somente para registros de várias medidas (ou seja, dentro de valores de medida do tipo`MULTI`). *YYYY*-*MM*-*DD* *hh*:*mm*:*ss*.*sssssssss* Grava carimbos de data/hora de suporte na faixa de. `1970-01-01 00:00:00.000000000` `2262-04-11 23:47:16.854775807`

Nenhuma definição de esquema inicial

Antes de enviar dados para o Amazon Timestream for Live Analytics, você deve criar um banco de dados e uma tabela usando AWS Management Console as operações Timestream for Live Analytics ou Timestream for Live SDKs Analytics. API Para ter mais informações, consulte Criar um banco de dados do e Criar uma tabela. Ao criar a tabela, você não precisa definir o esquema antecipadamente. O Amazon Timestream for Live Analytics detecta automaticamente o esquema com base nas medidas e dimensões dos pontos de dados enviados, para que você não precise mais alterar seu esquema off-line para adaptá-lo aos dados de séries temporais que mudam rapidamente.

Gravando dados (inserções e acréscimos)

A operação de gravação no Amazon Timestream for Live Analytics permite que você insira e atualize dados. Por padrão, as gravações no Amazon Timestream for Live Analytics seguem a semântica do primeiro escritor, em que os dados são armazenados somente como acréscimos e os registros duplicados são rejeitados. Embora a semântica do primeiro escritor ganha satisfaça os requisitos de muitos aplicativos de séries temporais, há cenários em que os aplicativos precisam atualizar os registros existentes de maneira idempotente e/ou gravar dados com a semântica do último gravador ganha, em que o registro com a versão mais alta é armazenado no serviço. Para lidar com esses cenários, o Amazon Timestream for Live Analytics oferece a capacidade de alterar dados. Upsert é uma operação que insere um registro no sistema quando o registro não existe ou atualiza o registro quando existe um. Quando o registro é atualizado, ele é atualizado de forma idempotente.

Não há uma operação em nível de registro para exclusão. Mas tabelas e bancos de dados podem ser excluídos.

Gravando dados no armazenamento de memória e no armazenamento magnético

O Amazon Timestream for Live Analytics oferece a capacidade de gravar dados diretamente no armazenamento de memória e no armazenamento magnético. O armazenamento de memória é otimizado para gravações de dados de alta taxa de transferência e o armazenamento magnético é otimizado para gravações de dados de chegada tardia com menor taxa de transferência.

Os dados de chegada tardia são dados com um registro de data e hora anterior à hora atual e fora do período de retenção do armazenamento de memória. Você deve habilitar explicitamente a capacidade de gravar dados que chegam tarde no armazenamento magnético habilitando gravações do armazenamento magnético para a tabela. Além disso, `MagneticStoreRejectedDataLocation` é definido quando uma tabela é criada. Para gravar no armazenamento magnético, os chamadores `WriteRecords` devem ter `S3:PutObject` permissões para o bucket do S3 especificado `MagneticStoreRejectedDataLocation` durante a criação da tabela. Para obter mais informações CreateTable, consulte WriteRecords, PutObjecte.

Gravando dados com registros de medida única e registros de várias medidas

O Amazon Timestream for Live Analytics oferece a capacidade de gravar dados usando dois tipos de registros, a saber, registros de medida única e registros de várias medidas.

Registros de medida única

Os registros de medida única permitem que você envie uma única medida por registro. Quando os dados são enviados para o Timestream for Live Analytics usando esse formato, o Timestream for Live Analytics cria uma linha de tabela por registro. Isso significa que, se um dispositivo emitir 4 métricas e cada métrica for enviada como um registro de medida única, o Timestream for Live Analytics criará 4 linhas na tabela para armazenar esses dados e os atributos do dispositivo serão repetidos para cada linha. Esse formato é recomendado nos casos em que você deseja monitorar uma única métrica de um aplicativo ou quando seu aplicativo não emite várias métricas ao mesmo tempo.

Registros de várias medidas

Com registros de várias medidas, você pode armazenar várias medidas em uma única linha da tabela, em vez de armazenar uma medida por linha da tabela. Portanto, os registros de várias

medidas permitem que você migre seus dados existentes de bancos de dados relacionais para o Amazon Timestream for Live Analytics com o mínimo de alterações.

Você também pode agrupar mais dados em uma única solicitação de gravação do que registros de medida única. Isso aumenta a taxa de transferência e o desempenho da gravação de dados, além de reduzir o custo da gravação de dados. Isso ocorre porque agrupar mais dados em uma solicitação de gravação permite que o Amazon Timestream for Live Analytics identifique mais dados repetíveis em uma única solicitação de gravação (quando aplicável) e cobre apenas uma vez por dados repetidos.

Tópicos

- Registros de várias medidas
- Gravando dados com um carimbo de data/hora que existe no passado ou no futuro

Registros de várias medidas

Com registros de várias medidas, você pode armazenar seus dados de séries temporais em um formato mais compacto na memória e no armazenamento magnético, o que ajuda a reduzir os custos de armazenamento de dados. Além disso, o armazenamento compacto de dados permite escrever consultas mais simples para recuperação de dados, melhora o desempenho das consultas e reduz o custo das consultas.

Além disso, os registros de várias medidas também suportam o tipo de TIMESTAMP dados para armazenar mais de um timestamp em um registro de série temporal. TIMESTAMPatributos em um registro de várias medidas oferecem suporte a registros de data e hora no futuro ou no passado. Portanto, os registros de várias medidas ajudam a melhorar o desempenho, o custo e a simplicidade da consulta, além de oferecer mais flexibilidade para armazenar diferentes tipos de medidas correlacionadas.

Benefícios

A seguir estão os benefícios do uso de registros de várias medidas.

- Desempenho e custo — os registros de várias medidas permitem que você grave várias medidas de séries temporais em uma única solicitação de gravação. Isso aumenta a taxa de transferência de gravação e também reduz o custo das gravações. Com registros de várias medidas, você pode armazenar dados de forma mais compacta, o que ajuda a reduzir os custos de armazenamento de dados. O armazenamento compacto de dados de registros de várias medidas resulta em menos dados sendo processados por consultas. Isso foi projetado para melhorar o desempenho geral da consulta e ajudar a reduzir o custo da consulta.

- Simplicidade da consulta — com registros de várias medidas, você não precisa escrever expressões de tabela comuns complexas (CTEs) em uma consulta para ler várias medidas com o mesmo timestamp. Isso ocorre porque as medidas são armazenadas como colunas em uma única linha da tabela. Portanto, registros de várias medidas permitem escrever consultas mais simples.
- Flexibilidade de modelagem de dados — você pode gravar timestamps futuros no Timestream for Live Analytics usando o tipo de TIMESTAMP dados e os registros de várias medidas. Um registro de várias medidas pode ter vários atributos do tipo de TIMESTAMP dados, além do campo de hora em um registro. TIMESTAMPatributos, em um registro de várias medidas, podem ter carimbos de data/hora no futuro ou no passado e se comportar como o campo de hora, exceto que o Timestream for Live Analytics não indexa os valores de tipo em um registro de várias medidas. TIMESTAMP

Casos de uso

Você pode usar registros de várias medidas para qualquer aplicativo de série temporal que gere mais de uma medição do mesmo dispositivo a qualquer momento. A seguir estão alguns exemplos de aplicativos.

- Uma plataforma de streaming de vídeo que gera centenas de métricas em um determinado momento.
- Dispositivos médicos que geram medições como níveis de oxigênio no sangue, frequência cardíaca e pulso.
- Equipamentos industriais, como plataformas de petróleo, que geram métricas, sensores de temperatura e clima.
- Outros aplicativos que são arquitetados com um ou mais microsserviços.

Exemplo: monitoramento do desempenho e da integridade de um aplicativo de streaming de vídeo

Considere um aplicativo de streaming de vídeo que esteja sendo executado em 200 EC2 instâncias. Você quer usar o Amazon Timestream for Live Analytics para armazenar e analisar as métricas emitidas pelo aplicativo, para que você possa entender o desempenho e a integridade do seu aplicativo, identificar rapidamente anomalias, resolver problemas e descobrir oportunidades de otimização.

Modelaremos esse cenário com registros de medida única e registros de várias medidas e, em seguida, compararemos e contrastaremos as duas abordagens. Para cada abordagem, fazemos as seguintes suposições.

- Cada EC2 instância emite quatro medidas (video_startup_time, rebuffering_ratio, video_playback_failures e average_frame_rate) e quatro dimensões (device_id, device_type, os_version e region) por segundo.
- Você deseja armazenar 6 horas de dados no armazenamento de memória e 6 meses de dados no armazenamento magnético.
- Para identificar anomalias, você configurou 10 consultas que são executadas a cada minuto para identificar qualquer atividade incomum nos últimos minutos. Você também criou um painel com oito widgets que exibem as últimas 6 horas de dados, para que você possa monitorar seu aplicativo com eficiência. Esse painel é acessado por cinco usuários a qualquer momento e é atualizado automaticamente a cada hora.

Usando registros de medida única

Modelagem de dados: com registros de medida única, criaremos um registro para cada uma das quatro medidas (tempo de inicialização do vídeo, taxa de rebuffer, falhas na reprodução do vídeo e taxa média de quadros). Cada registro terá as quatro dimensões (device_id, device_type, os_version e region) e um timestamp.

Gravações: quando você grava dados no Amazon Timestream for Live Analytics, os registros são construídos da seguinte forma.

```
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);
    }
  }
```

Quando você armazena registros de medida única, os dados são representados logicamente da seguinte forma.

Tempo	id_dispositivo	tipo_de_dispositivo	os_version	região	nome_medida	valor_medida::bigint	valor_medida::duplo
2021-09-07 21:48:44.00	12345678	iPhone 11	14,8	us-east-1	hora de inicialização do vídeo	200	
2021-09-07 21:48:44.00	12345678	iPhone 11	14,8	us-east-1	rácio_de_armazenamento em buffer		0,5
2021-09-07 21:48:44.00	12345678	iPhone 11	14,8	us-east-1	falhas na reprodução de vídeo	0	
2021-09-07 21:48:44.00	12345678	iPhone 11	14,8	us-east-1	taxa_de_quadro média		0,85
2021-09-07	12345678	iPhone 11	14,8	us-east-1	hora de inicializ	500	

Tempo	id_dispositivo	tipo_de_dispositivo	os_version	região	nome_medida	valor_medida::bigint	valor_medida::duplo
21:53:44 .00					ação do vídeo		
2021-09-07 21:53:44 .00	12345678	iPhone 11	14,8	us-east-1	rácio_de_armazenamento em buffer		1.5
2021-09-07 21:53:44 .00	12345678	iPhone 11	14,8	us-east-1	falhas na reprodução de vídeo	10	
2021-09-07 21:53:44 .00	12345678	iPhone 11	14,8	us-east-1	taxa_de_quadro média		0.2

Consultas: você pode escrever uma consulta que recupere todos os pontos de dados com o mesmo carimbo de data/hora recebido nos últimos 15 minutos da seguinte maneira.

```
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
```

Custo da carga de trabalho: o custo dessa carga de trabalho é estimado em $373,23 por mês com registros de medida única

Usando registros de várias medidas

Modelagem de dados: com registros de várias medidas, criaremos um registro que contém todas as quatro medidas (tempo de inicialização do vídeo, taxa de rebuffer, falhas na reprodução do vídeo e taxa média de quadros), todas as quatro dimensões (device_id, device_type, os_version e region) e um timestamp.

Gravações: quando você grava dados no Amazon Timestream for Live Analytics, os registros são construídos da seguinte forma.

```
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);
```

```
    }
  }
```

Quando você armazena registros de várias medidas, os dados são representados logicamente da seguinte forma.

Tempo	id_dispos itivo	tipo_de_ ispositiv o	os_versio n	região	nome_m da	hora de inicializ ação do vídeo	rácio_de_ armazen ento em buffer	falhas na reproduç o de vídeo	taxa_de_q uadro média
2021-09-7 21:48:44 0	1234567	iPhone 11	14,8	us-east-1	métricas_ de vídeo	200	0,5	0	0,85
2021-09-7 21:53:44 0	1234567	iPhone 11	14,8	us-east-1	métricas_ de vídeo	500	1.5	10	0.2

Consultas: você pode escrever uma consulta que recupere todos os pontos de dados com o mesmo carimbo de data/hora recebido nos últimos 15 minutos da seguinte maneira.

```
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)
```

Custo da carga de trabalho: o custo da carga de trabalho é estimado em $127,43 com registros de várias medidas.

> **Note**
>
> Nesse caso, o uso de registros de várias medidas reduz o gasto mensal geral estimado em 2,5 vezes, com o custo de gravação de dados reduzido em 3,3 vezes, o custo de armazenamento reduzido em 3,3 vezes e o custo de consulta reduzido em 1,2 vezes.

Gravando dados com um carimbo de data/hora que existe no passado ou no futuro

O Timestream for Live Analytics oferece a capacidade de gravar dados com um carimbo de data/hora que fica fora da janela de retenção do armazenamento de memória por meio de alguns mecanismos diferentes.

- Gravações no armazenamento magnético — Você pode gravar dados que chegam tarde diretamente no armazenamento magnético por meio de gravações no armazenamento magnético. Para usar gravações de armazenamento magnético, você deve primeiro habilitar gravações de armazenamento magnético para uma tabela. Em seguida, você pode ingerir dados na tabela usando o mesmo mecanismo usado para gravar dados no armazenamento de memória. O Amazon Timestream for Live Analytics gravará automaticamente os dados no armazenamento magnético com base em seu registro de data e hora.

  > **Note**
  >
  > A write-to-read latência do armazenamento magnético pode ser de até 6 horas, ao contrário da gravação de dados no armazenamento de memória, onde a write-to-read latência está na faixa de menos de um segundo.

- TIMESTAMPtipo de dados para medidas — Você pode usar o tipo de TIMESTAMP dados para armazenar dados do passado, do presente ou do futuro. Um registro de várias medidas pode ter vários atributos do tipo de TIMESTAMP dados, além do campo de hora em um registro. TIMESTAMPatributos, em um registro de várias medidas, podem ter carimbos de data/hora no futuro ou no passado e se comportar como o campo de hora, exceto que o Timestream for Live Analytics não indexa os valores de tipo em um registro de várias medidas. TIMESTAMP

  > **Note**
  >
  > O tipo de TIMESTAMP dados é suportado somente para registros de várias medidas.

Consistência eventual para leituras

O Timestream for Live Analytics oferece suporte à semântica de consistência eventual para leituras. Isso significa que quando você consulta dados imediatamente após gravar um lote de dados no Timestream for Live Analytics, os resultados da consulta podem não refletir os resultados de uma operação de gravação concluída recentemente. Se você repetir essas solicitações de consulta após um curto período de tempo, os resultados deverão retornar os dados mais recentes.

Gravações em lote com WriteRecords API

O Amazon Timestream for Live Analytics permite que você grave pontos de dados de uma única série temporal e/ou pontos de dados de várias séries em uma única solicitação de gravação. O agrupamento em lotes de vários pontos de dados em uma única operação de gravação é benéfico do ponto de vista de desempenho e custo. Consulte a Escreve seção Medição e preços para obter mais detalhes.

> **Note**
>
> Suas solicitações de gravação para o Timestream for Live Analytics podem ser limitadas à medida que o Timestream for Live Analytics se adapta às necessidades de ingestão de dados do seu aplicativo. Se seus aplicativos encontrarem exceções de limitação, você deverá continuar enviando dados com a mesma taxa de transferência (ou maior) para permitir que o Timestream for Live Analytics seja escalado automaticamente de acordo com as necessidades do seu aplicativo.

Carregamento em lote

Com o carregamento em lote do Amazon Timestream LiveAnalytics for, você pode CSV ingerir arquivos armazenados no Amazon S3 no Timestream em lotes. Com essa nova funcionalidade, você pode ter seus dados no Timestream LiveAnalytics sem precisar depender de outras ferramentas ou escrever código personalizado. Você pode usar o carregamento em lote para preencher dados com tempos de espera flexíveis, como dados que não são imediatamente necessários para consulta ou análise.

Você pode criar tarefas de carregamento em lote usando o AWS Management Console AWS CLI, o e AWS SDKs o. Para mais informações, consulte Usando o carregamento em lote com o console, Usando o carregamento em lote com o AWS CLI e Usando o carregamento em lote com o AWS SDKs.

Para obter mais informações sobre carregamento em lote, consulteUsando o carregamento em lote no Timestream para LiveAnalytics.

Escolhendo entre a WriteRecords API operação e a carga em lote

Com a WriteRecords API operação, você pode gravar seus dados de séries temporais de streaming no Timestream LiveAnalytics conforme eles são gerados pelo seu sistema. Ao usar WriteRecords, você pode ingerir continuamente um único ponto de dados ou lotes menores de dados em tempo real. O Timestream for LiveAnalytics oferece um esquema flexível que detecta automaticamente os nomes das colunas e os tipos de dados do seu Timestream para LiveAnalytics tabelas, com base nos nomes das dimensões e nos tipos de dados dos pontos de dados que você especifica ao invocar gravações no banco de dados.

Por outro lado, o carregamento em lote permite a ingestão robusta de dados de séries temporais em lote dos arquivos de origem (CSVarquivos) no Timestream for LiveAnalytics, usando um modelo de dados definido por você. Alguns exemplos de quando usar o carregamento em lote com um arquivo de origem são importar dados de séries temporais em massa para a avaliação do Timestream por LiveAnalytics meio de uma prova de conceito, importar dados de séries temporais em massa de um dispositivo de IoT que ficou off-line por algum tempo e migrar dados históricos de séries temporais do Amazon S3 para o Timestream for. LiveAnalytics Para obter informações sobre carregamento em lote, consulteUsando o carregamento em lote no Timestream para LiveAnalytics.

Ambas as soluções são seguras, confiáveis e de alto desempenho.

Use WriteRecords quando:

- Streaming de quantidades menores (menos de 10 MB) de dados por solicitação.
- Preenchendo tabelas existentes.
- Ingestão de dados de um stream de registros.
- Realizando análises em tempo real.
- Exigindo menor latência.

Use o carregamento em lote quando:

- Ingestão de cargas maiores de dados originados no Amazon CSV S3 em arquivos. Para obter mais informações sobre limites, consulte Cotas.
- Preencher novas tabelas, como no caso de uma migração de dados.
- Enriquecimento de bancos de dados com dados históricos (ingestão em novas tabelas).

- Você tem dados de origem que mudam lentamente ou não mudam.
- Você tem tempos de espera flexíveis porque uma tarefa de carregamento em lote pode estar em um estado pendente até que os recursos estejam disponíveis, especialmente se você carregar uma grande quantidade de dados. O carregamento em lote é adequado para dados que não precisam estar prontamente disponíveis para consulta ou análise para aumentar a clareza.

Armazenamento

O Timestream for Live Analytics armazena e organiza seus dados de séries temporais para otimizar o tempo de processamento de consultas e reduzir os custos de armazenamento. Ele oferece armazenamento de dados em camadas e suporta dois níveis de armazenamento: um armazenamento de memória e um armazenamento magnético. O armazenamento de memória é otimizado para gravações de dados de alto rendimento e point-in-time consultas rápidas. O armazenamento magnético é otimizado para gravação tardia de dados com menor taxa de transferência, armazenamento de dados de longo prazo e consultas analíticas rápidas.

O Timestream for Live Analytics garante a durabilidade de seus dados ao replicar automaticamente seus dados de memória e armazenamento magnético em diferentes zonas de disponibilidade em uma única zona. Região da AWS Todos os seus dados são gravados em disco antes de confirmar que sua solicitação de gravação foi concluída.

O Timestream for Live Analytics permite que você configure políticas de retenção para mover dados do armazenamento de memória para o armazenamento magnético. Quando os dados atingem o valor configurado, o Timestream for Live Analytics move automaticamente os dados para o armazenamento magnético. Você também pode definir um valor de retenção na loja magnética. Quando os dados expiram fora do armazenamento magnético, eles são excluídos permanentemente.

Por exemplo, considere um cenário em que você configura o armazenamento de memória para armazenar dados de uma semana e o armazenamento magnético para armazenar dados de 1 ano. A idade dos dados é calculada usando o timestamp associado ao ponto de dados. Quando os dados no armazenamento de memória completam uma semana, eles são automaticamente movidos para o armazenamento magnético. Depois, serão retido no armazenamento magnético por um ano. Quando os dados completam um ano, eles são excluídos do Timestream for Live Analytics. Os valores de retenção do armazenamento de memória e do armazenamento magnético definem cumulativamente a quantidade de tempo em que seus dados serão armazenados no Timestream for Live Analytics. Isso significa que, no cenário acima, a partir do momento da chegada dos dados, os dados são armazenados no Timestream for Live Analytics por um período total de 1 ano e 1 semana.

> **Note**
>
> Quando você atualiza o período de retenção da memória ou do armazenamento magnético, a alteração de retenção entra em vigor a partir desse ponto. Por exemplo, se o período de retenção do armazenamento de memória foi definido para 2 horas e depois alterado para 24 horas com a atualização das políticas de retenção da tabela, o armazenamento de memória será capaz de armazenar 24 horas de dados, mas será preenchido com 24 horas de dados 22 horas após a alteração ter sido feita. O Timestream for Live Analytics não recupera dados do armazenamento magnético para preencher o armazenamento de memória.

Para garantir a segurança de seus dados de série temporal, seus dados no Timestream for Live Analytics são sempre criptografados por padrão. Isso se aplica aos dados em trânsito e em repouso. Além disso, o Timestream for Live Analytics permite que você use chaves gerenciadas pelo cliente para proteger seus dados na loja magnética. Para obter mais informações sobre chaves gerenciadas pelo cliente, consulte AWS KMS keys.

Consultas

Com o Timestream for Live Analytics, você pode armazenar e analisar facilmente métricas DevOps, dados de sensores para aplicativos de IoT e dados de telemetria industrial para manutenção de equipamentos, além de muitos outros casos de uso. O mecanismo de consulta adaptável e criado especificamente no Timestream for Live Analytics permite que você acesse dados em vários níveis de armazenamento usando uma única instrução. SQL Ele acessa e combina dados de forma transparente em todos os níveis de armazenamento sem exigir que você especifique a localização dos dados. Você pode usar SQL para consultar dados no Timestream for Live Analytics para recuperar dados de séries temporais de uma ou mais tabelas. Você pode acessar as informações de metadados para bancos de dados e tabelas. O Timestream for Live Analytics SQL também oferece suporte a funções integradas para análise de séries temporais. Você pode consultar a Referência da linguagem de consulta referência para obter detalhes adicionais.

O Timestream for Live Analytics foi projetado para ter uma arquitetura de ingestão, armazenamento e consulta de dados totalmente dissociada, na qual cada componente pode ser escalado independentemente de outros componentes (permitindo que ele ofereça uma escala praticamente infinita para as necessidades de um aplicativo). Isso significa que o Timestream for Live Analytics não “tomba” quando seus aplicativos enviam centenas de terabytes de dados por dia ou executam milhões de consultas processando pequenas ou grandes quantidades de dados. À medida que seus dados crescem com o tempo, a latência da consulta no Timestream for Live Analytics permanece

praticamente inalterada. Isso ocorre porque a arquitetura de consulta Timestream for Live Analytics pode aproveitar grandes quantidades de paralelismo para processar volumes de dados maiores e escalar automaticamente para atender às necessidades de taxa de transferência de consultas de um aplicativo.

Modelo de dados

O Timestream oferece suporte a dois modelos de dados para consultas: o modelo plano e o modelo de série temporal.

> **Note**
>
> Os dados no Timestream são armazenados usando o modelo plano e é o modelo padrão para consultar dados. O modelo de séries temporais é um conceito de tempo de consulta e é usado para análise de séries temporais.

- Modelo plano
- Modelo de série temporal

Modelo plano

O modelo plano é o modelo de dados padrão do Timestream para consultas. Ele representa dados de séries temporais em formato tabular. Os nomes das dimensões, a hora, os nomes das medidas e os valores das medidas aparecem como colunas. Cada linha na tabela é um ponto de dados atômico correspondente a uma medição em um momento específico dentro de uma série temporal. Bancos de dados, tabelas e colunas do Timestream têm algumas restrições de nomenclatura. Esses são descritos emLimites do serviço.

A tabela abaixo mostra um exemplo ilustrativo de como o Timestream armazena dados que representam a CPU utilização, a utilização da memória e a atividade de rede das EC2 instâncias, quando os dados são enviados como um registro de medida única. Nesse caso, as dimensões são a região, a zona de disponibilidade, a nuvem privada virtual e a instância IDs das EC2 instâncias. As medidas são a CPU utilização, a utilização da memória e os dados de rede de entrada das instâncias. EC2 As colunas region, az, vpc e instance_id contêm os valores da dimensão. A hora da coluna contém a data e hora de cada registro. A coluna measure_name contém os nomes das medidas representadas por cpu-utilization, memory_utilization e network_bytes_in. As colunas

measure_value: :double contêm medições emitidas como duplas (por exemplo, utilização e CPU utilização de memória). A coluna measure_value: :bigint contém medições emitidas como números inteiros, por exemplo, os dados de entrada da rede.

Tempo	região	az	vpc	instance_id	nome_medida	valor_medida::duplo	valor_medida::bigint
2019-12-04 19:00:00.000000000	us-east-1	leste dos EUA - 1d	vpc-1a2b3c4d	i-1234567890abcdef0	utilização_da CPU	35,0	nulo
2019-12-04 19:00:01.000000000	us-east-1	leste dos EUA - 1d	vpc-1a2b3c4d	i-1234567890abcdef0	utilização_da CPU	38,2	nulo
2019-12-04 19:00:02.000000000	us-east-1	leste dos EUA - 1d	vpc-1a2b3c4d	i-1234567890abcdef0	utilização_da CPU	45,3	nulo
2019-12-04 19:00:00.000000000	us-east-1	leste dos EUA - 1d	vpc-1a2b3c4d	i-1234567890abcdef0	memory_utilization	54,9	nulo
2019-12-04 19:00:01.000000000	us-east-1	leste dos EUA - 1d	vpc-1a2b3c4d	i-1234567890abcdef0	memory_utilization	42.6	nulo

Tempo	região	az	vpc	instance_id	nome_medida	valor_medida::duplo	valor_medida::bigint
2019-12-04 19:00:02.000 000000	us-east-1	leste dos EUA - 1d	vpc-1a2b3c4d	i-1234567890abcdef0	memory_utilization	33.3	nulo
2019-12-04 19:00:00.000 000000	us-east-1	leste dos EUA - 1d	vpc-1a2b3c4d	i-1234567890abcdef0	bytes_de_rede	34.400	nulo
2019-12-04 19:00:01.000 000000	us-east-1	leste dos EUA - 1d	vpc-1a2b3c4d	i-1234567890abcdef0	bytes_de_rede	1.500	nulo
2019-12-04 19:00:02.000 000000	us-east-1	leste dos EUA - 1d	vpc-1a2b3c4d	i-1234567890abcdef0	bytes_de_rede	6.000	nulo

A tabela abaixo mostra um exemplo ilustrativo de como o Timestream armazena dados que representam a CPU utilização, a utilização da memória e a atividade de rede das EC2 instâncias, quando os dados são enviados como um registro de várias medidas.

Tempo	região	az	vpc	instance_id	nome_medida	utilização_da CPU	memory_utilization	bytes_de_rede
2019-12-04 19:00:00.000 000000	us-east-1	leste dos EUA - 1d	vpc-1a2b3c4d	i-1234567890abcde0	métricas	35,0	54,9	34.400
2019-12-04 19:00:01.000 000000	us-east-1	leste dos EUA - 1d	vpc-1a2b3c4d	i-1234567890abcde0	métricas	38,2	42.6	1.500
2019-12-04 19:00:02.000 000000	us-east-1	leste dos EUA - 1d	vpc-1a2b3c4d	i-1234567890abcde0	métricas	45,3	33.3	6.600

Modelo de série temporal

O modelo de séries temporais é uma construção de tempo de consulta usada para análise de séries temporais. Ele representa os dados como uma sequência ordenada de pares (tempo, valor da medida). O Timestream suporta funções de séries temporais, como interpolação, para permitir que você preencha as lacunas em seus dados. Para usar essas funções, você deve converter seus dados no modelo de série temporal usando funções como create_time_series. Consulte Referência da linguagem de consulta para obter mais detalhes.

Usando o exemplo anterior da EC2 instância, aqui estão os dados de CPU utilização expressos como uma série temporal.

região	az	vpc	instance_id	utilização_da CPU
us-east-1	leste dos EUA - 1d	vpc-1a2b3c4d	i-1234567 890abcdef0	[{horário: 04/12/2019 19:00:00.000 000000, valor: 35}, {horário: 04/12/2019 19:00:01.000 000000, valor: 38,2}, {horário: 04/12/2019 19:00:02.000 000000, valor: 45,3}]

Consultas programadas

O recurso de consulta agendada no Amazon Timestream for Live Analytics é uma solução totalmente gerenciada, sem servidor e escalável para calcular e armazenar agregados, rollups e outras formas de dados pré-processados, normalmente usadas para alimentar painéis operacionais, relatórios comerciais, análises ad hoc e outros aplicativos. As consultas agendadas tornam a análise em tempo real mais eficiente e econômica, para que você possa obter informações adicionais de seus dados e continuar tomando melhores decisões de negócios.

Para obter mais informações sobre a consulta agendada, consulteUsando consultas agendadas no Timestream para LiveAnalytics.

Unidade de computação Timestream () TCU

O Amazon Timestream for Live Analytics mede a capacidade computacional alocada para suas necessidades de consulta na unidade computacional Timestream (). TCU Um TCU é composto por 4 vCPUs e 16 GB de memória. Quando você executa consultas no Timestream for Live Analytics, o serviço aloca TCUs sob demanda com base na complexidade de suas consultas e na quantidade de dados que estão sendo processados. O número TCUs que uma consulta consome determina o custo associado.

> **Note**
>
> Tudo o Contas da AWS que estiver a bordo do serviço após 29 de abril de 2024 será usado como padrão TCUs para consulta de preços.

Neste tópico

- MaxQuery TCU
- Faturamento da TCU
- Configurando TCU
- Estimando as unidades computacionais necessárias
- Quando aumentar MaxQuery TCU
- Quando diminuir MaxQuery TCU
- Monitorando o uso com CloudWatch métricas
- Entendendo as variações no uso de unidades de computação

MaxQuery TCU

Essa configuração especifica o número máximo de unidades computacionais que o serviço usará em qualquer momento para atender às suas consultas. Para executar consultas, você deve definir a capacidade mínima como 4TCUs. Você pode definir o número máximo de TCUs em múltiplos de 4, por exemplo, 4, 8, 16, 32 e assim por diante. Você é cobrado somente pelos recursos computacionais que usa para sua carga de trabalho. Por exemplo, se você definir o máximo TCUs para 128, mas usar consistentemente somente 8TCUs. Você será cobrado apenas pela duração em que usou o 8TCUs. O padrão `MaxQueryTCU` na sua conta está definido como 200. Você pode ajustar `MaxQueryTCU` de 4 a 1000, usando a UpdateAccountSettingsAPIoperação AWS Management Console ou com o AWS SDK ou AWS CLI.

Recomendamos definir o `MaxQueryTCU` para sua conta. Definir um TCU limite máximo ajuda a controlar os custos ao restringir o número de unidades computacionais que o serviço pode usar para sua carga de trabalho de consulta. Isso permite que você preveja e gerencie melhor seus gastos com consultas.

Faturamento da TCU

Cada um TCU é cobrado por hora com granularidade por segundo e por um mínimo de 30 segundos. A unidade de uso dessas unidades computacionais é TCU -hora.

Ao executar consultas, você é cobrado pelo TCUs uso durante o tempo de execução da consulta, medido em TCU -horas. Por exemplo:

- Sua carga de trabalho usa 20 TCUs por 3 horas. Você é cobrado por 60 TCU horas (20 TCUs x 3 horas).
- Sua carga de trabalho usa 10 TCUs por 30 minutos e depois 20 TCUs pelos próximos 30 minutos. Você é cobrado por 15 TCU horas (10 TCUs x 0,5 horas + 20 TCUs x 0,5 horas).

O preço por TCU hora varia de acordo com. Região da AWS Consulte a definição de preço do Amazon Timestream para obter detalhes adicionais. Conforme sua carga de trabalho cresce, o serviço escala automaticamente a capacidade de computação até o TCU limite máximo especificado (`MaxQueryTCU`) para manter um desempenho consistente. A `MaxQueryTCU` configuração atua como um teto para a capacidade computacional para a qual o serviço pode ser escalado. Essa configuração ajuda você a controlar o número de recursos computacionais e, consequentemente, seus custos.

Configurando TCU

Quando você integra o serviço, cada um Conta da AWS tem um `MaxQueryTCU` limite padrão de 200. Você pode atualizar esse limite conforme necessário a qualquer momento usando a UpdateAccountSettingsAPIoperação AWS Management Console ou com o AWS SDK ou AWS CLI.

Se você não tiver certeza sobre os valores a serem configurados, monitore a `QueryTCU` métrica da sua conta. Essa métrica está disponível no AWS Management Console e na Amazon CloudWatch. Essa métrica fornece informações sobre o número máximo de granularidades TCUs usadas em um minuto. Com base em dados históricos e em sua estimativa de crescimento futuro, defina o `MaxQueryTCU` para acomodar os picos em seu uso. Recomendamos ter um espaço livre de pelo menos 4 a 16 TCUs acima do pico de uso. Por exemplo, se seu pico `QueryTCU` nos últimos 30 dias foi 128, recomendamos definir `MaxQueryTCU` entre 132 e 144.

Estimando as unidades computacionais necessárias

As unidades de computação podem processar consultas simultaneamente. Para determinar o número de unidades computacionais necessárias, considere as diretrizes gerais na tabela a seguir:

Consultas simultâneas	TCUs
7	4
14	8
21	12

Note

- Essas são diretrizes gerais e o número real de unidades computacionais necessárias depende de vários fatores, como:
  - A concorrência efetiva das consultas.
  - Padrões de consulta.
  - O número de partições digitalizadas.
  - Outras características específicas da carga de trabalho.
- Essa diretriz se refere às consultas que examinam os dados dos últimos minutos a uma hora e seguem as melhores práticas de consulta do Timestream e as diretrizes de modelagem de dados.
- Monitore o desempenho e a `QueryTCU` métrica do seu aplicativo para ajustar as unidades de computação, conforme necessário.

Quando aumentar MaxQuery TCU

Você deve considerar aumentar o `MaxQueryTCU` nos seguintes cenários:

- Seu pico de consumo de consultas está se aproximando ou atingindo a consulta TCU máxima configurada atualmente. Recomendamos definir a consulta máxima pelo TCU menos 4 a 16 a TCUs mais do que seu consumo máximo.
- Suas consultas estão retornando um erro 4xx com a mensagem MaxQuery TCU excedida. Se você prevê um aumento planejado em sua carga de trabalho, revise e ajuste adequadamente a consulta máxima configurada. TCU

Quando diminuir MaxQuery TCU

Você deve considerar a redução do `MaxQueryTCU` nos seguintes cenários:

- Sua carga de trabalho tem um padrão de uso previsível e estável, e você tem uma boa compreensão dos requisitos de uso da computação. Reduzir a consulta máxima para 4 TCU a 16 TCU acima do pico de consumo pode ajudar a evitar o uso e os custos não intencionais. Você pode modificar o valor usando a UpdateAccountSettingsAPIoperação.
- O pico de uso da sua carga de trabalho diminuiu com o tempo, seja devido a mudanças no aplicativo ou nos padrões de comportamento do usuário. Reduzir o máximo TCU pode ajudar a mitigar custos não intencionais.

> **Note**
>
> Dependendo do seu uso atual, a redução da alteração do TCU limite máximo pode levar até 24 horas para ser efetiva. Você é cobrado somente pelo TCUs que suas consultas realmente consomem. Ter um TCU limite máximo de consulta maior não afeta seus custos, a menos que eles TCUs sejam usados por sua carga de trabalho.

Monitorando o uso com CloudWatch métricas

Para monitorar seu TCU uso, Timestream for Live Analytics fornece a seguinte CloudWatch métrica:. `QueryTCU` Essa métrica especifica o número de unidades de computação usadas em um minuto e é emitida a cada minuto. Você pode optar por monitorar o máximo e o mínimo TCUs usados em um minuto. Você também pode definir alarmes nessa métrica para monitorar seus custos de consulta em tempo real.

Entendendo as variações no uso de unidades de computação

O número de recursos computacionais necessários para suas consultas pode aumentar ou diminuir com base em vários parâmetros. Por exemplo, volume de dados, padrões de ingestão de dados, latência da consulta, formato da consulta, eficiência da consulta e combinações de consultas que usam consultas analíticas e em tempo real. Esses parâmetros podem levar a TCU unidades maiores ou menores necessárias para sua carga de trabalho. Em um estado estável em que esses parâmetros não mudam, você pode observar que o número de unidades computacionais necessárias para sua carga de trabalho diminui. Consequentemente, isso pode reduzir seu custo mensal.

Além disso, se algum desses parâmetros em sua carga de trabalho ou dados mudar, o número de unidades computacionais necessárias poderá aumentar. Quando o Timestream recebe uma consulta, dependendo das partições de dados que a consulta acessa, o Timestream decide o número de recursos computacionais para tratar a consulta com desempenho.

Em intervalos periódicos, com base em seus padrões de acesso à ingestão e consulta, o Timestream otimiza o layout dos dados. O Timestream realiza a otimização agrupando partições menos acessadas em uma única partição ou dividindo uma partição ativa em várias partições para fins de desempenho. Consequentemente, a capacidade computacional usada pela mesma consulta pode variar um pouco em diferentes momentos.

> **Optando por usar os TCU preços para suas consultas**
>
> Como usuário existente, você pode optar TCUs por usar uma única vez para melhor gerenciamento de custos e remoção dos bytes mínimos medidos por consulta. Você pode optar por usar a UpdateAccountSettingsAPIoperação AWS Management Console ou com o AWS SDK ou AWS CLI. Na API operação, defina o `QueryPricingModel` parâmetro como`COMPUTE_UNITS`.
>
> Optar pelo modelo de preços baseado em computação é uma mudança irreversível.

Acessando o Timestream para LiveAnalytics

Você pode acessar o Timestream para LiveAnalytics usar o console CLI ou o. API Para obter informações sobre como acessar o Timestream para LiveAnalytics, consulte o seguinte:

Tópicos

- Inscreva-se para um Conta da AWS
- Criar um usuário com acesso administrativo
- Forneça um Timestream para acesso LiveAnalytics
- Conceder acesso programático

Inscreva-se para um Conta da AWS

Se você não tiver um Conta da AWS, conclua as etapas a seguir para criar um.

Para se inscrever em um Conta da AWS

1. Abra a https://portal.aws.amazon.com/billing/inscrição.
2. Siga as instruções online.

   Parte do procedimento de inscrição envolve receber uma chamada telefônica e inserir um código de verificação no teclado do telefone.

   Quando você se inscreve em um Conta da AWS, um Usuário raiz da conta da AWSé criado. O usuário raiz tem acesso a todos os Serviços da AWS e atributos na conta. Como prática recomendada de segurança, atribua o acesso administrativo a um usuário e use somente o usuário-raiz para executar tarefas que exigem acesso de usuário-raiz.

AWS envia um e-mail de confirmação após a conclusão do processo de inscrição. A qualquer momento, você pode visualizar a atividade atual da sua conta e gerenciar sua conta acessando https://aws.amazon.com/e escolhendo Minha conta.

Criar um usuário com acesso administrativo

Depois de se inscrever em um Conta da AWS, proteja seu Usuário raiz da conta da AWS AWS IAM Identity Center, habilite e crie um usuário administrativo para que você não use o usuário root nas tarefas diárias.

Proteja seu Usuário raiz da conta da AWS

1. Faça login AWS Management Consolecomo proprietário da conta escolhendo Usuário raiz e inserindo seu endereço de Conta da AWS e-mail. Na próxima página, insira sua senha.

   Para obter ajuda ao fazer login usando o usuário raiz, consulte Fazer login como usuário raiz no Guia do usuário do Início de Sessão da AWS .
2. Ative a autenticação multifator (MFA) para seu usuário root.

   Para obter instruções, consulte Habilitar um MFA dispositivo virtual para seu usuário Conta da AWS root (console) no Guia IAM do usuário.

Criar um usuário com acesso administrativo

1. Ative o IAM Identity Center.

Para obter instruções, consulte Habilitar AWS IAM Identity Center no Guia do usuário do AWS IAM Identity Center .

2. No IAM Identity Center, conceda acesso administrativo a um usuário.

   Para ver um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como fonte de identidade, consulte Configurar o acesso do usuário com o padrão Diretório do Centro de Identidade do IAM no Guia AWS IAM Identity Center do usuário.

Iniciar sessão como o usuário com acesso administrativo

- Para entrar com seu usuário do IAM Identity Center, use o login URL que foi enviado ao seu endereço de e-mail quando você criou o usuário do IAM Identity Center.

  Para obter ajuda para fazer login usando um usuário do IAM Identity Center, consulte Como fazer login no portal de AWS acesso no Guia Início de Sessão da AWS do usuário.

Atribuir acesso a usuários adicionais

1. No IAM Identity Center, crie um conjunto de permissões que siga as melhores práticas de aplicação de permissões com privilégios mínimos.

   Para obter instruções, consulte Create a permission set no Guia do usuário do AWS IAM Identity Center .

2. Atribua usuários a um grupo e, em seguida, atribua o acesso de autenticação única ao grupo.

   Para obter instruções, consulte Add groups no Guia do usuário do AWS IAM Identity Center .

Forneça um Timestream para acesso LiveAnalytics

As permissões necessárias para acessar o Timestream já foram concedidas ao administrador. LiveAnalytics Para outros usuários, você deve conceder a eles o Timestream para LiveAnalytics acesso usando a seguinte política:

```
{
  "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**
>
> Para obter informações sobre`dbqms`, consulte Ações, recursos e chaves de condição do Database Query Metadata Service. Para obter informações sobre, `kms` consulte Ações, recursos e chaves de condição para o AWS Key Management Service.

Conceder acesso programático

Os usuários precisam de acesso programático se quiserem interagir com pessoas AWS fora do AWS Management Console. A forma de conceder acesso programático depende do tipo de usuário que está acessando AWS.

Para conceder acesso programático aos usuários, selecione uma das seguintes opções:

Qual usuário precisa de acesso programático?	Para	Por
Identificação da força de trabalho	Use credenciais temporári as para assinar solicitações	Siga as instruções da interface que deseja utilizar.

Qual usuário precisa de acesso programático?	Para	Por
(Usuários gerenciados no IAM Identity Center)	programáticas para o AWS CLI AWS SDKs, ou. AWS APIs	• Para o AWS CLI, consulte Configurando o AWS CLI para uso AWS IAM Identity Center no Guia do AWS Command Line Interface usuário. • Para AWS SDKs, ferrament as e AWS APIs, consulte Autenticação do IAM Identity Center no Guia de referência de ferramentas AWS SDKs e ferramentas.
IAM	Use credenciais temporári as para assinar solicitações programáticas para o AWS CLI AWS SDKs, ou. AWS APIs	Siga as instruções em Uso de credenciais temporárias com AWS recursos no Guia do IAM usuário.

Qual usuário precisa de acesso programático?	Para	Por
IAM	(Não recomendado) Use credenciais de longo prazo para assinar solicitaç ões programáticas para o AWS CLI, AWS SDKs, ou. AWS APIs	Siga as instruções da interface que deseja utilizar. • Para o AWS CLI, consulte Autenticação usando credenciais de IAM usuário no Guia do AWS Command Line Interface usuário. • Para ferramentas AWS SDKs e ferramentas, consulte Autenticar usando credenciais de longo prazo no Guia de referência de ferramentas AWS SDKs e ferramentas. • Para AWS APIs, consulte Gerenciamento de chaves de acesso para IAM usuários no Guia IAM do usuário.

Usar o console

Você pode usar o AWS Management Console do Timestream Live Analytics para criar, editar, excluir, descrever e listar bancos de dados e tabelas. Você também pode usar o console para executar consultas.

Tópicos

- Tutorial
- Criar um banco de dados do
- Criar uma tabela
- Execute uma consulta

- Criar uma consulta agendada
- Excluir uma consulta agendada
- Excluir uma tabela
- Excluir um banco de dados
- Editar uma tabela
- Editar um banco de dados

Tutorial

Este tutorial mostra como criar um banco de dados preenchido com conjuntos de dados de amostra e executar consultas de amostra. Os conjuntos de dados de amostra usados neste tutorial são vistos com frequência em DevOps IoT e cenários. O conjunto de dados de IoT contém dados de séries temporais, como velocidade, localização e carga de um caminhão, para agilizar o gerenciamento da frota e identificar oportunidades de otimização. O DevOps conjunto de dados contém métricas de EC2 instânciaCPU, como utilização da rede e da memória, para melhorar o desempenho e a disponibilidade do aplicativo. Aqui está um tutorial em vídeo com as instruções descritas nesta seção

Siga estas etapas para criar um banco de dados preenchido com os conjuntos de dados de amostra e executar consultas de amostra usando o AWS console.

1. Abra o AWS console.
2. No painel de navegação, escolha Bancos de dados
3. Clique em Criar banco de dados.
4. Na página criar banco de dados, digite o seguinte:
   - Escolha a configuração — Selecione Banco de dados de amostra.
   - Nome — Insira um nome de banco de dados de sua escolha.
   - Escolha conjuntos de dados de amostra — Selecione IoT e. DevOps
   - Clique em Criar banco de dados para criar um banco de dados contendo duas tabelas — IoT e DevOps preenchidas com dados de amostra.
5. No painel de navegação, escolha Editor de consultas
6. Selecione Exemplos de consultas no menu superior.
7. Clique em uma das consultas de amostra. Isso o levará de volta ao editor de consultas com o editor preenchido com a consulta de amostra.
8. Clique em Executar para executar a consulta e ver os resultados da consulta.

Criar um banco de dados do

Siga estas etapas para criar um banco de dados usando o AWS console.

1. Abra o AWS console.
2. No painel de navegação, escolha Bancos de dados
3. Clique em Criar banco de dados.
4. Na página criar banco de dados, digite o seguinte.
   - Escolha a configuração — Selecione Banco de dados padrão.
   - Nome — Insira um nome de banco de dados de sua escolha.
   - Criptografia — Escolha uma KMS chave ou use a opção padrão, na qual o Timestream Live Analytics criará uma KMS chave em sua conta, caso ainda não exista.
5. Clique em Criar banco de dados para criar um banco de dados.

Criar uma tabela

Siga estas etapas para criar uma tabela usando o AWS console.

1. Abra o AWS console.
2. No painel de navegação, escolha Tabelas
3. Clique em Criar tabela.
4. Na página de criação da tabela, insira o seguinte.
   - Nome do banco de dados — Selecione o nome do banco de dados criado emCriar um banco de dados do .
   - Nome da tabela — Insira um nome de tabela de sua escolha.
   - Retenção do armazenamento de memória — especifique por quanto tempo você deseja reter os dados no armazenamento de memória. O armazenamento de memória processa os dados recebidos, incluindo dados de chegada tardia (dados com um registro de data e hora anterior à hora atual) e é otimizado para consultas rápidas. point-in-time
   - Retenção do armazenamento magnético — especifique por quanto tempo você deseja reter os dados no armazenamento magnético. O armazenamento magnético é destinado ao armazenamento de longo prazo e é otimizado para consultas analíticas rápidas.
5. Clique em Criar tabela.

Execute uma consulta

Siga estas etapas para executar consultas usando o AWS console.

1. Abra o AWS console.
2. No painel de navegação, escolha Editor de consultas
3. No painel esquerdo, selecione o banco de dados criado emCriar um banco de dados do .
4. No painel esquerdo, selecione o banco de dados criado emCriar uma tabela.
5. No editor de consultas, você pode executar uma consulta. Para ver as 10 linhas mais recentes na tabela, execute:

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

6. (Opcional) Ative a opção Ativar insights para obter informações sobre a eficiência de suas consultas.

Criar uma consulta agendada

Siga estas etapas para criar uma consulta agendada usando o AWS console.

1. Abra o AWS console.
2. No painel de navegação, escolha Consultas agendadas.
3. Clique em Criar consulta agendada.
4. Nas seções Nome da consulta e Tabela de destino, insira o seguinte.
   - Nome — Insira um nome de consulta.
   - Nome do banco de dados — Selecione o nome do banco de dados criado emCriar um banco de dados do .
   - Nome da tabela — Selecione o nome da tabela criada emCriar uma tabela.
5. Na seção Instrução de consulta, insira uma instrução de consulta válida. Em seguida, clique em Validar consulta.
6. Em Modelo de tabela de destino, defina o modelo para quaisquer atributos indefinidos. Você pode usar o Visual Builder ouJSON.
7. Na seção Programação de execução, escolha Taxa fixa ou Expressão crônica. Para expressões crônicas, consulte Expressões de Programação para Consultas Programadas para obter mais detalhes sobre expressões de programação.

8. Na seção de SNStópico, insira o SNS tópico que será usado para notificação.
9. Na seção Relatório de registro de erros, insira o local do S3 que será usado para relatar erros.

   Selecione o Encryption key type (Tipo de chave de criptografia).
10. Na seção Configurações de segurança, em AWS KMSChave, escolha o tipo de AWS KMS chave.

    Insira a IAMfunção que o Timestream LiveAnalytics usará para executar a consulta agendada. Consulte os exemplos IAM de políticas para consultas agendadas para obter detalhes sobre as permissões necessárias e a relação de confiança para a função.
11. Clique em Criar consulta agendada.

Excluir uma consulta agendada

Siga estas etapas para excluir ou desativar uma consulta agendada usando o AWS console.

1. Abra o AWS console.
2. No painel de navegação, escolha Consultas agendadas
3. Selecione a consulta agendada criada emCriar uma consulta agendada.
4. Selecione Ações.
5. Escolha Desativar ou Excluir.
6. Se você selecionou Excluir, confirme a ação e selecione Excluir.

Excluir uma tabela

Siga estas etapas para excluir um banco de dados usando o AWS console.

1. Abra o AWS console.
2. No painel de navegação, escolha Tabelas
3. Selecione a tabela que você criou emCriar uma tabela.
4. Clique em Excluir.
5. Digite delete na caixa de confirmação.

Excluir um banco de dados

Siga estas etapas para excluir um banco de dados usando o AWS console:

1. Abra o AWS console.
2. No painel de navegação, escolha Bancos de dados
3. Selecione o banco de dados que você criou em Criar um banco de dados.
4. Clique em Excluir.
5. Digite delete na caixa de confirmação.

Editar uma tabela

Siga estas etapas para editar uma tabela usando o AWS console.

1. Abra o AWS console.
2. No painel de navegação, escolha Tabelas
3. Selecione a tabela que você criou emCriar uma tabela.
4. Clique em Editar
5. Edite os detalhes da tabela e salve.

    - Retenção do armazenamento de memória — especifique por quanto tempo você deseja reter os dados no armazenamento de memória. O armazenamento de memória processa os dados recebidos, incluindo dados de chegada tardia (dados com um registro de data e hora anterior à hora atual) e é otimizado para consultas rápidas. point-in-time
    - Retenção do armazenamento magnético — especifique por quanto tempo você deseja reter os dados no armazenamento magnético. O armazenamento magnético é destinado ao armazenamento de longo prazo e é otimizado para consultas analíticas rápidas.

Editar um banco de dados

Siga estas etapas para editar um banco de dados usando o AWS console.

1. Abra o AWS console.
2. No painel de navegação, escolha Bancos de dados
3. Selecione o banco de dados que você criou em Criar um banco de dados.
4. Clique em Editar

5. Edite os detalhes do banco de dados e salve.

Acessando o Amazon LiveAnalytics Timestream para usar o AWS CLI

Você pode usar o AWS Command Line Interface (AWS CLI) para controlar vários AWS serviços na linha de comando e automatizá-los por meio de scripts. Você pode usar o AWS CLI para operações ad hoc. Você também pode usá-lo para incorporar o Amazon LiveAnalytics Timestream para operações em scripts de utilitários.

Antes de usar o AWS CLI with Timestream for LiveAnalytics, você deve configurar o acesso programático. Para obter mais informações, consulte Conceder acesso programático.

Para obter uma lista completa de todos os comandos disponíveis para o Timestream for LiveAnalytics Query API no AWS CLI, consulte a Referência de AWS CLI Comandos.

Para obter uma lista completa de todos os comandos disponíveis para o Timestream for LiveAnalytics Write API in the AWS CLI, consulte a Referência de AWS CLI Comandos.

Tópicos

- Download e configuração da AWS CLI
- Usando o AWS CLI com Timestream para LiveAnalytics

Download e configuração da AWS CLI

AWS CLI É executado em Windows, macOS ou Linux. Para baixá-lo, instalá-lo e configurá-lo, siga estas etapas:

1. Faça o download AWS CLI em http://aws.amazon.com/cli.
2. Siga as instruções para instalar AWS CLI e configurar o AWS CLI no Guia do AWS Command Line Interface usuário.

Usando o AWS CLI com Timestream para LiveAnalytics

O formato da linha de comando consiste em um Amazon Timestream LiveAnalytics para o nome da operação, seguido pelos parâmetros dessa operação. O AWS CLI suporta uma sintaxe abreviada para os valores dos parâmetros, além de. JSON

Use `help` para listar todos os comandos disponíveis no Timestream for. LiveAnalytics Por exemplo:

```
aws timestream-write help
```

```
aws timestream-query help
```

Você também pode usar `help` para descrever um comando específico e saber mais sobre seu uso:

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

Por exemplo, para criar um banco de dados:

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

Para criar uma tabela com gravações de armazenamento magnético ativadas:

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

Para gravar dados usando registros de medida única:

```
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\"}]"
```

Para gravar dados usando registros de várias medidas:

```
# 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
```

Para consultar uma tabela:

```
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)"
```

Para criar uma consulta agendada:

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

Usando o API

Além do SDKs, o Amazon Timestream LiveAnalytics for fornece acesso REST API direto por meio do padrão de descoberta de endpoints. O padrão de descoberta de endpoints é descrito abaixo, junto com seus casos de uso.

O padrão de descoberta de endpoints

Como o Timestream Live Analytics foi projetado para trabalhar de forma transparente com a arquitetura do serviço, incluindo o gerenciamento e o mapeamento dos endpoints do serviço, é

recomendável que você o use para a maioria dos SDKs aplicativos. SDKs No entanto, há alguns casos em que o uso do padrão Timestream para descoberta de LiveAnalytics REST API endpoints é necessário:

- Você está usando VPCendpoints (AWS PrivateLink) com Timestream para LiveAnalytics
- Seu aplicativo usa uma linguagem de programação que ainda não tem SDK suporte
- Você precisa de um melhor controle sobre a implementação do lado do cliente

Esta seção inclui informações sobre como o padrão de descoberta de endpoint funciona, como implementar o padrão de descoberta de endpoint e notas de uso. Selecione um tópico abaixo para saber mais.

Tópicos

- Como funciona o padrão de descoberta de endpoints
- Implementando o padrão de descoberta de endpoints

Como funciona o padrão de descoberta de endpoints

O Timestream é construído usando uma arquitetura celular para garantir melhores propriedades de escalonamento e isolamento de tráfego. Como cada conta de cliente é mapeada para uma célula específica em uma região, seu aplicativo deve usar os endpoints corretos específicos da célula para os quais sua conta foi mapeada. Ao usar oSDKs, esse mapeamento é tratado de forma transparente para você e você não precisa gerenciar os endpoints específicos da célula. No entanto, ao acessar diretamente o RESTAPI, você mesmo precisará gerenciar e mapear os endpoints corretos. Esse processo, o padrão de descoberta de endpoints, é descrito abaixo:

1. O padrão de descoberta do endpoint começa com uma chamada para a `DescribeEndpoints` ação (descrita na `DescribeEndpoints`seção).
2. O endpoint deve ser armazenado em cache e reutilizado pelo período de tempo especificado pelo valor retornado time-to-live (TTL) (the). `CachePeriodInMinutes` As chamadas para o Timestream Live Analytics API podem então ser feitas durante o. TTL
3. Após a TTL expiração, uma nova chamada DescribeEndpoints deve ser feita para atualizar o endpoint (em outras palavras, recomeçar na Etapa 1).

> **Note**
>
> A sintaxe, os parâmetros e outras informações de uso da `DescribeEndpoints` ação estão descritos na APIReferência. Observe que a `DescribeEndpoints` ação está disponível por meio de ambos SDKs e é idêntica para cada um.

Para a implementação do padrão de descoberta de endpoints, consulteImplementando o padrão de descoberta de endpoints.

Implementando o padrão de descoberta de endpoints

Para implementar o padrão de descoberta de endpoint, escolha uma API (gravação ou consulta), crie uma DescribeEndpointssolicitação e use os endpoints retornados durante a duração dos TTL valores retornados. O procedimento de implementação está descrito abaixo.

> **Note**
>
> Certifique-se de estar familiarizado com as notas de uso.

Procedimento de implementação

1. Adquira o endpoint para o API qual você gostaria de fazer chamadas (Gravação ou Consulta). usando a `DescribeEndpoints`solicitação.

   a. Crie uma solicitação `DescribeEndpoints`que API corresponda à de interesse (Gravação ou Consulta) usando um dos dois endpoints descritos abaixo. Não há parâmetros de entrada para a solicitação. Certifique-se de ler as notas abaixo.

      EscrevaSDK:

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

      ConsultaSDK:

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

`us-east-1`Segue um exemplo de CLI chamada para região.

```
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**
>
> O cabeçalho HTTP “Host” também deve conter o API endpoint. A solicitação falhará se o cabeçalho não for preenchido. Esse é um requisito padrão para todas as solicitações HTTP /1.1. Se você usar uma HTTP biblioteca compatível com 1.1 ou posterior, a HTTP biblioteca deverá preencher automaticamente o cabeçalho para você.

> **Note**
>
> Substituto *<region>* com o identificador de região para a região em que a solicitação está sendo feita, por exemplo `us-east-1`

b. Analise a resposta para extrair o (s) ponto (s) final (s) e o (s) TTL valor (es) do cache. A resposta é uma matriz de um ou mais `Endpoint`objetos. Cada `Endpoint` objeto contém um endereço de endpoint (`Address`) e o TTL para esse endpoint (`CachePeriodInMinutes`).

2. Armazene em cache o endpoint até o especificadoTTL.
3. Quando o TTL expirar, recupere um novo endpoint recomeçando na etapa 1 da implementação.

Notas de uso do padrão de descoberta de endpoints

- A DescribeEndpointsação é a única ação que os endpoints regionais do Timestream Live Analytics reconhecem.
- A resposta contém uma lista de endpoints para os quais fazer chamadas do Timestream Live AnalyticsAPI.

- Em caso de resposta bem-sucedida, deve haver pelo menos um endpoint na lista. Se houver mais de um endpoint na lista, qualquer um deles poderá ser usado igualmente para as API chamadas, e o chamador poderá escolher o endpoint a ser usado aleatoriamente.
- Além do DNS endereço do endpoint, cada endpoint na lista especificará um time to live (TTL) que é permitido para usar o endpoint especificado em minutos.
- O endpoint deve ser armazenado em cache e reutilizado pelo tempo especificado pelo TTL valor retornado (em minutos). Após a TTL expiração, uma nova chamada DescribeEndpointsdeve ser feita para atualizar o endpoint a ser usado, pois o endpoint não funcionará mais após a expiração. TTL

Usando o AWS SDKs

Você pode acessar o Amazon Timestream usando o. AWS SDKs O Timestream suporta dois SDKs por idioma; ou seja, o Write SDK e o Query. SDK O Write SDK é usado para realizar CRUD operações e inserir seus dados de série temporal no Timestream. A consulta SDK é usada para consultar seus dados de séries temporais existentes armazenados no Timestream.

Depois de concluir os pré-requisitos necessários para sua SDK escolha, você pode começar com o. Exemplos de código

Tópicos

- Java
- Java versão 2
- Go
- Python
- Node.js
- .NET

Java

Para começar a usar o Java 1.0 SDK e o Amazon Timestream, preencha os pré-requisitos descritos abaixo.

Depois de concluir os pré-requisitos necessários para o JavaSDK, você pode começar com o. Exemplos de código

Pré-requisitos

Antes de começar com o Java, você deve fazer o seguinte:

1. Siga as instruções AWS de configuração emAcessando o Timestream para LiveAnalytics.
2. Configure um ambiente de desenvolvimento Java baixando e instalando o seguinte:
   - Java SE Development Kit 8 (como o Amazon Corretto 8).
   - Java IDE (como Eclipse ou IntelliJ).

   Para obter mais informações, consulte Introdução ao AWS SDK for Java
3. Configure suas AWS credenciais e sua região para desenvolvimento:
   - Configure suas credenciais de AWS segurança para uso com o. AWS SDK for Java
   - Defina sua AWS região para determinar seu Timestream padrão para LiveAnalytics endpoint.

Uso do Apache Maven

Você pode usar o Apache Maven para configurar e criar AWS SDK for Java projetos.

> **Note**
>
> Para usar o Apache Maven, certifique-se de que seu Java SDK e tempo de execução sejam 1.8 ou superior.

Você pode configurar o AWS SDK como uma dependência do Maven conforme descrito em Usando o SDK com o Apache Maven.

Você pode executar, compilar e executar seu código-fonte com o seguinte comando:

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

> **Note**
>
> `<your source code Main class>`é o caminho para a classe principal do seu código-fonte Java.

Configurando suas AWS credenciais

Isso AWS SDK for Javaexige que você forneça AWS credenciais para seu aplicativo em tempo de execução. Os exemplos de código neste guia pressupõem que você esteja usando um arquivo de AWS credenciais, conforme descrito em Configurar AWS credenciais e região para desenvolvimento no Guia do AWS SDK for Java desenvolvedor.

Veja a seguir um exemplo de um arquivo de AWS credenciais chamado`~/.aws/credentials`, em que o caractere tilde (~) representa seu diretório inicial.

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

Java versão 2

Para começar a usar o Java 2.0 SDK e o Amazon Timestream, preencha os pré-requisitos descritos abaixo.

Depois de concluir os pré-requisitos necessários para o Java 2.0SDK, você pode começar com o. Exemplos de código

Pré-requisitos

Antes de começar com o Java, você deve fazer o seguinte:

1. Siga as instruções AWS de configuração emAcessando o Timestream para LiveAnalytics.
2. Você pode configurar o AWS SDK como uma dependência do Maven conforme descrito em Usando o SDK com o Apache Maven.
3. Configure um ambiente de desenvolvimento Java baixando e instalando o seguinte:
   - Java SE Development Kit 8 (como o Amazon Corretto 8).
   - Java IDE (como Eclipse ou IntelliJ).

     Para obter mais informações, consulte Introdução ao AWS SDK for Java

Uso do Apache Maven

Você pode usar o Apache Maven para configurar e criar AWS SDK for Java projetos.

> **Note**
>
> Para usar o Apache Maven, certifique-se de que seu Java SDK e tempo de execução sejam 1.8 ou superior.

Você pode configurar o AWS SDK como uma dependência do Maven conforme descrito em Usando o SDK com o Apache Maven. As alterações necessárias no arquivo pom.xml estão descritas aqui.

Você pode executar, compilar e executar seu código-fonte com o seguinte comando:

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

> **Note**
>
> `<your source code Main class>` é o caminho para a classe principal do seu código-fonte Java.

Go

Para começar a usar o Go SDK e o Amazon Timestream, preencha os pré-requisitos descritos abaixo.

Depois de concluir os pré-requisitos necessários para o GoSDK, você pode começar com o. Exemplos de código

Pré-requisitos

1. Faça o download do GO SDK 1.14.
2. Configure o GO SDK.
3. Construa seu cliente.

Python

Para começar a usar o Python SDK e o Amazon Timestream, preencha os pré-requisitos descritos abaixo.

Depois de concluir os pré-requisitos necessários para o PythonSDK, você pode começar com o. Exemplos de código

Pré-requisitos

Para usar o Python, instale e configure o Boto3 seguindo as instruções aqui.

Node.js

Para começar a usar o Node.js SDK e o Amazon Timestream, preencha os pré-requisitos descritos abaixo.

Depois de concluir os pré-requisitos necessários para o Node.jsSDK, você pode começar com o. Exemplos de código

Pré-requisitos

Antes de começar a usar o Node.js, você deve fazer o seguinte:

1. Instale o Node.js.
2. Instale o AWS SDK para JavaScript.

.NET

Para começar com o. NETSDKe Amazon Timestream, preencha os pré-requisitos descritos abaixo.

Depois de concluir os pré-requisitos necessários para o. NETSDK, você pode começar com Exemplos de código o.

Pré-requisitos

Antes de começar com. NET, instale os NuGet pacotes necessários e garanta que a AWSSDK versão.Core seja 3.3.107 ou mais recente executando os seguintes comandos:

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

Conceitos básicos

Esta seção inclui um tutorial para você começar a usar o Amazon Timestream Live Analytics, bem como instruções para configurar um aplicativo de amostra totalmente funcional. Você pode começar com o tutorial ou com o aplicativo de amostra selecionando um dos links abaixo.

Tópicos

- Tutorial
- Aplicação de exemplo

Tutorial

Este tutorial mostra como criar um banco de dados preenchido com conjuntos de dados de amostra e executar consultas de amostra. Os conjuntos de dados de amostra usados neste tutorial são vistos com frequência em IoT e DevOps cenários. O conjunto de dados de IoT contém dados de séries temporais, como velocidade, localização e carga de um caminhão, para agilizar o gerenciamento da frota e identificar oportunidades de otimização. O conjunto DevOps de dados contém métricas de EC2 instânciaCPU, como utilização da rede e da memória, para melhorar o desempenho e a disponibilidade do aplicativo. Aqui está um tutorial em vídeo com as instruções descritas nesta seção.

Siga estas etapas para criar um banco de dados preenchido com os conjuntos de dados de amostra e executar consultas de amostra usando o AWS console:

Usar o console do

Siga estas etapas para criar um banco de dados preenchido com os conjuntos de dados de amostra e executar consultas de amostra usando o AWS console:

1. Abra o AWS console.
2. No painel de navegação, escolha Bancos de dados
3. Clique em Criar banco de dados.
4. Na página criar banco de dados, digite o seguinte:
   - Escolha a configuração — Selecione Banco de dados de amostra.
   - Nome — Insira um nome de banco de dados de sua escolha.

> **Note**
>
> Depois de criar um banco de dados com conjuntos de dados de amostra, para usar as consultas de amostra que estão disponíveis no console, você pode ajustar o nome do banco de dados referenciado na consulta para corresponder ao nome do banco de dados inserido aqui. Há exemplos de consultas para cada combinação de conjunto de dados de amostra e tipo de registros de séries temporais.

- Escolha conjuntos de dados de amostra — Selecione IoT e. DevOps
- Escolha o tipo de registros de séries temporais — Selecione Registros de várias medidas.
- Clique em Criar banco de dados para criar um banco de dados contendo duas tabelas preenchidas com dados de amostra. Os nomes das tabelas para conjuntos de dados de amostra com registros de várias medidas são `DevOpsMulti` e. `IoTMulti` Os nomes das tabelas para conjuntos de dados de amostra com registros de medida única são e. `DevOps` `IoT`

5. No painel de navegação, escolha Editor de consultas
6. Selecione Exemplos de consultas no menu superior.
7. Clique em uma das consultas de amostra para um conjunto de dados que você escolheu ao criar o banco de dados de amostra. Isso o levará de volta ao editor de consultas com o editor preenchido com a consulta de amostra.
8. Ajuste o nome do banco de dados para a consulta de amostra.
9. Clique em Executar para executar a consulta e ver os resultados da consulta.

Usando o SDKs

O Timestream Live Analytics fornece um aplicativo de amostra totalmente funcional que mostra como criar um banco de dados e uma tabela, preencher a tabela com aproximadamente 126 mil linhas de dados de amostra e executar consultas de amostra. O aplicativo de amostra está disponível em GitHubJava, Python, Node.js, Go e. NET.

1. Clone os aplicativos de amostra do GitHub repositório Timestream Live Analytics seguindo as instruções de. GitHub
2. Configure o AWS SDK para se conectar ao Amazon Timestream Live Analytics seguindo as instruções descritas em. Usando o AWS SDKs

3. Compile e execute o aplicativo de amostra usando as instruções abaixo:

   - Instruções para o aplicativo de amostra Java.
   - Instruções para o aplicativo de amostra Java v2.
   - Instruções para o aplicativo de amostra Go.
   - Instruções para o aplicativo de amostra em Python.
   - Instruções para o aplicativo de amostra Node.js.
   - Instruções para o. NETaplicação de amostra.

Aplicação de exemplo

O Timestream vem com um aplicativo de amostra totalmente funcional que mostra como criar um banco de dados e uma tabela, preencher a tabela com aproximadamente 126 mil linhas de dados de amostra e executar consultas de amostra. Siga as etapas abaixo para começar a usar o aplicativo de amostra em qualquer um dos idiomas compatíveis:

Java

1. Clone o GitHub repositório Timestream para aplicativos de LiveAnalytics amostra seguindo as instruções de. GitHub
2. Configure o AWS SDK para se conectar ao Timestream LiveAnalytics seguindo as instruções descritas em Introdução ao. Java
3. Execute o aplicativo de amostra Java seguindo as instruções descritas aqui

Java v2

1. Clone o GitHub repositório Timestream para aplicativos de LiveAnalytics amostra seguindo as instruções de. GitHub
2. Configure o AWS SDK para se conectar ao Amazon Timestream LiveAnalytics seguindo as instruções descritas em Introdução ao. Java versão 2
3. Execute o aplicativo de amostra Java 2.0 seguindo as instruções descritas aqui

Go

1. Clone o GitHub repositório Timestream para aplicativos de LiveAnalytics amostra seguindo as instruções de. GitHub
2. Configure o AWS SDK para se conectar ao Amazon Timestream LiveAnalytics seguindo as instruções descritas em Introdução ao. Go
3. Execute o aplicativo de amostra Go seguindo as instruções descritas aqui

Python

1. Clone o GitHub repositório Timestream para aplicativos de LiveAnalytics amostra seguindo as instruções de. GitHub
2. Configure o AWS SDK para se conectar ao Amazon Timestream seguindo as LiveAnalytics instruções descritas em. Python
3. Execute o aplicativo de amostra em Python seguindo as instruções descritas aqui.

Node.js

1. Clone o GitHub repositório Timestream para aplicativos de LiveAnalytics amostra seguindo as instruções de. GitHub
2. Configure o AWS SDK para se conectar ao Amazon Timestream LiveAnalytics seguindo as instruções descritas em Introdução ao. Node.js
3. Execute o aplicativo de amostra Node.js seguindo as instruções descritas aqui

.NET

1. Clone o GitHub repositório Timestream para aplicativos de LiveAnalytics amostra seguindo as instruções de. GitHub
2. Configure o AWS SDK para se conectar ao Amazon Timestream LiveAnalytics seguindo as instruções descritas em Introdução ao. .NET
3. Execute o. NETaplicação de amostra seguindo as instruções descritas aqui

Exemplos de código

Você pode acessar o Amazon Timestream usando o. AWS SDKs O Timestream suporta dois SDKs por idioma; ou seja, o Write SDK e o Query. SDK O Write SDK é usado para realizar CRUD operações e inserir seus dados de série temporal no Timestream. A consulta SDK é usada para consultar seus dados de séries temporais existentes armazenados no Timestream. Selecione um tópico na lista abaixo para obter mais detalhes, incluindo exemplos de código para cada um dos compatíveisSDKs.

Tópicos

- Write SDK o cliente
- SDKCliente de consulta
- Criar banco de dados
- Descreva o banco
- Atualizar banco de dados
- Excluir banco de dados
- Listar bancos de dados
- Create table
- Descreva a tabela
- Atualizar tabela
- Excluir tabela
- Listar tabelas
- Gravar dados (inserções e upserts)
- Executar consulta
- Executar UNLOAD consulta
- Cancelar consulta
- Criar tarefa de carregamento em lote
- Descrever a tarefa de carregamento em lote
- Listar tarefas de carregamento em lote
- Retomar a tarefa de carregamento em lote
- Criar consulta agendada

- Listar consulta agendada
- Descrever a consulta agendada
- Executar consulta agendada
- Atualizar consulta agendada
- Excluir consulta agendada

Write SDK o cliente

Você pode usar os seguintes trechos de código para criar um cliente Timestream para o Write. SDK O Write SDK é usado para realizar CRUD operações e inserir seus dados de série temporal no Timestream.

> **Note**
>
> Esses trechos de código são baseados em exemplos completos de aplicativos em. GitHub Para obter mais informações sobre como começar a usar os aplicativos de amostra, consulteAplicação de exemplo.

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

O trecho a seguir é usado AWS SDK para JavaScript a v3. Para obter mais informações sobre como instalar o cliente e o uso, consulte Timestream Write Client - AWS SDK for JavaScript v3.

Uma importação de comando adicional é mostrada aqui. A `CreateDatabaseCommand` importação não é necessária para criar o cliente.

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

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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);
```

Recomendamos que você use a seguinte configuração.

- Defina a contagem SDK de novas tentativas como`10`.
- Usar `SDK DEFAULT_BACKOFF_STRATEGY`.
- `RequestTimeout`Defina para `20` segundos.
- Defina o máximo de conexões para `5000` ou mais.

SDKCliente de consulta

Você pode usar os seguintes trechos de código para criar um cliente Timestream para a consulta. SDK A consulta SDK é usada para consultar seus dados de séries temporais existentes armazenados no Timestream.

> **Note**
>
> Esses trechos de código são baseados em exemplos completos de aplicativos em. GitHub Para obter mais informações sobre como começar a usar os aplicativos de amostra, consulteAplicação de exemplo.

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

O trecho a seguir é usado AWS SDK para JavaScript a v3. Para obter mais informações sobre como instalar o cliente e o uso, consulte Timestream Query Client -,AWS SDK para JavaScript v3.

Uma importação de comando adicional é mostrada aqui. A `QueryCommand` importação não é necessária para criar o cliente.

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

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. GitHub

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

.NET

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

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

Criar banco de dados

Você pode usar os seguintes trechos de código para criar um banco de dados.

> **Note**
>
> Esses trechos de código são baseados em exemplos completos de aplicativos em. GitHub Para obter mais informações sobre como começar a usar os aplicativos de amostra, consulteAplicação de exemplo.

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

O trecho a seguir é usado AWS SDK para JavaScript a v3. Para obter mais informações sobre como instalar o cliente e o uso, consulte Timestream Write Client - AWS SDK for JavaScript v3.

Veja também Classe CreateDatabaseCommand CreateDatabasee.

```
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);
    }
}
```

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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());
            }

        }
```

Descreva o banco

Você pode usar os seguintes trechos de código para obter informações sobre os atributos do seu banco de dados recém-criado.

> **Note**
>
> Esses trechos de código são baseados em exemplos completos de aplicativos em. GitHub Para obter mais informações sobre como começar a usar os aplicativos de amostra, consulteAplicação de exemplo.

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

O trecho a seguir é usado AWS SDK para JavaScript a v3. Para obter mais informações sobre como instalar o cliente e o uso, consulte Timestream Write Client - AWS SDK for JavaScript v3.

Veja também Classe DescribeDatabaseCommand DescribeDatabasee.

```
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;
    }
}
```

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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());
            }

        }
```

Atualizar banco de dados

Você pode usar os seguintes trechos de código para atualizar seus bancos de dados.

> **Note**
>
> Esses trechos de código são baseados em exemplos completos de aplicativos em. GitHub Para obter mais informações sobre como começar a usar os aplicativos de amostra, consulteAplicação de exemplo.

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

O trecho a seguir é usado AWS SDK para JavaScript a v3. Para obter mais informações sobre como instalar o cliente e o uso, consulte Timestream Write Client - AWS SDK for JavaScript v3.

Veja também Classe UpdateDatabaseCommand UpdateDatabasee.

```
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);
    }
}
```

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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}");
        }
```

Excluir banco de dados

Você pode usar o seguinte trecho de código para excluir um banco de dados.

> **Note**
>
> Esses trechos de código são baseados em exemplos completos de aplicativos em. GitHub Para obter mais informações sobre como começar a usar os aplicativos de amostra, consulteAplicação de exemplo.

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

O trecho a seguir é usado AWS SDK para JavaScript a v3. Para obter mais informações sobre como instalar o cliente e o uso, consulte Timestream Write Client - AWS SDK for JavaScript v3.

Veja também Classe DeleteDatabaseCommand DeleteDatabasee.

```
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;
    }
}
```

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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());
            }
        }
```

Listar bancos de dados

Você pode usar os seguintes trechos de código para listar seus bancos de dados.

> **Note**
>
> Esses trechos de código são baseados em exemplos completos de aplicativos em. GitHub Para obter mais informações sobre como começar a usar os aplicativos de amostra, consulteAplicação de exemplo.

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

O trecho a seguir é usado AWS SDK para JavaScript a v3. Para obter mais informações sobre como instalar o cliente e o uso, consulte Timestream Write Client - AWS SDK for JavaScript v3.

Veja também Classe ListDatabasesCommand ListDatabasese.

```
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);
    }
}
```

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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

Tópicos

- Gravações no armazenamento de memória
- Magnetic store escreve

Gravações no armazenamento de memória

Você pode usar o seguinte trecho de código para criar uma tabela com as gravações do armazenamento magnético desativadas. Como resultado, você só pode gravar dados na janela de retenção do armazenamento de memória.

> **Note**
>
> Esses trechos de código são baseados em exemplos completos de aplicativos em. GitHub Para obter mais informações sobre como começar a usar os aplicativos de amostra, consulteAplicação de exemplo.

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

O trecho a seguir é usado AWS SDK para JavaScript a v3. Para obter mais informações sobre como instalar o cliente e o uso, consulte Timestream Write Client - AWS SDK for JavaScript v3.

Veja também Classe CreateTableCommand CreateTable e.

```
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;
    }
}
```

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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 escreve

Você pode usar o seguinte trecho de código para criar uma tabela com gravações de armazenamento magnético ativadas. Com as gravações do armazenamento magnético, você pode gravar dados na janela de retenção do armazenamento de memória e na janela de retenção do armazenamento magnético.

> **Note**
>
> Esses trechos de código são baseados em exemplos completos de aplicativos em. GitHub Para obter mais informações sobre como começar a usar os aplicativos de amostra, consulteAplicação de exemplo.

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());
            }

        }
```

Descreva a tabela

Você pode usar os trechos de código a seguir para obter informações sobre os atributos da sua tabela.

> **Note**
>
> Esses trechos de código são baseados em exemplos completos de aplicativos em. GitHub Para obter mais informações sobre como começar a usar os aplicativos de amostra, consulteAplicação de exemplo.

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

O trecho a seguir é usado AWS SDK para JavaScript a v3. Para obter mais informações sobre como instalar o cliente e o uso, consulte Timestream Write Client - AWS SDK for JavaScript v3.

Veja também Classe DescribeTableCommand DescribeTable e.

```
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;
    }
}
```

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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());
            }

        }
```

Atualizar tabela

Você pode usar os seguintes trechos de código para atualizar uma tabela.

> **Note**
>
> Esses trechos de código são baseados em exemplos completos de aplicativos em. GitHub Para obter mais informações sobre como começar a usar os aplicativos de amostra, consulteAplicação de exemplo.

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

O trecho a seguir é usado AWS SDK para JavaScript a v3. Para obter mais informações sobre como instalar o cliente e o uso, consulte Timestream Write Client - AWS SDK for JavaScript v3.

Veja também Classe UpdateTableCommand UpdateTable e.

```
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);
}
```

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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());
            }

        }
```

Excluir tabela

Você pode usar os seguintes trechos de código para excluir uma tabela.

> **Note**
>
> Esses trechos de código são baseados em exemplos completos de aplicativos em. GitHub Para obter mais informações sobre como começar a usar os aplicativos de amostra, consulteAplicação de exemplo.

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

O trecho a seguir é usado AWS SDK para JavaScript a v3. Para obter mais informações sobre como instalar o cliente e o uso, consulte Timestream Write Client - AWS SDK for JavaScript v3.

Veja também Classe DeleteTableCommand DeleteTablee.

```
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;
    }
}
```

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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());
            }
        }
```

Listar tabelas

Você pode usar os seguintes trechos de código para listar tabelas.

> **Note**
>
> Esses trechos de código são baseados em exemplos completos de aplicativos em. GitHub Para obter mais informações sobre como começar a usar os aplicativos de amostra, consulteAplicação de exemplo.

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

O trecho a seguir é usado AWS SDK para JavaScript a v3. Para obter mais informações sobre como instalar o cliente e o uso, consulte Timestream Write Client - AWS SDK for JavaScript v3.

Veja também Classe ListTablesCommand ListTablese.

```
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);
    }
}
```

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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}");
    }
```

Gravar dados (inserções e upserts)

Tópicos

- Gravando lotes de registros
- Gravando lotes de registros com atributos comuns
- Atualizando registros
- Exemplo de atributo de várias medidas
- Lidando com falhas de gravação

Gravando lotes de registros

Você pode usar os seguintes trechos de código para gravar dados em uma tabela do Amazon Timestream. Gravar dados em lotes ajuda a otimizar o custo das gravações. Consulte Calculando o número de gravações Para mais informações.

> **Note**
>
> Esses trechos de código são baseados em exemplos completos de aplicativos em. GitHub Para obter mais informações sobre como começar a usar os aplicativos de amostra, consulteAplicação de exemplo.

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

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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());
    }
  }
```

Gravando lotes de registros com atributos comuns

Se seus dados de série temporal tiverem medidas e/ou dimensões comuns em vários pontos de dados, você também poderá usar a seguinte versão otimizada do writeRecords API para inserir dados no Timestream for. LiveAnalytics O uso de atributos comuns com o agrupamento em lotes pode otimizar ainda mais o custo das gravações, conforme descrito emCalculando o número de gravações.

> **Note**
>
> Esses trechos de código são baseados em exemplos completos de aplicativos em. GitHub Para obter mais informações sobre como começar a usar os aplicativos de amostra, consulteAplicação de exemplo.

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

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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());
    }
  }
```

Atualizando registros

Embora as gravações padrão no Amazon Timestream sigam a semântica do primeiro escritor ganha, em que os dados são armazenados somente como acréscimos e os registros duplicados são rejeitados, há aplicativos que exigem a capacidade de gravar dados no Amazon Timestream usando a semântica do último gravador ganha, em que o registro com a versão mais alta é armazenado no sistema. Também existem aplicativos que exigem a capacidade de atualizar os registros existentes. Para lidar com esses cenários, o Amazon Timestream oferece a capacidade de alterar dados. Upsert é uma operação que insere um registro no sistema quando o registro não existe ou atualiza o registro, quando existe um.

Você pode atualizar os registros incluindo a definição `Version` no registro ao enviar uma `WriteRecords` solicitação. O Amazon Timestream armazenará o registro com o recorde mais alto. `Version` O exemplo de código abaixo mostra como você pode alterar os dados:

> **Note**
>
> Esses trechos de código são baseados em exemplos completos de aplicativos em. GitHub Para obter mais informações sobre como começar a usar os aplicativos de amostra, consulteAplicação de exemplo.

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

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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());
    }
```

```
}
```

Exemplo de atributo de várias medidas

Este exemplo ilustra como escrever atributos de várias medidas. Os atributos de várias medidas são úteis quando um dispositivo ou aplicativo que você está rastreando emite várias métricas ou eventos no mesmo timestamp.

> **Note**
>
> Esses trechos de código são baseados em exemplos completos de aplicativos em. GitHub Para obter mais informações sobre como começar a usar os aplicativos de amostra, consulteAplicação de exemplo.

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

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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());
      }
    }
  }
}
```

Lidando com falhas de gravação

As gravações no Amazon Timestream podem falhar por um ou mais dos seguintes motivos:

- Há registros com carimbos de data/hora que estão fora da duração de retenção do armazenamento de memória.
- Há registros contendo dimensões e/ou medidas que excedem os limites definidos pelo Timestream.
- O Amazon Timestream detectou registros duplicados. Os registros são marcados como duplicados quando há vários registros com as mesmas dimensões, carimbos de data/hora e nomes de medidas, mas:
  - Os valores das medidas são diferentes.
  - A versão não está presente na solicitação ou o valor da versão no novo registro é igual ou inferior ao valor existente. Se o Amazon Timestream rejeitar dados por esse motivo, `ExistingVersion` o campo conterá a versão atual `RejectedRecords` do registro conforme armazenada no Amazon Timestream. Para forçar uma atualização, você pode reenviar a solicitação com uma versão do conjunto de registros com um valor maior que o. `ExistingVersion`

Para obter mais informações sobre erros e registros rejeitados, consulte Erros RejectedRecorde.

Se seu aplicativo receber um `RejectedRecordsException` ao tentar gravar registros no Timestream, você pode analisar os registros rejeitados para saber mais sobre as falhas de gravação, conforme mostrado abaixo.

> **Note**
>
> Esses trechos de código são baseados em exemplos completos de aplicativos em. GitHub Para obter mais informações sobre como começar a usar os aplicativos de amostra, consulteAplicação de exemplo.

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

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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());
 }
```

Executar consulta

Tópicos

- Como paginar resultados
- Analisando conjuntos de resultados
- Acessando o status da consulta

Como paginar resultados

Quando você executa uma consulta, o Timestream retorna o conjunto de resultados de forma paginada para otimizar a capacidade de resposta de seus aplicativos. O trecho de código abaixo mostra como você pode paginar por meio do conjunto de resultados. Você deve percorrer todas as páginas do conjunto de resultados até encontrar um valor nulo. Os tokens de paginação expiram 3 horas após serem emitidos pela Timestream for. LiveAnalytics

> **Note**
>
> Esses trechos de código são baseados em exemplos completos de aplicativos em. GitHub Para obter mais informações sobre como começar a usar os aplicativos de amostra, consulteAplicação de exemplo.

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

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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());
            }
        }
```

Analisando conjuntos de resultados

Você pode usar os seguintes trechos de código para extrair dados do conjunto de resultados. Os resultados da consulta ficam acessíveis por até 24 horas após a conclusão da consulta.

> **Note**
>
> Esses trechos de código são baseados em exemplos completos de aplicativos em. GitHub Para obter mais informações sobre como começar a usar os aplicativos de amostra, consulteAplicação de exemplo.

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

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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)}]";
        }
```

Acessando o status da consulta

Você pode acessar o status da consulta por meio de`QueryResponse`, que contém informações sobre o progresso de uma consulta, os bytes verificados por uma consulta e os bytes medidos por uma consulta. Os `bytesScanned` valores `bytesMetered` e são cumulativos e atualizados continuamente durante a paginação dos resultados da consulta. Você pode usar essas informações para entender os bytes digitalizados por uma consulta individual e também usá-las para tomar determinadas decisões. Por exemplo, supondo que o preço da consulta seja de 0,01 USD por GB digitalizado, talvez você queira cancelar consultas que excedam 25 USD por consulta ou GB. X O trecho de código abaixo mostra como isso pode ser feito.

> **Note**
>
> Esses trechos de código são baseados em exemplos completos de aplicativos em. GitHub Para obter mais informações sobre como começar a usar os aplicativos de amostra, consulteAplicação de exemplo.

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

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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());
    }
}
```

Para obter detalhes adicionais sobre como cancelar uma consulta, consulteCancelar consulta.

Executar UNLOAD consulta

Os exemplos de código a seguir chamam uma UNLOAD consulta. Para obter mais informações sobre o UNLOAD, consulte Usando UNLOAD para exportar os resultados da consulta para o S3 do Timestream for LiveAnalytics. Para obter exemplos de UNLOAD consultas, consulteExemplo de caso de uso UNLOAD do Timestream for LiveAnalytics.

Tópicos

- Crie e execute uma UNLOAD consulta
- Analise a UNLOAD resposta e obtenha a contagem de linhas, o link do manifesto e o link de metadados
- Leia e analise o conteúdo do manifesto
- Leia e analise o conteúdo de metadados

Crie e execute uma UNLOAD consulta

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
    }
}
```

Analise a UNLOAD resposta e obtenha a contagem de linhas, o link do manifesto e o link de metadados

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
}
```

Leia e analise o conteúdo do manifesto

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};
}
```

Leia e analise o conteúdo de metadados

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};
}
```

Cancelar consulta

Você pode usar os seguintes trechos de código para cancelar uma consulta.

> **Note**
>
> Esses trechos de código são baseados em exemplos completos de aplicativos em. GitHub Para obter mais informações sobre como começar a usar os aplicativos de amostra, consulteAplicação de exemplo.

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

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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);
            }
        }
```

Criar tarefa de carregamento em lote

Você pode usar os trechos de código a seguir para criar tarefas de carregamento em lote.

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

O trecho a seguir é usado AWS SDK para JavaScript a v3. Para obter mais informações sobre como instalar o cliente e o uso, consulte Timestream Write Client - AWS SDK for JavaScript v3.

Para API obter detalhes, consulte Classe CreateBatchLoadCommand CreateBatchLoadTaske.

```
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();
        }
    }
}
```

Descrever a tarefa de carregamento em lote

Você pode usar os seguintes trechos de código para descrever tarefas de carregamento em lote.

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

O trecho a seguir é usado AWS SDK para JavaScript a v3. Para obter mais informações sobre como instalar o cliente e o uso, consulte Timestream Write Client - AWS SDK for JavaScript v3.

Para API obter detalhes, consulte Classe DescribeBatchLoadCommand DescribeBatchLoadTaske.

```
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>");
        }
    }
}
```

Listar tarefas de carregamento em lote

Você pode usar os trechos de código a seguir para listar as tarefas de carregamento em lote.

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

O trecho a seguir é usado AWS SDK para JavaScript a v3. Para obter mais informações sobre como instalar o cliente e o uso, consulte Timestream Write Client - AWS SDK for JavaScript v3.

Para API obter detalhes, consulte Classe DescribeBatchLoadCommand DescribeBatchLoadTaske.

```
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();
        }
    }
}
```

Retomar a tarefa de carregamento em lote

Você pode usar os trechos de código a seguir para retomar as tarefas de carregamento em lote.

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

O trecho a seguir é usado AWS SDK para JavaScript a v3. Para obter mais informações sobre como instalar o cliente e o uso, consulte Timestream Write Client - AWS SDK for JavaScript v3.

Para API obter detalhes, consulte Classe CreateBatchLoadCommand CreateBatchLoadTaske.

```
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());
            }
        }
    }
}
```

Criar consulta agendada

Você pode usar os seguintes trechos de código para criar uma consulta agendada com mapeamento de várias medidas.

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

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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;
    }
}
```

Listar consulta agendada

Você pode usar os seguintes trechos de código para listar suas consultas agendadas.

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

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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;
     }
 }
```

Descrever a consulta agendada

Você pode usar os seguintes trechos de código para descrever uma consulta agendada.

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

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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;
        }
    }
```

Executar consulta agendada

Você pode usar os seguintes trechos de código para executar uma consulta agendada.

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

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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;
    }
}
```

Atualizar consulta agendada

Você pode usar os seguintes trechos de código para atualizar uma consulta agendada.

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

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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;
    }
}
```

Excluir consulta agendada

Você pode usar os seguintes trechos de código para excluir uma consulta agendada.

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

O trecho a seguir usa o estilo AWS SDK for JavaScript V2. Ele se baseia no aplicativo de exemplo em Node.js, exemplo do Amazon Timestream LiveAnalytics para aplicação em. 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;
     }
 }
```

Usando o carregamento em lote no Timestream para LiveAnalytics

Com o carregamento em lote do Amazon Timestream LiveAnalytics for, você pode CSV ingerir arquivos armazenados no Amazon S3 no Timestream em lotes. Com essa nova funcionalidade, você pode ter seus dados no Timestream LiveAnalytics sem precisar depender de outras ferramentas ou escrever código personalizado. Você pode usar o carregamento em lote para preencher dados com tempos de espera flexíveis, como dados que não são imediatamente necessários para consulta ou análise.

Você pode criar tarefas de carregamento em lote usando o AWS Management Console AWS CLI, o e AWS SDKs o. Para mais informações, consulte Usando o carregamento em lote com o console, Usando o carregamento em lote com o AWS CLI e Usando o carregamento em lote com o AWS SDKs.

Além do carregamento em lote, você pode gravar vários registros ao mesmo tempo com a WriteRecords API operação. Para obter orientação sobre qual usar, consulteEscolhendo entre a WriteRecords API operação e a carga em lote.

Tópicos

- Conceitos de carregamento em lote no Timestream
- Pré-requisitos de carregamento em lote
- Práticas recomendadas de carregamento em lote
- Preparando um arquivo de dados de carregamento em lote
- Mapeamentos de modelos de dados para carregamento em lote
- Usando o carregamento em lote com o console
- Usando o carregamento em lote com o AWS CLI
- Usando o carregamento em lote com o AWS SDKs
- Usando relatórios de erro de carregamento em lote

Conceitos de carregamento em lote no Timestream

Analise os conceitos a seguir para entender melhor a funcionalidade de carregamento em lote.

Tarefa de carregamento em lote — A tarefa que define seus dados de origem e destino no Amazon Timestream. Você especifica configurações adicionais, como o modelo de dados, ao criar a tarefa de carregamento em lote. Você pode criar tarefas de carregamento em lote por meio do AWS Management Console AWS CLI, do e do AWS SDKs.

Destino de importação — O banco de dados e a tabela de destino no Timestream. Para obter informações sobre a criação de bancos de dados Criar um banco de dados do e tabelas, consulte Criar uma tabela e.

Fonte de dados — O CSV arquivo de origem armazenado em um bucket do S3. Para obter informações sobre como preparar o arquivo de dados, consultePreparando um arquivo de dados de carregamento em lote. Para obter informações sobre os preços do S3, consulte os preços do Amazon S3.

Relatório de erro de carregamento em lote — Um relatório que armazena informações sobre os erros de uma tarefa de carregamento em lote. Você define a localização do S3 para relatórios de erro de carregamento em lote como parte de uma tarefa de carregamento em lote. Para obter informações sobre as informações nos relatórios, consulteUsando relatórios de erro de carregamento em lote.

Mapeamento do modelo de dados — Um mapeamento de carga em lote para tempo, dimensões e medidas que vai de uma fonte de dados em um local do S3 para um Timestream de destino para a tabela. LiveAnalytics Para obter mais informações, consulte Mapeamentos de modelos de dados para carregamento em lote.

Pré-requisitos de carregamento em lote

Essa é uma lista de pré-requisitos para usar o carregamento em lote. Para ver as práticas recomendadas, consulte Práticas recomendadas de carregamento em lote.

- Os dados da fonte de carregamento em lote são armazenados no Amazon S3 em CSV formato com cabeçalhos.
- Para cada bucket de origem do Amazon S3, você deve ter as seguintes permissões em uma política anexada:

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

Da mesma forma, para cada bucket de saída do Amazon S3 em que os relatórios são gravados, você deve ter as seguintes permissões em uma política anexada:

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

Por exemplo:

```
{
  "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 para LiveAnalytics analisar o CSV mapeando as informações fornecidas no modelo de dados para cabeçalhos. CSV Os dados devem ter uma coluna que represente o carimbo de data/hora, pelo menos uma coluna de dimensão e pelo menos uma coluna de medida.
- Os buckets do S3 usados com carregamento em lote devem estar na mesma região e na mesma conta da LiveAnalytics tabela Timestream for usada no carregamento em lote.
- A `timestamp` coluna deve ser um tipo de dados longo que represente o tempo desde a época do Unix. Por exemplo, o timestamp `2021-03-25T08:45:21Z` seria representado como. `1616661921` O Timestream suporta segundos, milissegundos, microssegundos e nanossegundos para a precisão do timestamp. Ao usar a linguagem de consulta, você pode converter entre formatos com funções como`to_unixtime`. Para obter mais informações, consulte Funções de data/hora.
- O Timestream suporta o tipo de dados de string para valores de dimensão. Ele suporta tipos de dados longos, duplos, de sequência de caracteres e booleanos para colunas de medida.

Para limites e cotas de carregamento em lote, consulteCarregamento em lote.

Práticas recomendadas de carregamento em lote

O carregamento em lote funciona melhor (alto rendimento) ao seguir as seguintes condições e recomendações:

1. CSVos arquivos enviados para ingestão são pequenos, especificamente com um tamanho de arquivo de 100 MB a 1 GB, para melhorar o paralelismo e a velocidade de ingestão.

2. Evite ingerir dados simultaneamente na mesma tabela (por exemplo, usando a WriteRecords API operação ou uma consulta agendada) quando o carregamento do lote estiver em andamento. Isso pode causar acelerações e a tarefa de carregamento em lote falhará.
3. Não adicione, modifique ou remova arquivos do bucket do S3 usado no carregamento em lote enquanto a tarefa de carregamento em lote estiver em execução.
4. Não exclua nem revogue permissões de tabelas ou fontes, nem relate buckets do S3 que tenham tarefas de carregamento em lote agendadas ou em andamento.
5. Ao ingerir dados com um conjunto de valores de dimensão de alta cardinalidade, siga as orientações em. Recomendações para particionar registros de várias medidas
6. Certifique-se de testar a exatidão dos dados enviando um pequeno arquivo. Você será cobrado por todos os dados enviados para carregamento em lote, independentemente da exatidão. Para obter mais informações sobre preços, consulte os preços do Amazon Timestream.
7. Não retome uma tarefa de carregamento em lote, a menos que `ActiveMagneticStorePartitions` esteja abaixo de 250. O trabalho pode ser interrompido e falhar. O envio de vários trabalhos ao mesmo tempo para o mesmo banco de dados deve reduzir o número.

A seguir estão as melhores práticas do console:

1. Use o construtor somente para modelagem de dados mais simples que usa apenas um nome de medida para registros de várias medidas.
2. Para modelagem de dados mais complexa, useJSON. Por exemplo, use JSON ao usar vários nomes de medidas ao usar registros de várias medidas.

Para obter mais informações sobre o Timestream sobre as LiveAnalytics melhores práticas, consulte. Práticas recomendadas

Preparando um arquivo de dados de carregamento em lote

Um arquivo de dados de origem tem valores separados por delimitador. O termo mais específico, valores separados por vírgula (CSV), é usado genericamente. Os separadores de coluna válidos incluem vírgulas e tubos. Os registros são separados por novas linhas. Os arquivos devem ser armazenados no Amazon S3. Quando você cria uma nova tarefa de carregamento em lote, a localização dos dados de origem é especificada por um ARN para o arquivo. Um arquivo contém cabeçalhos. Uma coluna representa o timestamp. Pelo menos uma outra coluna representa uma medida.

Os buckets S3 usados com carregamento em lote devem estar na mesma região do Timestream da LiveAnalytics tabela usada no carregamento em lote. Não adicione nem remova arquivos do bucket do S3 usado no carregamento em lote após o envio da tarefa de carregamento em lote. Para obter informações sobre como trabalhar com buckets do S3, consulte Introdução ao Amazon S3.

> **Note**
>
> CSVarquivos gerados por alguns aplicativos, como o Excel, podem conter uma marca de ordem de bytes (BOM) que entra em conflito com a codificação esperada. Timestream para tarefas de carregamento LiveAnalytics em lote que fazem referência a um CSV arquivo com BOM um erro quando são processadas programaticamente. Para evitar isso, você pode remover oBOM, que é um caractere invisível.
> Por exemplo, você pode salvar o arquivo de um aplicativo como o Notepad++, que permite especificar uma nova codificação. Você também pode usar uma opção programática que lê a primeira linha, remove o caractere da linha e grava o novo valor na primeira linha do arquivo. Ao salvar do Excel, há várias CSV opções. Salvar com uma CSV opção diferente pode evitar o problema descrito. Mas você deve verificar o resultado porque uma alteração na codificação pode afetar alguns caracteres.

CSVparâmetros de formato

Você usa caracteres de escape quando representa um valor que, de outra forma, é reservado pelos parâmetros de formato. Por exemplo, se o caractere de aspa for uma aspa dupla, para representar uma aspa dupla nos dados, coloque o caractere de escape antes das aspas duplas.

Para obter informações sobre quando especificá-las ao criar uma tarefa de carregamento em lote, consulteCriar uma tarefa de carregamento em lote.

Parâmetro	Opções
Separador de colunas	(Vírgula (',') | Tubo ('|') | Ponto e vírgula (';') | Tab ('/t') | Espaço em branco (''))
Personagem de fuga	nenhuma
Personagem de citação	Console: (Aspas duplas (“) | Aspas simples ('))
Valor nulo	Espaço em branco ('')

Parâmetro	Opções
Corte o espaço em branco	Console: (Não | Sim)

Mapeamentos de modelos de dados para carregamento em lote

O seguinte discute o esquema para mapeamentos de modelos de dados e fornece um exemplo.

Esquema de mapeamentos do modelo de dados

A sintaxe da `CreateBatchLoadTask` solicitação e um `BatchLoadTaskDescription` objeto retornado por uma chamada para `DescribeBatchLoadTask` incluir um `DataModelConfiguration` objeto que inclui o `DataModel` para carregamento em lote. `DataModel`Define mapeamentos de dados de origem armazenados em CSV formato em um local do S3 para um Timestream de destino para banco de dados e tabela. LiveAnalytics

O `TimeColumn` campo indica a localização dos dados de origem para o valor a ser mapeado para a `time` coluna da tabela de destino no Timestream for. LiveAnalytics O `TimeUnit` especifica a unidade para o`TimeColumn`, e pode ser uma das`MILLISECONDS`, `SECONDSMICROSECONDS`, ou`NANOSECONDS`. Também há mapeamentos para dimensões e medidas. Os mapeamentos de dimensões são compostos por colunas de origem e campos de destino.

Para obter mais informações, consulte DimensionMapping. Os mapeamentos para medidas têm duas opções, e. `MixedMeasureMappings` `MultiMeasureMappings`

Para resumir, a `DataModel` contém mapeamentos de uma fonte de dados em um local do S3 para um Timestream de destino para a tabela a seguir. LiveAnalytics

- Tempo
- Dimensões
- Medidas

Se possível, recomendamos que você mapeie os dados de medida para registros de várias medidas no Timestream for. LiveAnalytics Para obter informações sobre os benefícios dos registros de várias medidas, consulteRegistros de várias medidas.

Se várias medidas nos dados de origem forem armazenadas em uma linha, você poderá mapear essas várias medidas para registros de várias medidas no Timestream para uso. LiveAnalytics

`MultiMeasureMappings` Se houver valores que devem ser mapeados para um registro de medida única, você pode usar`MixedMeasureMappings`.

`MixedMeasureMappings`se `MultiMeasureMappings` ambos incluem`MultiMeasureAttributeMappings`. Registros de várias medidas são suportados, independentemente da necessidade de registros de medida única.

Se somente registros de destino de várias medidas forem necessários no Timestream for LiveAnalytics, você poderá definir mapeamentos de medidas na estrutura a seguir.

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

> **Note**
>
> Recomendamos usar `MultiMeasureMappings` sempre que possível.

Se forem necessários registros de destino de medida única no Timestream for LiveAnalytics, você poderá definir mapeamentos de medida na estrutura a seguir.

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

Quando você usa`MultiMeasureMappings`, a `MultiMeasureAttributeMappings` matriz é sempre necessária. Quando você usa a `MixedMeasureMappings` matriz, se `MeasureValueType` for `MULTI` para um determinado`MixedMeasureMapping`, `MultiMeasureAttributeMappings` é necessário para isso`MixedMeasureMapping`. Caso contrário, `MeasureValueType` indica o tipo de medida para o registro de medida única.

De qualquer forma, há uma variedade de opções `MultiMeasureAttributeMapping` disponíveis. Você define os mapeamentos para registros de várias medidas em cada um`MultiMeasureAttributeMapping`, da seguinte forma:

`SourceColumn`

A coluna nos dados de origem que está localizada no Amazon S3.

`TargetMultiMeasureAttributeName`

O nome do nome de várias medidas de destino na tabela de destino. Essa entrada é necessária quando não `MeasureNameColumn` é fornecida. Se `MeasureNameColumn` for fornecido, o valor dessa coluna será usado como nome de várias medidas.

`MeasureValueType`

Um dos`DOUBLE`, `BIGINT``BOOLEAN`,`VARCHAR`, ou`TIMESTAMP`.

Mapeamentos de modelos de dados com exemplo **`MultiMeasureMappings`**

Este exemplo demonstra o mapeamento para registros de várias medidas, a abordagem preferida, que armazena cada valor de medida em uma coluna dedicada. Você pode baixar uma amostra CSV em amostra CSV. A amostra tem os seguintes cabeçalhos para mapear para uma coluna de destino em um Timestream for table. LiveAnalytics

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

Identifique as `measure_name` colunas `time` e no CSV arquivo. Nesse caso, eles são mapeados diretamente para o Timestream para colunas de LiveAnalytics tabelas com os mesmos nomes.

- `time`mapas para `time`
- `measure_name`mapeia para `measure_name` (ou o valor escolhido)

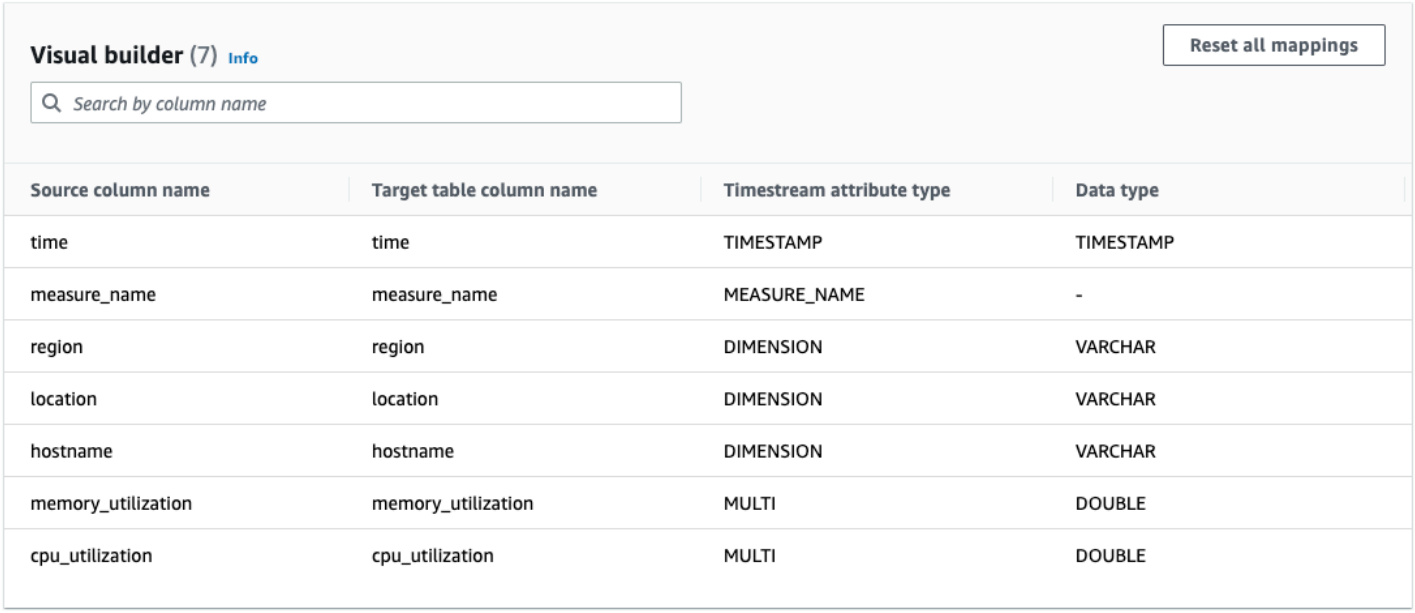
Ao usar oAPI, você especifica `time` no `TimeColumn` campo um valor de unidade de tempo compatível, como `MILLISECONDS` no `TimeUnit` campo. Eles correspondem ao nome da coluna de origem e à entrada da hora do carimbo de data/hora no console. Você pode agrupar ou particionar registros usando `measure_name` o que é definido com a `MeasureNameColumn` chave.

Na amostra,`region`,`location`, e `hostname` são dimensões. As dimensões são mapeadas em uma matriz de `DimensionMapping` objetos.

Para medidas, o valor se `TargetMultiMeasureAttributeName` tornará uma coluna na tabela Timestream for LiveAnalytics . Você pode manter o mesmo nome, como neste exemplo. Ou você pode especificar um novo. `MeasureValueType`é um dos `DOUBLE`,`BIGINT`,`BOOLEAN`,`VARCHAR`, ou`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"
```

```
        }
      ]
    }
}
```



Visual builder (7) Info

Reset all mappings

Search by column name

Source column name	Target table column name	Timestream attribute type	Data type
time	time	TIMESTAMP	TIMESTAMP
measure_name	measure_name	MEASURE_NAME	-
region	region	DIMENSION	VARCHAR
location	location	DIMENSION	VARCHAR
hostname	hostname	DIMENSION	VARCHAR
memory_utilization	memory_utilization	MULTI	DOUBLE
cpu_utilization	cpu_utilization	MULTI	DOUBLE

Mapeamentos de modelos de dados com exemplo `MixedMeasureMappings`

Recomendamos que você use essa abordagem somente quando precisar mapear registros de medida única no Timestream para. LiveAnalytics

Usando o carregamento em lote com o console

A seguir estão as etapas para usar o carregamento em lote com AWS Management Console o. Você pode baixar uma amostra CSV em amostra CSV.

Tópicos

- Acesse o carregamento em lote
- Criar uma tarefa de carregamento em lote
- Retomar uma tarefa de carregamento em lote
- Usando o construtor visual

Acesse o carregamento em lote

Siga estas etapas para acessar o carregamento em lote usando AWS Management Console o.

1. Abra o console do Amazon Timestream.
2. No painel de navegação, escolha Ferramentas de gerenciamento e, em seguida, escolha Tarefas de carregamento em lote.
3. A partir daqui, você pode ver a lista de tarefas de carregamento em lote e detalhar uma determinada tarefa para obter mais detalhes. Você também pode criar e retomar tarefas.

Criar uma tarefa de carregamento em lote

Siga estas etapas para criar uma tarefa de carregamento em lote usando AWS Management Console o.

1. Abra o console do Amazon Timestream.
2. No painel de navegação, escolha Ferramentas de gerenciamento e, em seguida, escolha Tarefas de carregamento em lote.
3. Escolha Criar tarefa de carregamento em lote.
4. Em Destino de importação, escolha o seguinte.

   - Banco de dados de destino — Selecione o nome do banco de dados criado emCriar um banco de dados do .
   - Tabela de destino — Selecione o nome da tabela criada emCriar uma tabela.

   Se necessário, você pode adicionar uma tabela desse painel com o botão Criar nova tabela.
5. Em Localização da fonte de dados S3 em Fonte de dados, selecione o bucket do S3 em que os dados de origem são armazenados. Use o botão Procurar no S3 para visualizar os recursos do S3 aos quais a AWS conta ativa tem acesso ou inserir a localização do S3. URL A fonte de dados deve estar localizada na mesma região.
6. Em Configurações de formato de arquivo (seção expansível), você pode usar as configurações padrão para analisar os dados de entrada. Você também pode escolher Configurações avançadas. A partir daí, você pode escolher os parâmetros de CSV formato e selecionar os parâmetros para analisar os dados de entrada. Para obter informações sobre esses parâmetros, consulteCSVparâmetros de formato.

7. Em Configurar mapeamento do modelo de dados, configure o modelo de dados. Para obter orientações adicionais sobre o modelo de dados, consulte Mapeamentos de modelos de dados para carregamento em lote

   - Em Mapeamento do modelo de dados, escolha Entrada de configuração de mapeamento e escolha uma das opções a seguir.
     - Construtor visual — Para mapear dados visualmente, escolha TargetMultiMeasureNameou MeasureNameColumn. Em seguida, no Visual Builder, mapeie as colunas.

       O Visual Builder detecta e carrega automaticamente os cabeçalhos das colunas de origem do arquivo da fonte de dados quando um único CSV arquivo é selecionado como fonte de dados. Escolha o atributo e o tipo de dados para criar seu mapeamento.

       Para obter informações sobre como usar o construtor visual, consulteUsando o construtor visual.
     - JSONeditor — Um JSON editor de formato livre para configurar seu modelo de dados. Escolha essa opção se você estiver familiarizado com o Timestream for LiveAnalytics e quiser criar mapeamentos avançados de modelos de dados.
     - JSONarquivo do S3 — Selecione um arquivo de JSON modelo que você armazenou no S3. Escolha essa opção se você já configurou um modelo de dados e deseja reutilizá-lo para cargas adicionais em lote.
8. Em Localização do S3 dos registros de erros em Relatório do registro de erros, selecione a localização do S3 que será usada para relatar erros. Para obter informações sobre como usar esse relatório, consulteUsando relatórios de erro de carregamento em lote.
9. Em Tipo de chave de criptografia, escolha uma das opções a seguir.

   - Chave gerenciada pelo Amazon S3 (SSE-S3) — Uma chave de criptografia que o Amazon S3 cria, gerencia e usa para você.
   - AWS KMS key (SSE-KMS) — Uma chave de criptografia protegida por AWS Key Management Service (AWS KMS).
10. Escolha Próximo.
11. Na página Revisar e criar, revise as configurações e edite conforme necessário.

> **Note**
>
> Você não pode alterar as configurações da tarefa de carregamento em lote após a criação da tarefa. Os tempos de conclusão da tarefa variam com base na quantidade de dados que estão sendo importados.

12. Escolha Criar tarefa de carregamento em lote.

Retomar uma tarefa de carregamento em lote

Quando você seleciona uma tarefa de carregamento em lote com o status “Progresso interrompido” que ainda pode ser retomada, você é solicitado a retomar a tarefa. Também há um banner com um botão Retomar tarefa quando você visualiza os detalhes dessas tarefas. As tarefas retomáveis têm uma data de validade. Depois que essa data expirar, as tarefas não poderão ser retomadas.

Usando o construtor visual

Você pode usar o Visual Builder para mapear colunas de dados de origem, um ou mais CSV arquivos armazenados em um bucket do S3, para colunas de destino em um Timestream para tabela. LiveAnalytics

> **Note**
>
> Sua função precisará da `SelectObjectContent` permissão para o arquivo. Sem isso, você precisará adicionar e excluir colunas manualmente.

Modo de colunas de fonte de carregamento automático

O Timestream for LiveAnalytics pode verificar automaticamente o CSV arquivo de origem em busca de nomes de colunas se você especificar apenas um bucket. Quando não há mapeamentos existentes, você pode escolher Importar colunas de origem.

1. Com a opção Visual builder selecionada nas configurações de entrada da configuração de mapeamento, defina a entrada de hora do timestamp. `Millisecondsé` a configuração padrão.
2. Clique no botão Carregar colunas de origem para importar os cabeçalhos das colunas encontrados no arquivo de dados de origem. A tabela será preenchida com os nomes dos cabeçalhos da coluna de origem do arquivo da fonte de dados.

3. Escolha o nome da coluna da tabela de destino, o tipo de atributo Timestream e o tipo de dados para cada coluna de origem.

   Para obter detalhes sobre essas colunas e valores possíveis, consulteMapear campos.

4. Use o drag-to-fill recurso para definir o valor de várias colunas ao mesmo tempo.

Adicionar colunas de origem manualmente

Se você estiver usando um intervalo ou CSV prefixo e não um únicoCSV, poderá adicionar e excluir mapeamentos de colunas do editor visual com os botões Adicionar mapeamento de coluna e Excluir mapeamento de coluna. Também há um botão para redefinir os mapeamentos.

Mapear campos

- Nome da coluna de origem — O nome de uma coluna no arquivo de origem que representa uma medida a ser importada. O Timestream for LiveAnalytics pode preencher esse valor automaticamente quando você usa colunas de origem de importação.
- Nome da coluna da tabela de destino — Entrada opcional que indica o nome da coluna da medida na tabela de destino.
- Tipo de atributo Timestream — O tipo de atributo dos dados na coluna de origem especificada, como. `DIMENSION`
  - TIMESTAMP— Especifica quando uma medida foi coletada.
  - MULTI— Várias medidas são representadas.
  - DIMENSION— Metadados de séries temporais.
  - MEASURE_ NAME — Para registros de medida única, esse é o nome da medida.
- Tipo de dados — O tipo de coluna Timestream, como. `BOOLEAN`
  - BIGINT— Um número inteiro de 64 bits.
  - BOOLEAN— Os dois valores verdadeiros da lógica — verdadeiro e falso.
  - DOUBLE— número de precisão variável de 64 bits.
  - TIMESTAMP— Uma instância no tempo que usa tempo de precisão de nanossegundos e rastreia o tempo desde a época do Unix. UTC

Usando o carregamento em lote com o AWS CLI

Configuração

Para começar a usar o carregamento em lote, siga as etapas a seguir.

1. Instale o AWS CLI usando as instruções emAcessando o Amazon LiveAnalytics Timestream para usar o AWS CLI.
2. Execute o comando a seguir para verificar se os CLI comandos Timestream foram atualizados. Verifique se create-batch-load-task está na lista.

   ```
   aws timestream-write help
   ```
3. Prepare uma fonte de dados usando as instruções emPreparando um arquivo de dados de carregamento em lote.
4. Crie um banco de dados e uma tabela usando as instruções emAcessando o Amazon LiveAnalytics Timestream para usar o AWS CLI.
5. Crie um bucket do S3 para a saída do relatório. O bucket deve estar na mesma região. Para obter mais informações sobre buckets, consulte Criação, configuração e trabalho com buckets do Amazon S3.
6. Crie uma tarefa de carregamento em lote. Para obter as etapas, consulte Criar uma tarefa de carregamento em lote.
7. Confirme o status da tarefa. Para obter as etapas, consulte Descrever a tarefa de carregamento em lote.

Criar uma tarefa de carregamento em lote

Você pode criar uma tarefa de carregamento em lote com o `create-batch-load-task` comando. Ao criar uma tarefa de carregamento em lote usando oCLI, você pode usar um JSON parâmetro,`cli-input-json`, que permite agregar os parâmetros em um único JSON fragmento. Você também pode separar esses detalhes usando vários outros parâmetros`data-model-configuration`, incluindo `data-source-configuration``report-configuration`,`target-database-name`,, `target-table-name` e.

Para ver um exemplo, consulte Exemplo de criação de tarefa de carregamento em lote

Descrever a tarefa de carregamento em lote

Você pode recuperar uma descrição da tarefa de carregamento em lote da seguinte forma.

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

Veja a seguir uma resposta de exemplo.

```
{
    "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
    }
}
```

Listar tarefas de carregamento em lote

Você pode listar as tarefas de carregamento em lote da seguinte forma.

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

Uma saída aparece da seguinte forma.

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

Retomar a tarefa de carregamento em lote

Você pode retomar uma tarefa de carregamento em lote da seguinte maneira.

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

Uma resposta pode indicar sucesso ou conter informações de erro.

Exemplo de criação de tarefa de carregamento em lote

Example

1. Crie um Timestream para o LiveAnalytics banco de dados nomeado `BatchLoad` e uma tabela chamada. `BatchLoadTest` Verifique e, se necessário, ajuste os valores para `MemoryStoreRetentionPeriodInHours` `MagneticStoreRetentionPeriodInDays` e.

   ```
   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. Usando o console, crie um bucket S3 e copie o `sample.csv` arquivo para esse local. Você pode baixar uma amostra CSV em amostra CSV.

3. Usando o console, crie um bucket S3 para o Timestream LiveAnalytics para escrever um relatório se a tarefa de carregamento em lote for concluída com erros.
4. Crie uma tarefa de carregamento em lote. Certifique-se de substituir *$INPUT_BUCKET* e *$REPORT_BUCKET* com os buckets que você criou nas etapas anteriores.

```
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
```

O comando anterior retorna a seguinte saída.

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

5. Verifique o progresso da tarefa. Certifique-se de substituir *$TASK_ID* com o ID da tarefa que foi retornado na etapa anterior.

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

Exemplo de saída

```
{
    "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
    }
}
```

Usando o carregamento em lote com o AWS SDKs

Para obter exemplos de como criar, descrever e listar tarefas de carregamento em lote com o AWS SDKs Criar tarefa de carregamento em loteDescrever a tarefa de carregamento em lote, consulteListar tarefas de carregamento em lote,, Retomar a tarefa de carregamento em lote e.

Usando relatórios de erro de carregamento em lote

As tarefas de carregamento em lote têm um dos seguintes valores de status:

- `CREATED`(Criado) — A tarefa é criada.
- `IN_PROGRESS`(Em andamento) — A tarefa está em andamento.
- `FAILED`(Falha) — A tarefa foi concluída. Mas um ou mais erros foram detectados.
- `SUCCEEDED`(Concluído) — A tarefa foi concluída sem erros.
- `PROGRESS_STOPPED`(Progresso interrompido) — A tarefa foi interrompida, mas não foi concluída. Você pode tentar retomar a tarefa.
- `PENDING_RESUME`(Currículo pendente) — A tarefa está pendente para ser retomada.

Quando há erros, um relatório de registro de erros é criado no bucket do S3 definido para isso. Os erros são categorizados como taskErrors ou fileErrors em matrizes separadas. Veja a seguir um exemplo de relatório de erro.

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

Usando consultas agendadas no Timestream para LiveAnalytics

O recurso de consulta agendada no Amazon Timestream LiveAnalytics for é uma solução totalmente gerenciada, sem servidor e escalável para calcular e armazenar agregados, rollups e outras formas de dados pré-processados, normalmente usadas para painéis operacionais, relatórios comerciais, análises ad hoc e outros aplicativos. As consultas agendadas tornam a análise em tempo real mais eficiente e econômica, para que você possa obter informações adicionais de seus dados e continuar tomando melhores decisões de negócios.

Com as consultas programadas, você define as consultas analíticas em tempo real que computam agregados, rollups e outras operações nos dados. Além disso, o Amazon Timestream executa essas consultas de forma LiveAnalytics periódica e automática e grava de forma confiável os resultados da consulta em uma tabela separada. Normalmente, os dados são calculados e atualizados nessas tabelas em alguns minutos.

Em seguida, você pode direcionar seus painéis e relatórios para consultar as tabelas que contêm dados agregados em vez de consultar as tabelas de origem consideravelmente maiores. Isso leva a ganhos de desempenho e custo que podem exceder ordens de magnitude. Isso ocorre porque as tabelas com dados agregados contêm muito menos dados do que as tabelas de origem, oferecendo consultas mais rápidas e armazenamento de dados mais barato.

Além disso, tabelas com consultas agendadas oferecem todas as funcionalidades existentes de um Timestream para tabela. LiveAnalytics Por exemplo, você pode consultar as tabelas usandoSQL. Você pode visualizar os dados armazenados nas tabelas usando o Grafana. Você também pode ingerir dados na tabela usando o Amazon Kinesis, o AmazonMSK, o AWS IoT Core e o Telegraf. Você pode configurar políticas de retenção de dados nessas tabelas para o gerenciamento automático do ciclo de vida dos dados.

Como a retenção de dados das tabelas que contêm dados agregados está totalmente dissociada da retenção das tabelas de origem, você também pode optar por reduzir a retenção de dados das tabelas de origem e manter os dados agregados por um período muito maior, por uma fração do custo de armazenamento de dados. As consultas programadas tornam a análise em tempo real mais rápida, barata e, portanto, mais acessível a muitos outros clientes, para que eles possam monitorar seus aplicativos e tomar melhores decisões de negócios baseadas em dados.

Tópicos

- Benefícios da consulta programada
- Casos de uso de consultas programadas
- Exemplo: uso de análises em tempo real para detectar pagamentos fraudulentos e tomar melhores decisões comerciais
- Conceitos de consulta agendada
- Expressões de agendamento para consultas agendadas
- Mapeamentos de modelos de dados para consultas agendadas
- Mensagens de notificação de consulta agendada
- Relatórios de erros de consultas agendadas
- Padrões e exemplos de consultas agendadas

Benefícios da consulta programada

A seguir estão os benefícios das consultas agendadas:

- Facilidade operacional — as consultas agendadas são sem servidor e totalmente gerenciadas.
- Desempenho e custo — Como as consultas agendadas pré-computam os agregados, os rollups ou outras operações de análise em tempo real dos seus dados e armazenam os resultados em uma tabela, as consultas que acessam tabelas preenchidas por consultas agendadas contêm menos dados do que as tabelas de origem. Portanto, as consultas executadas nessas tabelas são mais rápidas e baratas. As tabelas preenchidas por cálculos programados contêm menos dados do que suas tabelas de origem e, portanto, ajudam a reduzir o custo de armazenamento. Você também pode reter esses dados por mais tempo no armazenamento de memória por uma fração do custo de reter os dados de origem no armazenamento de memória.
- Interoperabilidade — As tabelas preenchidas por consultas agendadas oferecem todas as funcionalidades existentes do Timestream para LiveAnalytics tabelas e podem ser usadas com todos os serviços e ferramentas que funcionam com o Timestream for. LiveAnalytics Consulte Trabalhando com outros serviços para obter detalhes.

Casos de uso de consultas programadas

Você pode usar consultas agendadas para relatórios comerciais que resumem a atividade do usuário final em seus aplicativos, para que você possa treinar modelos de aprendizado de máquina para personalização. Você também pode usar consultas programadas para alarmes que detectam anomalias, intrusões na rede ou atividades fraudulentas, para que você possa tomar medidas corretivas imediatas.

Além disso, você pode usar consultas agendadas para uma governança de dados mais eficaz. Você pode fazer isso concedendo acesso à tabela de origem exclusivamente às consultas agendadas e fornecendo aos desenvolvedores acesso somente às tabelas preenchidas pelas consultas agendadas. Isso minimiza o impacto de consultas não intencionais e de longa duração.

Exemplo: uso de análises em tempo real para detectar pagamentos fraudulentos e tomar melhores decisões comerciais

Considere um sistema de pagamento que processa transações enviadas de vários point-of-sale terminais distribuídos nas principais cidades metropolitanas dos Estados Unidos. Você quer usar o Amazon Timestream LiveAnalytics para armazenar e analisar os dados da transação, para que você possa detectar transações fraudulentas e executar consultas analíticas em tempo real. Essas consultas podem ajudá-lo a responder perguntas comerciais, como identificar os point-of-sale terminais mais movimentados e menos usados por hora, a hora mais movimentada do dia em cada cidade e a cidade com mais transações por hora.

O sistema processa aproximadamente 100 mil transações por minuto. Cada transação armazenada no Amazon Timestream LiveAnalytics é de 100 bytes. Você configurou 10 consultas que são executadas a cada minuto para detectar vários tipos de pagamentos fraudulentos. Você também criou 25 consultas que agregam e dividem/dividem seus dados em várias dimensões para ajudar a responder suas perguntas comerciais. Cada uma dessas consultas processa os dados da última hora.

Você criou um painel para exibir os dados gerados por essas consultas. O painel contém 25 widgets, é atualizado a cada hora e normalmente é acessado por 10 usuários a qualquer momento. Finalmente, seu armazenamento de memória é configurado com um período de retenção de dados de 2 horas e o armazenamento magnético está configurado para ter um período de retenção de dados de 6 meses.

Nesse caso, você pode usar consultas de análise em tempo real que recalculam os dados sempre que o painel é acessado e atualizado, ou usar tabelas derivadas para o painel. O custo da consulta para painéis baseados em consultas de análise em tempo real será de $120,70 por mês. Por outro lado, o custo das consultas de painel baseadas em tabelas derivadas será de 12,27 USD por mês (consulte Amazon Timestream para ver os preços). LiveAnalytics Nesse caso, o uso de tabelas derivadas reduz o custo da consulta em aproximadamente 10 vezes.

Conceitos de consulta agendada

Cadeia de caracteres de consulta - Essa é a consulta cujo resultado você está pré-computando e armazenando em outro Timestream for table. LiveAnalytics Você pode definir uma consulta agendada usando toda a área de SQL superfície do Timestream for LiveAnalytics, que fornece a flexibilidade de escrever consultas com expressões de tabela comuns, consultas aninhadas, funções de janela ou qualquer tipo de função agregada e escalar compatível com o Timestream para linguagem de consulta. LiveAnalytics

Expressão de agendamento - permite que você especifique quando suas instâncias de consulta agendadas serão executadas. Você pode especificar as expressões usando uma expressão cron (como executar às 8h UTC todos os dias) ou uma expressão de taxa (como executar a cada 10 minutos).

Configuração de destino - Permite especificar como mapear o resultado de uma consulta agendada na tabela de destino em que os resultados dessa consulta agendada serão armazenados.

Configuração de notificação - Timestream para executar LiveAnalytics automaticamente instâncias de uma consulta agendada com base em sua expressão de agendamento. Você recebe uma

notificação para cada consulta executada em um SNS tópico que você configura ao criar uma consulta agendada. Essa notificação especifica se a instância foi executada com sucesso ou se encontrou algum erro. Além disso, ele fornece informações como os bytes medidos, os dados gravados na tabela de destino, a hora da próxima invocação e assim por diante.

Veja a seguir um exemplo desse tipo de mensagem de notificação.

```
{
    "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
        }
    }
}
```

Nessa mensagem de notificação, `bytesMetered` estão os bytes que a consulta examinou na tabela de origem e dataWrites os bytes gravados na tabela de destino.

> **Note**
>
> Se você estiver consumindo essas notificações programaticamente, saiba que novos campos poderão ser adicionados à mensagem de notificação no futuro.

Local do relatório de erros - as consultas agendadas são executadas e armazenadas de forma assíncrona na tabela de destino. Se uma instância encontrar algum erro (por exemplo, dados inválidos que não puderam ser armazenados), os registros que encontraram erros serão gravados em um relatório de erros no local do relatório de erros especificado na criação de uma consulta

agendada. Você especifica o bucket e o prefixo do S3 para o local. Timestream for LiveAnalytics acrescenta o nome da consulta agendada e a hora da invocação a esse prefixo para ajudá-lo a identificar os erros associados a uma instância específica de uma consulta agendada.

Marcação - opcionalmente, você pode especificar tags que podem ser associadas a uma consulta agendada. Para obter mais detalhes, consulte Como marcar o Timestream para recursos. LiveAnalytics

Exemplo

No exemplo a seguir, você calcula um agregado simples usando uma consulta agendada:

```
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`- Neste exemplo, você notará que a consulta está aceitando um parâmetro com nome especial`@scheduled_runtime`. Esse é um parâmetro especial (do tipo Timestamp) que o serviço define ao invocar uma instância específica de uma consulta agendada para que você possa controlar deterministicamente o intervalo de tempo durante o qual uma instância específica de uma consulta agendada analisa os dados na tabela de origem. Você pode usar `@scheduled_runtime` em sua consulta em qualquer local em que um tipo de carimbo de data/hora seja esperado.

Considere um exemplo em que você define uma expressão de cronograma: cron (0/5 * * *? *) onde a consulta agendada será executada no minuto 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55 de cada hora. Para a instância que é acionada em 01/12/2021 00:05:00, o parâmetro @scheduled_runtime é inicializado com esse valor, de forma que a instância neste momento opere com dados no intervalo 2021-11-30 23:55:00 a 2021-12-01 00:06:00.

Instâncias com intervalos de tempo sobrepostos - Como você verá neste exemplo, duas instâncias subsequentes de uma consulta agendada podem se sobrepor em seus intervalos de tempo. Isso é algo que você pode controlar com base em seus requisitos, nos predicados de tempo que você especifica e na expressão do cronograma. Nesse caso, essa sobreposição permite que esses cálculos atualizem os agregados com base em qualquer dado cuja chegada tenha sido um pouco atrasada, até 10 minutos neste exemplo. A execução da consulta acionada em 2021-12-01 00:00:00 cobrirá o intervalo de tempo 2021-11-30 23:50:00 a 2021-12-30 00:01:00 e a execução da consulta acionada em 2021-12-01 00:05:00 cobrirá o intervalo 2021-11-30 23:55:00 a 2021-12-01 00:06:00.

Para garantir a exatidão e garantir que os agregados armazenados na tabela de destino correspondam aos agregados calculados a partir da tabela de origem, o Timestream for LiveAnalytics garante que o cálculo em 01/12/2021 00:05:00 seja executado somente após a conclusão do cálculo em 01/12/2021 00:00:00. Os resultados dos últimos cálculos podem atualizar qualquer agregado materializado anteriormente se um valor mais novo for gerado. Internamente, o Timestream for LiveAnalytics usa versões de registro em que os registros gerados pelas últimas instâncias de uma consulta agendada receberão um número de versão maior. Portanto, os agregados calculados pela invocação em 01/12/2021 00:05:00 podem atualizar os agregados calculados pela invocação em 01/12/2021 00:00:00, supondo que dados mais recentes estejam disponíveis na tabela de origem.

Acionadores automáticos versus acionadores manuais - Depois que uma consulta agendada for criada, o Timestream for LiveAnalytics executará automaticamente as instâncias com base na programação especificada. Esses gatilhos automatizados são gerenciados inteiramente pelo serviço.

No entanto, pode haver cenários em que talvez você queira iniciar manualmente algumas instâncias de uma consulta agendada. Os exemplos incluem se uma instância específica falhou na execução de uma consulta, se houve chegada tardia de dados ou atualizações na tabela de origem após a execução automática do agendamento ou se você quiser atualizar a tabela de destino para intervalos de tempo que não são cobertos por execuções de consultas automatizadas (por exemplo, para intervalos de tempo antes da criação de uma consulta agendada).

Você pode usar o ExecuteScheduledQuery API para iniciar manualmente uma instância específica de uma consulta agendada passando o InvocationTime parâmetro, que é um valor usado para o parâmetro @scheduled_runtime. A seguir estão algumas considerações importantes ao usar o ExecuteScheduledQuery API:

- Se você estiver acionando várias dessas invocações, precisará garantir que essas invocações não gerem resultados em intervalos de tempo sobrepostos. Se você não puder garantir intervalos de tempo não sobrepostos, certifique-se de que essas execuções de consulta sejam iniciadas sequencialmente uma após a outra. Se você iniciar simultaneamente várias execuções de consulta que se sobrepõem em seus intervalos de tempo, poderá ver falhas de gatilho nas quais poderá ver conflitos de versão nos relatórios de erro dessas execuções de consulta.
- Você pode iniciar as invocações com qualquer valor de timestamp para @scheduled_runtime. Portanto, é sua responsabilidade definir adequadamente os valores para que os intervalos de tempo apropriados sejam atualizados na tabela de destino correspondente aos intervalos em que os dados foram atualizados na tabela de origem.

Expressões de agendamento para consultas agendadas

Você pode criar consultas programadas em uma programação automatizada usando o Amazon LiveAnalytics Timestream para consultas programadas que usam expressões cron ou de taxa. Todas as consultas agendadas usam o fuso UTC horário, e a precisão mínima possível para agendamentos é de 1 minuto.

Duas maneiras de especificar as expressões do cronograma são cron e rate. As expressões Cron oferecem um controle de cronograma mais refinado, enquanto as expressões de taxa são mais simples de expressar, mas não têm o controle refinado.

Por exemplo, com uma expressão cron, você pode definir uma consulta agendada que é acionada em um horário específico em um determinado dia de cada semana ou mês, ou em um minuto específico a cada hora somente de segunda a sexta-feira e assim por diante. Por outro lado, as expressões de taxa iniciam uma consulta agendada em uma taxa regular, como uma vez a cada minuto, hora ou dia, a partir da hora exata em que a consulta agendada é criada.

Expressão cron

- Sintaxe

```
cron(fields)
```

  Expressões cron têm seis campos obrigatórios, que são separados por um espaço em branco.

Campo	Valores	Curingas
minutos	0-59	, - * /
Horas	0-23	, - * /
D ay-of-month	1-31	, - * ? / L W
Mês	1-12 ou JAN - DEC	, - * /
D ay-of-week	1-7 ou SUN - SAT	, - * ? L #
Ano	1970-2199	, - * /

Caracteres curinga

- O caractere curinga *, * (vírgula) inclui valores adicionais. No campo Mês,JAN,FEB, MAR incluiria janeiro, fevereiro e março.
- O curinga *-* (traço) especifica os intervalos. No campo Dia, 1-15 incluiria dias 1 a 15 do mês especificado.
- O curinga *** (asterisco) inclui todos os valores no campo. No campo Horas, *** incluiria todas as horas. Você não pode usar *** nos Day-of-week campos Day-of-month e. Se você usá-lo em um, você deve usar *? * no outro.
- O curinga */* (barra invertida) especifica incrementos. No campo Minutos, você pode inserir 1/10 para especificar a cada 10 minutos, começando pelo primeiro minuto da hora (por exemplo, 11, 21 e 31 minutos e assim por diante).
- O *? * (ponto de interrogação) curinga especifica um ou outro. No Day-of-month campo, você poderia inserir *7* e, se não se importasse em que dia da semana era o 7º, você poderia inserir *? * no Day-of-week campo.
- O caractere curinga *L* nos Day-of-week campos Day-of-month ou especifica o último dia do mês ou da semana.
- O curinga W no Day-of-month campo especifica um dia da semana. No Day-of-month campo, 3W especifica o dia da semana mais próximo do terceiro dia do mês.
- O curinga *#* no Day-of-week campo especifica uma determinada instância do dia da semana especificado em um mês. Por exemplo, 3#2 seria a segunda terça-feira do mês: o 3 refere-se a terça-feira, porque é o terceiro dia de cada semana, e o 2 refere-se ao segundo dia desse tipo dentro do mês.

> **Note**
>
> Se você usar um caractere '#', poderá definir somente uma expressão no day-of-week campo. Por exemplo, "3#1,6#3" não é válido porque é interpretado como duas expressões.

Limitações

- Você não pode especificar os Day-of-week campos Day-of-month e na mesma expressão cron. Se você especificar um valor (ou um *) em um dos campos, deverá usar um *? * (ponto de interrogação) no outro.

- As expressões Cron que levam a taxas mais rápidas do que 1 minuto não têm suporte.

Exemplos

Minutos	Horas	Dia do mês	Mês	Dia da semana	Ano	Significado
0	10	*	*	?	*	Corra às 10:00 da manhã (UTC) todos os dias.
15	12	*	*	?	*	Corra às 12h15 (UTC) todos os dias.
0	18	?	*	MON-FRI	*	Corra às 18h (UTC) de segunda a sexta-feira.
0	8	1	*	?	*	Corra às 8:00 da manhã (UTC) todo primeiro dia do mês.
0/15	*	*	*	?	*	Corra a cada 15 minutos.

Minutos	Horas	Dia do mês	Mês	Dia da semana	Ano	Significado
0/10	*	*	*	MON-FRI	*	Corra a cada 10 minutos, de segunda a sexta-feira.
0/5	8-17	?	*	MON-FRI	*	Corra a cada 5 minutos, de segunda a sexta-feira, das 8h às 17h55 ()UTC.

Expressões rate

- Uma expressão rate começa quando você cria a regra de evento programado e, em seguida, e a executa em sua programação definida. As expressões rate tem dois campos obrigatórios. Os campos são separados por um espaço em branco.

  Sintaxe

  ```
  rate(value unit)
  ```

  - `value`: Um número positivo.
  - `unit`: A unidade de tempo. Unidades diferentes são necessárias para valores de 1 (por exemplo, minuto) e valores acima de 1 (por exemplo, minutos). Valores válidos: minuto | minutos | hora | horas | dia | dias

Mapeamentos de modelos de dados para consultas agendadas

O Timestream for LiveAnalytics oferece suporte à modelagem flexível de dados em suas tabelas e essa mesma flexibilidade se aplica aos resultados de consultas agendadas que são materializadas em outro Timestream para tabela. LiveAnalytics Com consultas agendadas, você pode consultar qualquer tabela, quer ela tenha dados em registros de várias medidas ou registros de medida única, e gravar os resultados da consulta usando registros de várias medidas ou de medida única.

Você usa o TargetConfiguration na especificação de uma consulta agendada para mapear os resultados da consulta para as colunas apropriadas na tabela derivada de destino. As seções a seguir descrevem as diferentes maneiras de especificar isso TargetConfiguration para obter diferentes modelos de dados na tabela derivada. Especificamente, você verá:

- Como gravar em registros de várias medidas quando o resultado da consulta não tem um nome de medida e você especifica o nome da medida de destino no TargetConfiguration.
- Como você usa o nome da medida no resultado da consulta para gravar registros de várias medidas.
- Como você pode definir um modelo para gravar vários registros com diferentes atributos de várias medidas.
- Como você pode definir um modelo para gravar em registros de medida única na tabela derivada.
- Como você pode consultar registros de medida única e/ou registros de várias medidas em uma consulta programada e ter os resultados materializados em um registro de medida única ou em um registro de várias medidas, o que permite escolher a flexibilidade dos modelos de dados.

Exemplo: nome da medida alvo para registros de várias medidas

Neste exemplo, você verá que a consulta está lendo dados de uma tabela com dados de várias medidas e gravando os resultados em outra tabela usando registros de várias medidas. O resultado da consulta agendada não tem uma coluna de nome de medida natural. Aqui, você especifica o nome da medida na tabela derivada usando a TargetMultiMeasureName propriedade no 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"
        }
    }
}
```

O mapeamento neste exemplo cria um registro de várias medidas com nome de medida dashboard-metrics e nomes de atributos, 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 Observe o uso opcional de TargetMultiMeasureAttributeName renomear as colunas de saída da consulta para um nome de atributo diferente usado para materialização de resultados.

Veja a seguir o esquema da tabela de destino quando essa consulta agendada for materializada. Como você pode ver no Timestream do tipo de LiveAnalytics atributo no resultado a seguir,

os resultados são materializados em um registro de várias medidas com um nome de medida únicadashboard-metrics, conforme mostrado no esquema de medidas.

Coluna	Tipo	Fluxo de tempo para o tipo de LiveAnalytics atributo
região	varchar	DIMENSION
nome_medida	varchar	MEASURE_NAME
horário	timestamp	TIMESTAMP
CpuSystemP100	double	MULTI
avgMemCached	double	MULTI
min_cpu_idle_1h	double	MULTI
avg_disk_free_1h	double	MULTI
avg_disk_used_1h	double	MULTI
totalDiskWrites	double	MULTI
mem_max_used_1h	double	MULTI
min_mem_free_1h	double	MULTI
CpuUserP100	double	MULTI

A seguir estão as medidas correspondentes obtidas com uma SHOW MEASURES consulta.

nome_medida	data_type	Dimensões
métricas do painel	multi	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]

Exemplo: uso do nome da medida da consulta agendada em registros de várias medidas

Neste exemplo, você verá uma consulta lendo uma tabela com registros de medida única e materializando os resultados em registros de várias medidas. Nesse caso, o resultado da consulta agendada tem uma coluna cujos valores podem ser usados como nomes de medidas na tabela de destino em que os resultados da consulta agendada são materializados. Em seguida, você pode especificar o nome da medida para o registro de várias medidas na tabela derivada usando a MeasureNameColumn propriedade em 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"
        }
    }
}
```

O mapeamento neste exemplo criará registros de várias medidas com os atributos avg_1h, p0_1h, sum_1h, p100_1h e usará os valores da coluna measure_name no resultado da consulta como o nome da medida para os registros de várias medidas na tabela de destino. Além disso, observe que os exemplos anteriores usam opcionalmente o TargetMultiMeasureAttributeName com um subconjunto dos mapeamentos para renomear os atributos. Por exemplo, min_1h foi renomeado para p0_1h e max_1h foi renomeado para p100_1h.

Veja a seguir o esquema da tabela de destino quando essa consulta agendada for materializada. Como você pode ver no Timestream do tipo de LiveAnalytics atributo no resultado a seguir, os resultados são materializados em um registro de várias medidas. Se você observar o esquema de medidas, nove nomes de medidas diferentes foram ingeridos e correspondem aos valores vistos nos resultados da consulta.

Coluna	Tipo	Fluxo de tempo para o tipo de LiveAnalytics atributo
região	varchar	DIMENSION
nome_medida	varchar	MEASURE_NAME
horário	timestamp	TIMESTAMP
soma_1h	double	MULTI
p100_1h	double	MULTI
p0_1h	double	MULTI
avg_1h	double	MULTI

A seguir estão as medidas correspondentes obtidas com uma SHOW MEASURES consulta.

nome_medida	data_type	Dimensões
cpu_idle	multi	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
sistema_cpu	multi	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
usuário_CPU	multi	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
disk_free	multi	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]

nome_medida	data_type	Dimensões
disk_io_writes	multi	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
disk_used	multi	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
memória armazenada em cache	multi	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
sem memória	multi	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
sem memória	multi	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]

Exemplo: mapeamento de resultados para diferentes registros de várias medidas com atributos diferentes

O exemplo a seguir mostra como você pode mapear diferentes colunas no resultado da consulta em diferentes registros de várias medidas com nomes de medidas diferentes. Se você ver a seguinte definição de consulta agendada, o resultado da consulta tem as seguintes colunas: 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_1h usuário_1h, max_cpu_system_1h, min_cpu_system_1h. `region`oné mapeado para a dimensão e `hour` é mapeado para a coluna de tempo.

A MixedMeasureMappings propriedade em TargetConfiguration. TimestreamConfigurationespecifica como mapear as medidas para registros de várias medidas na tabela derivada.

Neste exemplo específico, avg_mem_cached_1h, min_mem_free_1h, max_mem_used_1h são usados em um registro de várias medidas com nome de medida mem_aggregates, total_disk_io_writes_1h, avg_disk_used_1h, avg_disk_free_1h são usados em outro registro de várias medidas com medida nome de disk_aggregates e, finalmente, max_cpu_user_1h, max_cpu_system_1h, min_cpu_system_1h são usados em outro registro de várias medidas com o nome de medida cpu_aggregates.

Nesses mapeamentos, você também pode usar opcionalmente TargetMultiMeasureAttributeName para renomear a coluna de resultados da consulta para ter um nome de atributo diferente na tabela de destino. Por exemplo, a coluna de resultados avg_mem_cached_1h é renomeada para, total_disk_io_writes_1h é renomeada para, etc. avgMemCached totalIOWrites

Quando você define os mapeamentos para registros de várias medidas, o Timestream for LiveAnalytics inspeciona cada linha nos resultados da consulta e ignora automaticamente os valores da coluna que têm valores. NULL Como resultado, no caso de mapeamentos com vários nomes de medidas, se todos os valores da coluna desse grupo no mapeamento forem NULL de uma determinada linha, nenhum valor desse nome de medida será ingerido para essa linha.

Por exemplo, no mapeamento a seguir, avg_mem_cached_1h, min_mem_free_1h e max_mem_used_1h são mapeados para medir o nome mem_aggregates. Se, para uma determinada linha do resultado da consulta, todos esses valores da coluna foremNULL, o Timestream for LiveAnalytics não ingerirá a medida mem_aggregates dessa linha. Se todas as nove colunas de uma determinada linha foremNULL, você verá um erro do usuário relatado em seu relatório de erros.

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

Veja a seguir o esquema da tabela de destino quando essa consulta agendada for materializada.

Coluna	Tipo	Fluxo de tempo para o tipo de LiveAnalytics atributo
região	varchar	DIMENSION
nome_medida	varchar	MEASURE_NAME
horário	timestamp	TIMESTAMP
minCpuIdle	double	MULTI
max_cpu_system_1h	double	MULTI
usuário_cpu_máximo 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

A seguir estão as medidas correspondentes obtidas com uma SHOW MEASURES consulta.

nome_medida	data_type	Dimensões
agregados de CPU	multi	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
agregados de disco	multi	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
mem_aggregates	multi	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]

Exemplo: mapeamento de resultados para registros de medida única com o nome da medida dos resultados da consulta

Veja a seguir um exemplo de uma consulta programada cujos resultados são materializados em registros de medida única. Neste exemplo, o resultado da consulta tem a coluna measure_name cujos valores serão usados como nomes de medidas na tabela de destino. Você usa o MixedMeasureMappings atributo no TargetConfiguration. TimestreamConfiguration para especificar o mapeamento da coluna de resultados da consulta para a medida escalar na tabela de destino.

No exemplo de definição a seguir, espera-se que o resultado da consulta tenha nove valores distintos de measure_name. Você lista todos esses nomes de medida no mapeamento e especifica qual coluna usar para o valor de medida única desse nome de medida. Por exemplo, nesse mapeamento, se o nome da medida memory_cached for visto para uma determinada linha de resultados, o valor na coluna avg_1h será usado como o valor da medida quando os dados forem gravados na tabela de destino. Opcionalmente, você pode usar TargetMeasureName para fornecer um novo nome de medida para esse valor.

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

Veja a seguir o esquema da tabela de destino quando essa consulta agendada for materializada. Como você pode ver no esquema, a tabela está usando registros de medida única. Se você listar o esquema de medidas da tabela, verá as nove medidas gravadas com base no mapeamento fornecido na especificação.

Coluna	Tipo	Fluxo de tempo para o tipo de LiveAnalytics atributo
região	varchar	DIMENSION
nome_medida	varchar	MEASURE_NAME
horário	timestamp	TIMESTAMP

Coluna	Tipo	Fluxo de tempo para o tipo de LiveAnalytics atributo
valor_medida::duplo	double	MEASURE_VALUE

A seguir estão as medidas correspondentes obtidas com uma SHOW MEASURES consulta.

nome_medida	data_type	Dimensões
AvgMemCached	double	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
MinMemFree	double	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
cpu_idle	double	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
sistema_cpu	double	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
usuário_CPU	double	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
disk_free	double	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
disk_used	double	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
maxMemUsed	double	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
total-disk-io-writes	double	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]

Exemplo: mapeamento de resultados para registros de medida única com colunas de resultados de consulta como nomes de medidas

Neste exemplo, você tem uma consulta cujos resultados não têm uma coluna de nome de medida. Em vez disso, você deseja que o nome da coluna do resultado da consulta seja o nome da medida ao mapear a saída para registros de medida única. Anteriormente, houve um exemplo em que um resultado semelhante foi gravado em um registro de várias medidas. Neste exemplo, você verá como mapeá-lo para registros de medida única, se isso se adequar ao cenário do seu aplicativo.

Novamente, você especifica esse mapeamento usando a MixedMeasureMappings propriedade em TargetConfiguration. TimestreamConfiguration. No exemplo a seguir, você vê que o resultado da consulta tem nove colunas. Você usa as colunas de resultados como nomes de medidas e os valores como valores de medida única.

Por exemplo, para uma determinada linha no resultado da consulta, o nome da coluna avg_mem_cached_1h é usado como o nome e o valor da coluna associados à coluna, e avg_mem_cached_1h é usado como o valor da medida para o registro de medida única. Você também pode usar TargetMeasureName para usar um nome de medida diferente na tabela de destino. Por exemplo, para valores na coluna sum_1h, o mapeamento especifica o uso de total_disk_io_writes_1h como o nome da medida na tabela de destino. Se o valor de qualquer coluna forNULL, a medida correspondente será ignorada.

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

Veja a seguir o esquema da tabela de destino quando essa consulta agendada for materializada. Como você pode ver, a tabela de destino está armazenando registros com valores de medida única do tipo double. Da mesma forma, o esquema de medidas da tabela mostra os nove nomes de medidas. Observe também que o nome da medida total_disk_io_writes_1h está presente desde que o mapeamento foi renomeado sum_1h para total_disk_io_writes_1h.

Coluna	Tipo	Fluxo de tempo para o tipo de LiveAnalytics atributo
região	varchar	DIMENSION
nome_medida	varchar	MEASURE_NAME
horário	timestamp	TIMESTAMP
valor_medida::duplo	double	MEASURE_VALUE

A seguir estão as medidas correspondentes obtidas com uma SHOW MEASURES consulta.

nome_medida	data_type	Dimensões
avg_disk_free_1h	double	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
avg_disk_used_1h	double	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
avg_mem_cached_1h	double	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
max_cpu_system_1h	double	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
usuário_cpu_máximo 1h	double	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
mem_max_used_1h	double	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
min_cpu_idle_1h	double	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
min_mem_free_1h	double	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]
total-disk-io-writes	double	[{'dimension_name': 'região', 'tipo de dados': 'varchar'}]

Mensagens de notificação de consulta agendada

Esta seção descreve as mensagens enviadas pelo Timestream para LiveAnalytics criar, excluir, executar ou atualizar o estado de uma consulta agendada.

Nome da mensagem de notificação	Estrutura	Descrição
CreatingNotificationMessage	`CreatingNotificationMessage { String arn; NotificationType type; }`	Essa mensagem de notificação é enviada antes de enviar a resposta para `CreateScheduledQuery`. A consulta agendada é ativada após o envio dessa notificação. arn - A ARN da consulta agendada que está sendo criada. tipo - SCHEDULED _ QUERY _ CREATING
UpdateNotificationMessage	`UpdateNotificationMessage { String arn; NotificationType type; QueryState state; }`	Essa mensagem de notificação é enviada quando uma consulta agendada é atualizada. O Timestream for LiveAnalytics pode desativar a consulta agendada, automaticamente, caso seja encontrado um erro não recuperável, como: • AssumeRole falha • Quaisquer erros 4xx encontrados ao se comunicar com a especificação KMS de uma KMS chave gerenciada pelo cliente. • Quaisquer erros 4xx encontrados durante a execução da consulta agendada.

Nome da mensagem de notificação	Estrutura	Descrição
		• Quaisquer erros 4xx encontrados durante a ingestão dos resultados da consulta arn - A ARN da consulta agendada que está sendo atualizada. tipo - SCHEDULED _ QUERY _ UPDATE estado - ENABLED ou DISABLED
DeleteNotificationMessage	`DeletionNotificationMessage { String arn; NotificationType type; }`	Essa mensagem de notificação é enviada quando uma consulta agendada é excluída. arn - A ARN da consulta agendada que está sendo criada. tipo - SCHEDULED _ QUERY _ DELETED

Nome da mensagem de notificação	Estrutura	Descrição
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 {`	Essa mensagem de notificação é enviada depois que a consulta agendada é executada e os resultados são ingeridos com sucesso. ARN- A ARN da consulta agendada que está sendo excluída. NotificationType- AUTO _ TRIGGER _ SUCCESS ou MANUAL _ TRIGGER _SUCCESS. nextInvocationEpochSegundo - Na próxima vez, o Timestream for LiveAnalytics executará a consulta agendada. runSummary- Informações sobre a execução da consulta agendada.

Nome da mensagem de notificação	Estrutura	Descrição
	S3ReportLocation s3ReportLocation; } S3ReportLocation { String bucketName; String objectKey; }	

Nome da mensagem de notificação	Estrutura	Descrição
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;`	Essa mensagem de notificação é enviada quando ocorre uma falha durante a execução de uma consulta agendada ou ao ingerir os resultados da consulta. arn - A ARN consulta agendada que está sendo executada. digite - AUTO _ TRIGGER _ FAILURE ou MANUAL _ TRIGGER _FAILURE. runSummary- Informações sobre a execução da consulta agendada.

Nome da mensagem de notificação	Estrutura	Descrição
	`}` `S3ReportLocation {` `  String bucketName;` `  String objectKey;` `}`	

Relatórios de erros de consultas agendadas

Esta seção descreve a localização, o formato e os motivos dos relatórios de erro gerados pelo Timestream para LiveAnalytics quando erros são encontrados ao executar consultas agendadas.

Tópicos

- Motivos dos relatórios de erros de consulta agendados
- Localização dos relatórios de erros da consulta agendada
- Formato de relatórios de erros de consulta agendada
- Tipos de erro de consulta agendada
- Exemplo de relatórios de erros de consultas agendadas

Motivos dos relatórios de erros de consulta agendados

Os relatórios de erros são gerados para erros recuperáveis. Os relatórios de erros não são gerados para erros não recuperáveis. O Timestream for LiveAnalytics pode desativar as consultas agendadas automaticamente quando erros não recuperáveis são encontrados. Isso inclui:

- `AssumeRole`falha
- Quaisquer erros 4xx encontrados ao se comunicar com KMS quando uma chave gerenciada pelo cliente KMS é especificada
- Quaisquer erros 4xx encontrados quando uma consulta agendada é executada
- Quaisquer erros 4xx encontrados durante a ingestão dos resultados da consulta

Para erros não recuperáveis, o Timestream for LiveAnalytics envia uma notificação de falha com uma mensagem de erro não recuperável. Também é enviada uma notificação de atualização que indica que a consulta agendada está desativada.

Localização dos relatórios de erros da consulta agendada

Um local de relatório de erro de consulta agendada tem a seguinte convenção de nomenclatura:

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

Veja a seguir um exemplo de consulta agendadaARN:

```
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 indica consultas agendadas agendadas automaticamente pelo Timestream para e LiveAnalytics *Manual* indica consultas programadas acionadas manualmente por um usuário por meio de uma `ExecuteScheduledQuery` API ação no Amazon LiveAnalytics Timestream for Query. Para obter mais informações sobre`ExecuteScheduledQuery`, consulte ExecuteScheduledQuery.

Formato de relatórios de erros de consulta agendada

Os relatórios de erros têm o seguinte JSON formato:

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

Tipos de erro de consulta agendada

O `Error` objeto pode ser de um dos três tipos:

- Erros de ingestão de registros

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

```
}
```

- Erros de análise e validação de linhas

```
{
    "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.
}
```

- Erros gerais

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

Exemplo de relatórios de erros de consultas agendadas

Veja a seguir um exemplo de um relatório de erro produzido devido a erros de ingestão.

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

Padrões e exemplos de consultas agendadas

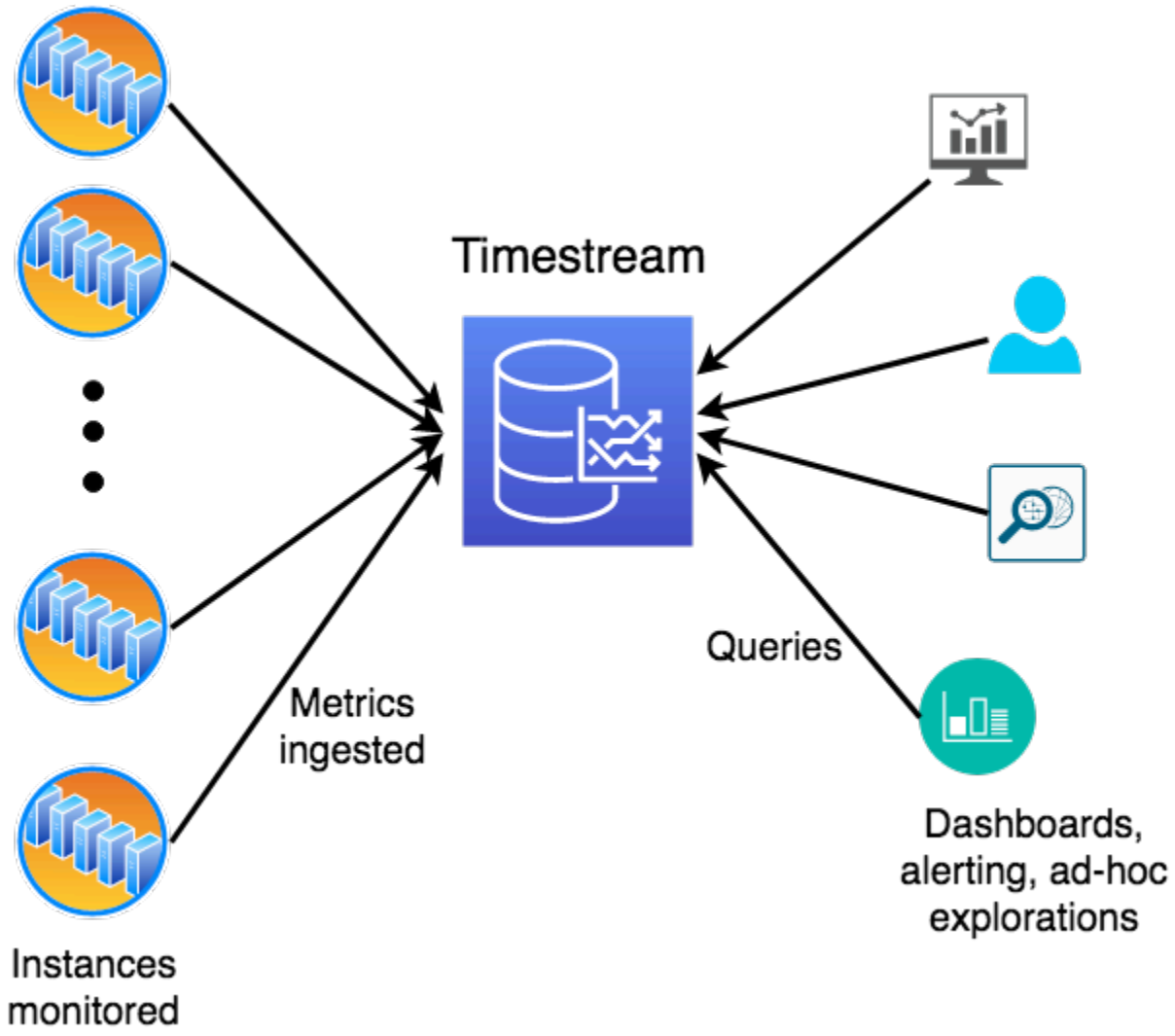
Esta seção descreve os padrões de uso para consultas agendadas, bem como end-to-end exemplos.

Tópicos

- Exemplo de esquema de consultas agendadas
- Padrões de consulta agendados
- Exemplos de consultas agendadas

Exemplo de esquema de consultas agendadas

Neste exemplo, usaremos um aplicativo de amostra que imita um DevOps cenário de monitoramento de métricas de uma grande frota de servidores. Os usuários querem alertar sobre o uso anômalo de recursos, criar painéis sobre o comportamento e a utilização agregados da frota e realizar análises sofisticadas em dados recentes e históricos para encontrar correlações. O diagrama a seguir ilustra a configuração em que um conjunto de instâncias monitoradas emite métricas para o Timestream. LiveAnalytics Outro conjunto de usuários simultâneos emite consultas para alertas, painéis ou análises ad hoc, em que as consultas e a ingestão são executadas paralelamente.



O aplicativo que está sendo monitorado é modelado como um serviço altamente escalável que é implantado em várias regiões do mundo. Cada região é subdividida em várias unidades de escalonamento chamadas células que têm um nível de isolamento em termos de infraestrutura dentro da região. Cada célula é subdividida em silos, que representam um nível de isolamento

do software. Cada silo tem cinco microsserviços que compõem uma instância isolada do serviço. Cada microsserviço tem vários servidores com diferentes tipos de instância e versões de sistema operacional, que são implantados em três zonas de disponibilidade. Esses atributos que identificam os servidores que emitem as métricas são modelados como dimensões no Timestream for. LiveAnalytics Nessa arquitetura, temos uma hierarquia de dimensões (como região, célula, silo e microservice_name) e outras dimensões que atravessam a hierarquia (como instance_type e availability_zone).

O aplicativo emite uma variedade de métricas (como cpu_user e memory_free) e eventos (como task_completed e gc_reclaimed). Cada métrica ou evento está associado a oito dimensões (como região ou célula) que identificam de forma exclusiva o servidor que o emite. Os dados são gravados com as 20 métricas armazenadas juntas em um registro de várias medidas com métricas de nome de medida e todos os 5 eventos são armazenados juntos em outro registro de várias medidas com eventos de nome de medida. O modelo de dados, o esquema e a geração de dados podem ser encontrados no gerador de dados de código aberto. Além do esquema e das distribuições de dados, o gerador de dados fornece um exemplo do uso de vários gravadores para ingerir dados em paralelo, usando a escala de ingestão do Timestream para ingerir milhões de medições por segundo LiveAnalytics . Abaixo, mostramos o esquema (esquema de tabela e medida) e alguns dados de amostra do conjunto de dados.

Tópicos

- Registros de várias medidas
- Registros de medida única

Registros de várias medidas

Esquema da tabela

Abaixo está o esquema da tabela quando os dados são ingeridos usando registros de várias medidas. É a saída da DESCRIBE consulta. Supondo que os dados sejam ingeridos em um banco de dados raw_data e na tabela devops, abaixo está a consulta.

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

Coluna	Tipo	Fluxo de tempo para o tipo de LiveAnalytics atributo
availability_zone	varchar	DIMENSION
nome_do_microsserviço	varchar	DIMENSION
nome_instância	varchar	DIMENSION
nome_de_processo	varchar	DIMENSION
os_version	varchar	DIMENSION
versão_jdk	varchar	DIMENSION
célula	varchar	DIMENSION
região	varchar	DIMENSION
silo	varchar	DIMENSION
instance_type	varchar	DIMENSION
nome_medida	varchar	MEASURE_NAME
horário	timestamp	TIMESTAMP
sem memória	double	MULTI
roubar CPU	double	MULTI
cpu_iowait	double	MULTI
usuário_CPU	double	MULTI
memória armazenada em cache	double	MULTI
disk_io_reads	double	MULTI
cpu_hi	double	MULTI

Coluna	Tipo	Fluxo de tempo para o tipo de LiveAnalytics atributo
latência por leitura	double	MULTI
saída de bytes de rede	double	MULTI
cpu_idle	double	MULTI
disk_free	double	MULTI
memória_usada	double	MULTI
sistema_cpu	double	MULTI
descritores de arquivo em uso	double	MULTI
disk_used	double	MULTI
cpu_nice	double	MULTI
disk_io_writes	double	MULTI
cpu_si	double	MULTI
latência por gravação	double	MULTI
network_bytes_in	double	MULTI
estado_fim_final da tarefa	varchar	MULTI
gc_pause	double	MULTI
tarefa_concluída	bigint	MULTI
gc_recuperado	double	MULTI

Esquema de medidas

Abaixo está o esquema de medidas retornado pela SHOW MEASURES consulta.

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

nome_medida	data_type	Dimensões
eventos	multi	[{"data_type” :"varchar ”, "nome_da-dimensão” :"zona_disponibilidade "}, {" data_type ” :"varchar”, "nome_da-dimensão” :"nome_do _microserviço "}, {" data_type” :"varchar”, "nome_dimensão” :"nome_instância "}, {" data_type ” :"varchar”, "nome_da-dimensão” :"nome_do _processo "}, {" data_type ” :"varchar”, "nome_da-dimensão” :"jdk_version "}, {" data_type” :"varchar”, "nome_dimensão” :"célula "}, {" data_type” :"varchar”, "nome_dimensão” :"região "}, {" data_type” :"data_região "}, {" data_type” :"data_região "} type” :"varchar”, "nome_da_ dimensão” :"silo "}]
métricas	multi	[{"data_type” :"varchar ”, "nome_da-dimensão” :"zona_disponibilidade "}, {" data_type ” :"varchar”, "nome_da-dimensão” :"nome_do _microserviço "}, {" data_type” :"varchar”,

nome_medida	data_type	Dimensões
		"nome_dimensão” :"nome_instância "}, {" data_type ” :"varchar”, "nome_da-dimensão” :"os_versão_"}, {"data_type” :"varchar”, "nome_dimensão” :"célula "}, {" data_type” :"varchar”, "nome_dimensão” :"região "}, {" data_type” :"varchar ”, "nome_dimensão” :"silo "}, {" data_type” "varchar”, "nome_dimensão” :"tipo_instância "}]

Dados de exemplo

re	C	G	a	n	n	in	o	n	v	n	T	u	si	c	ro	c	c	c	c	m	m	d	la	di	la	d	d	n	sa	d	s	e	g	ta	gerecu
												P	p		C	t					s	a	e	p	ri	p			y	s	e	m				rádo
us	us	us	le	a	i-	m	A			m	1	6	0	3	0	0	0	0	0	5	8	5	9	3	2	6	2	8	4	3	5				
e	e-	-	d		za	e					1																									
	c	c	e		-						1																									
		s			-2																															
		il			a																															
					u																															
					-																															
					e																															
					c																															
					-																															
					si																															

re	C	G	a	n	n	in	o	n	v	n	T	u	si	c	ro	c	c	c	c	m	m	d	la	d	la	d	d	n	sa	d	d	s	e	g	ta	g	ge
			ity	ic	â	ty	n	ro	k	d		P	p		C	t					sa	a	e	le	ri	p			yt	d	e	m			n	rá
				iç	0																	a			g					e		u	da			
					m																	e															
					c																	c															
u	u	u	le	a	i-	m	A			m	1	5	0	3	0	0	0	0	0	9	3	5	3	2	9	5	3	7	5	6	2					
	e	e	d		z	e					1																										
	-	-	e		-						1																										
	c	c			2																																
		s			a																																
		il			u																																
					-																																
					e																																
					c																																
					-																																
					si																																
					0																																
					m																																
					c																																

re	C	G	a	n	n	in	o	n	v	n	T	u	si	c	ro	c	c	c	c	m	m	d	la	d	la	d	d	n	sa	d	s	e	g	ta	g	ge
			ity	ic	â	ty	on	ro	k	d		P	p		C	t					sa	a	e	p	g			yt	d	b	e	m				
				iç																		a	e	le						d	d	a				
																						ca								re	e	u				
u	u	u	le	a	i-	m	A			m	1	4	0	4	0	0	0	0	0	9	4	5	3	8	3	7	6	8	5	4	8				
e	e	e	d		za	e					1																									
	-	-	e		-						1																									
	co	co			-2						1																									
		sil			au																															
					-																															
					e																															
					co																															
					-																															
					si																															
					0																															
					m																															
					co																															
u	u	u	le	a	i-	m	A			m	1	3	0	5	0	0	0	0	0	4	7	2	4	4	3	8	6	2	1	2	1				
e	e	e	d		za	e					1																									
	-	-	e		-						1																									
	co	co			-2						1																									
		sil			au																															
					-																															
					e																															
					co																															
					-																															
					si																															
					0																															
					m																															
					co																															

re	C	G	a	n	n	in	o	n	v	n	T	u	si	c	ro	c	c	c	c	m	m	d	la	d	la	d	d	n	s	d	s	e	g	ta	g	_ec	up
			ity	ic	â	ty	n	ro	k	d		P	p		C	t					sa	a	e	p	ri	p			yt	d	e		m			ráda	
				iç																		ea		le		g				b	e	u		da				
u	u	u	le	a	i-	m	A			m	1	5	0	3	0	0	0	0	0	5	3	8	5	7	1	8	6	7	6	5	3					
	e	e	d		z	e					1																										
	-	-	e		-						1																										
	c	c			2																																
		s			a																																
		il			u																																
					-																																
					e																																
					c																																
					-																																
					si																																
					0																																
					m																																
					c																																
u	u	u	le	a	i-			g	J	e	1																			7	S	8	3	6	3	4.8
e	e	e	d		z			o			1																				W					
	-	-	e		-			ia			1																				_					
	c	c			2																										R					
		s			a																															
		il			u																															
					-																															
					e																															
					c																															
					-																															
					si																															
					0																															
					m																															
					c																															

re	C	G	a ity	n ic iç	n â	ir ty	o n	n rc	v k	n d	T	u P	si p	c	rc C	c t	c	c	c	m sa	m a a e ca	d e	la p le	d ri	la p g	d	d	n yt	sa d b d re	d e d a e u	sc m	e m d ta	g	ta n	gerecup rada
u e	u e - c	u e - c s il	le d e	a	i- z - -2 a u - e c - si 0 m c			g o iã	Jl	e	1 1 1.																				7	S IT T	2	4	7345W JL
u e	u e - c	u e - c s il	le d e	a	i- z - -2 a u - e c - si 0 m c			g o iã	Jl	e	1 1 1.																				6	S IT T	6	2	723W JL

re	C	G	a ity	n ic iç	n â	ir ty	o n	n rc	v k	n d	T	u P	si p	c	rc C	c t	c	c	c	m s	m a a e c	d e	la p le	d ri	la p g	d	d	n y	s d b d re	d e d a e u	s m	e m d ta	g	ta n	ge rec up ráda
u e	u e - c	u e - c s il	le d e	a	i- z - -2 a u - e c - si 0 m c			g o iã	Jl	e	1 1 1.																				2	S IT T	5	1	9 9_W JL
u e	u e - c	u e - c s il	le d e	a	i- z - -2 a u - e c - si 0 m c			g o iã	Jl	e	1 1 1.																				3	S W _ R	7	2	8 3_9

Registros de medida única

O Timestream for LiveAnalytics também permite que você consuma os dados com uma medida por registro de série temporal. Abaixo estão os detalhes do esquema quando ingerido usando registros de medida única.

Esquema da tabela

Abaixo está o esquema da tabela quando os dados são ingeridos usando registros de várias medidas. É a saída da DESCRIBE consulta. Supondo que os dados sejam ingeridos em um banco de dados raw_data e na tabela devops, abaixo está a consulta.

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

Coluna	Tipo	Fluxo de tempo para o tipo de LiveAnalytics atributo
availability_zone	varchar	DIMENSION
nome_do_microsserviço	varchar	DIMENSION
nome_instância	varchar	DIMENSION
nome_de_processo	varchar	DIMENSION
os_version	varchar	DIMENSION
versão_jdk	varchar	DIMENSION
célula	varchar	DIMENSION
região	varchar	DIMENSION
silo	varchar	DIMENSION
instance_type	varchar	DIMENSION
nome_medida	varchar	MEASURE_NAME
horário	timestamp	TIMESTAMP

Coluna	Tipo	Fluxo de tempo para o tipo de LiveAnalytics atributo
valor_medida::duplo	double	MEASURE_VALUE
valor_medida::bigint	bigint	MEASURE_VALUE
valor_medida::varchar	varchar	MEASURE_VALUE

Esquema de medidas

Abaixo está o esquema de medidas retornado pela SHOW MEASURES consulta.

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

nome_medida	data_type	Dimensões
cpu_hi	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'os_versi on', 'data_type': 'varchar'}, {'dimension_name': 'célula', 'data_type': 'varchar'}, {'dimension_name': 'região', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'tipo de instância', 'tipo de dados': 'varchar'}]

nome_medida	data_type	Dimensões
cpu_idle	double	[{'dimension_name': 'availability_zone', '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': 'célula', 'data_type': 'varchar'}, {'dimension_name': 'região', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'tipo de instância', 'tipo de dados': 'varchar'}]

nome_medida	data_type	Dimensões
cpu_iowait	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'os_versi on', 'data_type': 'varchar'}, {'dimension_name': 'célula', 'data_type': 'varchar'}, {'dimension_name': 'região', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'tipo de instância', 'tipo de dados': 'varchar'}]

nome_medida	data_type	Dimensões
cpu_nice	double	[{'dimension_name': 'availability_zone', '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': 'célula', 'data_type': 'varchar'}, {'dimension_name': 'região', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'tipo de instância', 'tipo de dados': 'varchar'}]

nome_medida	data_type	Dimensões
cpu_si	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'os_versi on', 'data_type': 'varchar'}, {'dimension_name': 'célula', 'data_type': 'varchar'}, {'dimension_name': 'região', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'tipo de instância', 'tipo de dados': 'varchar'}]

nome_medida	data_type	Dimensões
roubar CPU	double	[{'dimension_name': 'availability_zone', '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': 'célula', 'data_type': 'varchar'}, {'dimension_name': 'região', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'tipo de instância', 'tipo de dados': 'varchar'}]

nome_medida	data_type	Dimensões
sistema_cpu	double	[{'dimension_name': 'availability_zone', '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': 'célula', 'data_type': 'varchar'}, {'dimension_name': 'região', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'tipo de instância', 'tipo de dados': 'varchar'}]

nome_medida	data_type	Dimensões
usuário_CPU	double	[{'dimension_name': 'availability_zone', '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': 'célula', 'data_type': 'varchar'}, {'dimension_name': 'região', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'tipo de instância', 'tipo de dados': 'varchar'}]

nome_medida	data_type	Dimensões
disk_free	double	[{'dimension_name': 'availability_zone', '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': 'célula', 'data_type': 'varchar'}, {'dimension_name': 'região', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'tipo de instância', 'tipo de dados': 'varchar'}]

nome_medida	data_type	Dimensões
disk_io_reads	double	[{'dimension_name': 'availability_zone', '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': 'célula', 'data_type': 'varchar'}, {'dimension_name': 'região', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'tipo de instância', 'tipo de dados': 'varchar'}]

nome_medida	data_type	Dimensões
disk_io_writes	double	[{'dimension_name': 'availability_zone', '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': 'célula', 'data_type': 'varchar'}, {'dimension_name': 'região', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'tipo de instância', 'tipo de dados': 'varchar'}]

nome_medida	data_type	Dimensões
disk_used	double	[{'dimension_name': 'availability_zone', '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': 'célula', 'data_type': 'varchar'}, {'dimension_name': 'região', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'tipo de instância', 'tipo de dados': 'varchar'}]

nome_medida	data_type	Dimensões
descritores de arquivo em uso	double	[{'dimension_name': 'availability_zone', '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': 'célula', 'data_type': 'varchar'}, {'dimension_name': 'região', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'tipo de instância', 'tipo de dados': 'varchar'}]

nome_medida	data_type	Dimensões
gc_pause	double	[{'dimension_name': 'availability_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'nome do processo', 'data_type': 'varchar'}, {'dimension_name': 'jdk_version', 'data_type': 'varchar'}, {'dimension_name': 'célula', 'data_type': 'varchar'}, {'dimension_name': 'região', 'data_type': 'varchar'}, {'dimension_name': 'varchar'}, {'dimension_name': 'data_type': 'varchar'}, {'dimension_name': 'varchar'}, {'dimension_name': 'data_type': 'varchar'}, {'dimension_name': 'varchar'}, {'dimension_name': 'data_type': 'varchar' nome ':' silo ',' data_type ':' varchar '}]

nome_medida	data_type	Dimensões
gc_recuperado	double	[{'dimension_name': 'availability_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'nome do processo', 'data_type': 'varchar'}, {'dimension_name': 'jdk_version', 'data_type': 'varchar'}, {'dimension_name': 'célula', 'data_type': 'varchar'}, {'dimension_name': 'região', 'data_type': 'varchar'}, {'dimension_name': 'varchar'}, {'dimension_name': 'data_type': 'varchar'}, {'dimension_name': 'varchar'}, {'dimension_name': 'data_type': 'varchar'}, {'dimension_name': 'varchar'}, {'dimension_name': 'data_type': 'varchar' nome ':' silo ',' data_type ':' varchar '}]

nome_medida	data_type	Dimensões
latência por leitura	double	[{'dimension_name': 'availability_zone', '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': 'célula', 'data_type': 'varchar'}, {'dimension_name': 'região', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'tipo de instância', 'tipo de dados': 'varchar'}]

nome_medida	data_type	Dimensões
latência por gravação	double	[{'dimension_name': 'availability_zone', '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': 'célula', 'data_type': 'varchar'}, {'dimension_name': 'região', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'tipo de instância', 'tipo de dados': 'varchar'}]

nome_medida	data_type	Dimensões
memória armazenada em cache	double	[{'dimension_name': 'availability_zone', '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': 'célula', 'data_type': 'varchar'}, {'dimension_name': 'região', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'tipo de instância', 'tipo de dados': 'varchar'}]

nome_medida	data_type	Dimensões
sem memória	double	[{'dimension_name': 'availabi lity_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_typ e': 'varchar'}, {'dimensi on_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'nome do processo', 'data_type': 'varchar'}, {'dimension_name': 'os_version', 'data_type': 'varchar'}, {'dimension_name': 'jdk_version', 'data_typ e': 'varchar'}, {'dimensi on_name': 'célula', 'data_typ e': 'varchar'}, {'dimensi on_name': 'região', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'instance _type', 'data_ tipo': 'varchar'}]

nome_medida	data_type	Dimensões
memória_usada	double	[{'dimension_name': 'availability_zone', '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': 'célula', 'data_type': 'varchar'}, {'dimension_name': 'região', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'tipo de instância', 'tipo de dados': 'varchar'}]

nome_medida	data_type	Dimensões
network_bytes_in	double	[{'dimension_name': 'availability_zone', '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': 'célula', 'data_type': 'varchar'}, {'dimension_name': 'região', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'tipo de instância', 'tipo de dados': 'varchar'}]

nome_medida	data_type	Dimensões
saída de bytes de rede	double	[{'dimension_name': 'availability_zone', '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': 'célula', 'data_type': 'varchar'}, {'dimension_name': 'região', 'data_type': 'varchar'}, {'dimension_name': 'silo', 'data_type': 'varchar'}, {'dimension_name': 'tipo de instância', 'tipo de dados': 'varchar'}]

nome_medida	data_type	Dimensões
tarefa_concluída	bigint	[{'dimension_name': 'availability_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'nome do processo', 'data_type': 'varchar'}, {'dimension_name': 'jdk_version', 'data_type': 'varchar'}, {'dimension_name': 'célula', 'data_type': 'varchar'}, {'dimension_name': 'região', 'data_type': 'varchar'}, {'dimension_name': 'varchar'}, {'dimension_name': 'data_type': 'varchar'}, {'dimension_name': 'varchar'}, {'dimension_name': 'data_type': 'varchar'}, {'dimension_name': 'varchar'}, {'dimension_name': 'data_type': 'varchar' nome ':' silo ',' data_type ':' varchar '}]

nome_medida	data_type	Dimensões
estado_fim_final da tarefa	varchar	[{'dimension_name': 'availability_zone', 'data_type': 'varchar'}, {'dimension_name': 'microservice_name', 'data_type': 'varchar'}, {'dimension_name': 'instance_name', 'data_type': 'varchar'}, {'dimension_name': 'nome do processo', 'data_type': 'varchar'}, {'dimension_name': 'jdk_version', 'data_type': 'varchar'}, {'dimension_name': 'célula', 'data_type': 'varchar'}, {'dimension_name': 'região', 'data_type': 'varchar'}, {'dimension_name': 'varchar'}, {'dimension_name': 'data_type': 'varchar'}, {'dimension_name': 'varchar'}, {'dimension_name': 'data_type': 'varchar'}, {'dimension_name': 'varchar'}, {'dimension_name': 'data_type': 'varchar' nome ':' silo ',' data_type ':' varchar '}]

Dados de exemplo

availa ity_zc	nome icross iço	nome ância	nome roces	os_ve n	versã k	Célul	regiã	Gran	instar type	nome da	Temp	valor_ ida::d o	valor_ ida::b nt	valor_med ida::varc har
eu-west--1	hércu	i-zaZs -		AL20		eu-west-	eu-west-	eu-west-	r5.4xl e	cpu_l	34:57	0,871		

availa ity_zo	nome icross iço	nome ância	nome roces	os_ve n	versã k	Célul	regiã	Gran	instar type	nome da	Temp	valor_ ida::d o	valor_ ida::b nt	valor_med ida::varc har
		hercu eu- west -1- célula -9- silo-2 -000(.ama s.com				- cell-9		cell-9 s ilo-2						
eu- west- -1	hércu	i- zaZsv - hercu eu- west -1- célula -9- silo-2 -000(.ama s.com		AL20		eu- west- - cell-9	eu- west-	eu- west- - cell-9 s ilo-2	r5.4xl e	cpu_i	34:57	3.462		

availa ity_zo	nome icross iço	nome ância	nome roces	os_ve n	versã k	Célul	regiã	Gran	instar type	nome da	Temp	valor_ ida::d o	valor_ ida::b nt	valor_med ida::varc har
eu- west- -1	hércu	i- zaZsv - hercu eu- west -1- célula -9- silo-2 -000(.ama s.com		AL20		eu- west- - cell-9	eu- west-	eu- west- - cell-9 s ilo-2	r5.4xl e	cpu_i t	34:57	0,102		
eu- west- -1	hércu	i- zaZsv - hercu eu- west -1- célula -9- silo-2 -000(.ama s.com		AL20		eu- west- - cell-9	eu- west-	eu- west- - cell-9 s ilo-2	r5.4xl e	cpu_r	34:57	0,630		

availa ity_zo	nome icross iço	nome ância	nome roces	os_ve n	versã k	Célul	regiã	Gran	instar type	nome da	Temp	valor_ ida::d o	valor_ ida::b nt	valor_med ida::varc har
eu- west- -1	hércu	i- zaZsv - hercu eu- west -1- célula -9- silo-2 -000(.amaz s.com		AL20		eu- west- - cell-9	eu- west-	eu- west- - cell-9 s ilo-2	r5.4xl e	cpu_s	34:57	0,164		
eu- west- -1	hércu	i- zaZsv - hercu eu- west -1- célula -9- silo-2 -000(.amaz s.com		AL20		eu- west- - cell-9	eu- west-	eu- west- - cell-9 s ilo-2	r5.4xl e	rouba CPU	34:57	0,107		

availa ity_zo	nome icross iço	nome ância	nome roces	os_ve n	versã k	Célul	regiã	Gran	instar type	nome da	Temp	valor_ ida::d o	valor_ ida::b nt	valor_med ida::varc har
eu- west- -1	hércu	i- zaZsv - hercu eu- west -1- célula -9- silo-2 -000(.amaz s.com		AL20		eu- west- - cell-9	eu- west-	eu- west- - cell-9 s ilo-2	r5.4xl e	sister pu	34:57	0,457		
eu- west- -1	hércu	i- zaZsv - hercu eu- west -1- célula -9- silo-2 -000(.amaz s.com		AL20		eu- west- - cell-9	eu- west-	eu- west- - cell-9 s ilo-2	r5.4xl e	usuár PU	34:57	94.20		

availa ity_zo	nome icross iço	nome ância	nome roces	os_ve n	versã k	Célul	regiã	Gran	instar type	nome da	Temp	valor_ ida::d o	valor_ ida::b nt	valor_med ida::varc har
eu- west- -1	hércu	i- zaZsv - hercu eu- west -1- célula -9- silo-2 -000(.amaz s.com		AL20		eu- west- - cell-9	eu- west-	eu- west- - cell-9 s ilo-2	r5.4xl e	disk_t	34:57	72.51		
eu- west- -1	hércu	i- zaZsv - hercu eu- west -1- célula -9- silo-2 -000(.amaz s.com		AL20		eu- west- - cell-9	eu- west-	eu- west- - cell-9 s ilo-2	r5.4xl e	disk_i eads	34:57	81.73		

availa ity_zo	nome icross iço	nome ância	nome roces	os_ve n	versã k	Célul	regiã	Gran	instar type	nome da	Temp	valor_ ida::d o	valor_ ida::b nt	valor_med ida::varc har
eu- west- -1	hércu	i- zaZsv - hercu eu- west -1- célula -9- silo-2 -000(.amaz s.com		AL20		eu- west- - cell-9	eu- west-	eu- west- - cell-9 s ilo-2	r5.4xl e	disk_i rites	34:57	77.16		
eu- west- -1	hércu	i- zaZsv - hercu eu- west -1- célula -9- silo-2 -000(.amaz s.com		AL20		eu- west- - cell-9	eu- west-	eu- west- - cell-9 s ilo-2	r5.4xl e	disk_	34:57	89.42		

availa ity_zo	nome icross iço	nome ância	nome roces	os_ve n	versã k	Célul	regiã	Gran	instar type	nome da	Temp	valor_ ida::d o	valor_ ida::b nt	valor_med ida::varc har
eu- west- -1	hércu	i- zaZsv - hercu eu- west -1- célula -9- silo-2 -000(.amaz s.com		AL20		eu- west- - cell-9	eu- west-	eu- west- - cell-9 s ilo-2	r5.4xl e	descr es de arquiv em uso	34:57	30.08		
eu- west- -1	hércu	i- zaZsv - hercu eu- west -1- célula -9- silo-2 -000(.amaz s.com	servic		JDK_	eu- west- - cell-9	eu- west-	eu- west- - cell-9 s ilo-2		gc_pa	34:57	60.28		

availa ity_zo	nome icross iço	nome ância	nome roces	os_ve n	versã k	Célul	regiã	Gran	instar type	nome da	Temp	valor_ ida::d o	valor_ ida::b nt	valor_med ida::varc har
eu- west- -1	hércu	i- zaZsv - hercu eu- west -1- célula -9- silo-2 -000(.amaz s.com	servic		JDK_	eu- west- - cell-9	eu- west-	eu- west- - cell-9 s ilo-2		gc_re rado	34:57	75.28		
eu- west- -1	hércu	i- zaZsv - hercu eu- west -1- célula -9- silo-2 -000(.amaz s.com		AL20		eu- west- - cell-9	eu- west-	eu- west- - cell-9 s ilo-2	r5.4xl e	latênc por leitura	34:57	8.076		

availa ity_zo	nome icross iço	nome ância	nome roces	os_ve n	versã k	Célul	regiã	Gran	instar type	nome da	Temp	valor_ ida::d o	valor_ ida::b nt	valor_med ida::varc har
eu- west- -1	hércu	i- zaZsv - hercu eu- west -1- célula -9- silo-2 -000(.amaz s.com		AL20		eu- west- - cell-9	eu- west-	eu- west- - cell-9 s ilo-2	r5.4xl e	latênc por grava	34:57	58.12		
eu- west- -1	hércu	i- zaZsv - hercu eu- west -1- célula -9- silo-2 -000(.amaz s.com		AL20		eu- west- - cell-9	eu- west-	eu- west- - cell-9 s ilo-2	r5.4xl e	memo arma a em cache	34:57	87.56		

availa ity_zo	nome icross iço	nome ância	nome roces	os_ve n	versã k	Célul	regiã	Gran	instar type	nome da	Temp	valor_ ida::d o	valor_ ida::b nt	valor_med ida::varc har
eu- west- -1	hércu	i- zaZsv - hercu eu- west -1- célula -9- silo-2 -000(.amaz s.com	servic		JDK_	eu- west- - cell-9	eu- west-	eu- west- - cell-9 s ilo-2		sem memo	34:57	18.95		
eu- west- -1	hércu	i- zaZsv - hercu eu- west -1- célula -9- silo-2 -000(.amaz s.com		AL20		eu- west- - cell-9	eu- west-	eu- west- - cell-9 s ilo-2	r5.4xl e	sem memo	34:57	97.20		

availa ity_zo	nome icross iço	nome ância	nome roces	os_ve n	versã k	Célul	regiã	Gran	instar type	nome da	Temp	valor_ ida::d o	valor_ ida::b nt	valor_med ida::varc har
eu- west- -1	hércu	i- zaZsv - hercu eu- west -1- célula -9- silo-2 -000(.amaz s.com		AL20		eu- west- - cell-9	eu- west-	eu- west- - cell-9 s ilo-2	r5.4xl e	mem sada	34:57	12.37		
eu- west- -1	hércu	i- zaZsv - hercu eu- west -1- célula -9- silo-2 -000(.amaz s.com		AL20		eu- west- - cell-9	eu- west-	eu- west- - cell-9 s ilo-2	r5.4xl e	netwo ytes_	34:57	31.02		

availa ity_zo	nome icross iço	nome ância	nome roces	os_ve n	versã k	Célul	regiã	Gran	instar type	nome da	Temp	valor_ ida::d o	valor_ ida::b nt	valor_med ida::varc har
eu- west- -1	hércu	i- zaZsv - hercu eu- west -1- célula -9- silo-2 -000(.amaz s.com		AL20		eu- west- - cell-9	eu- west-	eu- west- - cell-9 s ilo-2	r5.4xl e	saída de bytes de rede	34:57	0,514		
eu- west- -1	hércu	i- zaZsv - hercu eu- west -1- célula -9- silo-2 -000(.amaz s.com	servic		JDK_	eu- west- - cell-9	eu- west-	eu- west- - cell-9 s ilo-2		tarefa ncluíc	34:57		69	

availa ity_zo	nome icross iço	nome ância	nome roces	os_ve n	versã k	Célul	regiã	Gran	instar type	nome da	Temp	valor_ ida::d o	valor_ ida::b nt	valor_med ida::varc har
eu-west--1	hércu	i-zaZsv - hercu eu-west -1- célula -9- silo-2 -000(.amaz s.com	servic		JDK_	eu-west- - cell-9	eu-west-	eu-west- - cell-9 s ilo-2		estad m_fin da tarefa	34:57			SUCCESS_ ITH_RESUL T

Padrões de consulta agendados

Nesta seção, você encontrará alguns padrões comuns de como você pode usar o Amazon Timestream LiveAnalytics para consultas programadas para otimizar seus painéis para que sejam carregados mais rapidamente e com custos reduzidos. Os exemplos abaixo usam um cenário de DevOps aplicativo para ilustrar os principais conceitos que se aplicam às consultas agendadas em geral, independentemente do cenário do aplicativo.

As consultas agendadas no Timestream for LiveAnalytics permitem que você expresse suas consultas usando toda a área de SQL superfície do Timestream for. LiveAnalytics Sua consulta pode incluir uma ou mais tabelas de origem, realizar agregações ou qualquer outra consulta permitida pela SQL linguagem LiveAnalytics do Timestream for e, em seguida, materializar os resultados da consulta em outra tabela de destino no Timestream for. LiveAnalytics Para facilitar a exposição, esta seção se refere a essa tabela de destino de uma consulta agendada como uma tabela derivada.

A seguir estão os principais pontos abordados nesta seção.

- Usando um agregado simples em nível de frota para explicar como você pode definir uma consulta agendada e entender alguns conceitos básicos.

- Como você pode combinar os resultados do destino de uma consulta agendada (a tabela derivada) com os resultados da tabela de origem para obter os benefícios de custo e desempenho da consulta agendada.
- Quais são suas vantagens e desvantagens ao configurar o período de atualização das consultas agendadas.
- Usando consultas agendadas para alguns cenários comuns.
  - Rastreando o último ponto de dados de cada instância antes de uma data específica.
  - Valores distintos para uma dimensão a serem usados para preencher variáveis em um painel.
- Como você lida com dados que chegam tardiamente no contexto de consultas agendadas.
- Como você pode usar execuções manuais únicas para lidar com uma variedade de cenários não cobertos diretamente por acionadores automatizados para consultas agendadas.

Tópicos

- Cenário
- Agregados simples em nível de frota
- Último ponto de cada dispositivo
- Valores de dimensão exclusivos
- Lidando com dados que chegam tardiamente
- Preenchendo pré-cálculos históricos

Cenário

Os exemplos a seguir usam um cenário de DevOps monitoramento descrito emExemplo de esquema de consultas agendadas.

Os exemplos fornecem a definição de consulta agendada, na qual você pode inserir as configurações apropriadas de onde receber notificações de status de execução para consultas agendadas, onde receber relatórios de erros encontrados durante a execução de uma consulta agendada e a IAM função que a consulta agendada usa para realizar suas operações.

Você pode criar essas consultas agendadas após preencher as opções anteriores, criar a tabela de destino (ou derivada) e executar a por meio do. AWS CLI Por exemplo, suponha que uma definição de consulta agendada esteja armazenada em um arquivo,`scheduled_query_example.json`. Você pode criar a consulta usando o CLI comando.

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

No comando anterior, o perfil passado usando a opção --profile deve ter as permissões apropriadas para criar consultas agendadas. Consulte Políticas baseadas em identidade para consultas agendadas para obter instruções detalhadas sobre as políticas e permissões.

Agregados simples em nível de frota

Este primeiro exemplo mostra alguns dos conceitos básicos ao trabalhar com consultas agendadas usando um exemplo simples de computação de agregados em nível de frota. Usando esse exemplo, você aprenderá o seguinte.

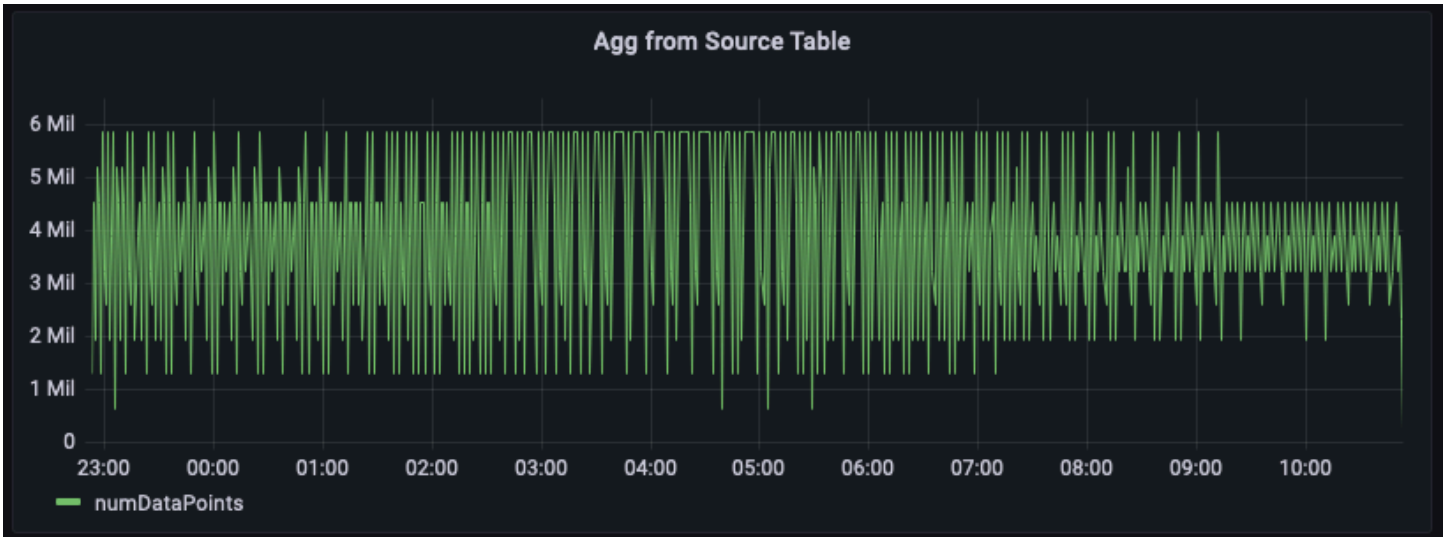
- Como fazer a consulta do painel usada para obter estatísticas agregadas e mapeá-la para uma consulta agendada.
- Como o Timestream for LiveAnalytics gerencia a execução das diferentes instâncias de sua consulta agendada.
- Como você pode fazer com que diferentes instâncias de consultas agendadas se sobreponham em intervalos de tempo e como a exatidão dos dados é mantida na tabela de destino para garantir que seu painel usando os resultados da consulta agendada forneça resultados que correspondam ao mesmo agregado calculado nos dados brutos.
- Como definir o intervalo de tempo e a cadência de atualização para sua consulta agendada.
- Como você pode monitorar por conta própria os resultados das consultas agendadas para ajustá-las de forma que a latência de execução das instâncias de consulta esteja dentro dos prazos aceitáveis de atualização de seus painéis.

Tópicos

- Agregar a partir de tabelas de origem
- Consulta programada para pré-computar agregados
- Agregar da tabela derivada
- Combinação agregada de tabelas de origem e derivadas
- Agregado a partir de cálculos programados frequentemente atualizados

Agregar a partir de tabelas de origem

Neste exemplo, você está rastreando o número de métricas emitidas pelos servidores em uma determinada região em cada minuto. O gráfico abaixo é um exemplo de plotagem dessa série temporal para a região us-east-1.



Abaixo está um exemplo de consulta para calcular esse agregado a partir dos dados brutos. Ele filtra as linhas da região us-east-1 e, em seguida, calcula a soma por minuto contabilizando as 20 métricas (se o nome da medida for métrica) ou 5 eventos (se o nome_medida for eventos). Neste exemplo, a ilustração gráfica mostra que o número de métricas emitidas varia entre 1,5 milhão e 6 milhões por minuto. Ao traçar essa série temporal por várias horas (últimas 12 horas nesta figura), essa consulta sobre os dados brutos analisa centenas de milhões de linhas.

```
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
```

Consulta programada para pré-computar agregados

Se você quiser otimizar seus painéis para carregar mais rápido e reduzir seus custos digitalizando menos dados, você pode usar uma consulta agendada para pré-computar esses agregados. As consultas agendadas no Timestream for LiveAnalytics permitem que você materialize esses pré-cálculos em outro Timestream for LiveAnalytics table, que você pode usar posteriormente em seus painéis.

A primeira etapa na criação de uma consulta agendada é identificar a consulta que você deseja pré-computar. Observe que o painel anterior foi desenhado para a região us-east-1. No entanto, um usuário diferente pode querer o mesmo agregado para uma região diferente, por exemplo, us-west-2 ou eu-west-1. Para evitar a criação de uma consulta agendada para cada consulta, você pode pré-calcular o agregado para cada região e materializar os agregados por região em outro Timestream para tabela. LiveAnalytics

A consulta abaixo fornece um exemplo da pré-computação correspondente. Como você pode ver, é semelhante à expressão de tabela comum grouped_data usada na consulta dos dados brutos, exceto por duas diferenças: 1) ela não usa um predicado de região, de modo que podemos usar uma consulta para pré-computar para todas as regiões; e 2) ela usa um predicado de tempo parametrizado com um parâmetro especial @scheduled_runtime, que é explicado em detalhes abaixo.

```
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
```

A consulta anterior pode ser convertida em uma consulta agendada usando a especificação a seguir. A consulta agendada recebe um Nome, que é um mnemônico fácil de usar. Em seguida, inclui o QueryString, a ScheduleConfiguration, que é uma expressão cron. Ele especifica o TargetConfiguration que mapeia os resultados da consulta para a tabela de destino no Timestream for. LiveAnalytics Por fim, ele especifica várias outras configurações, como a NotificationConfiguration, em que as notificações são enviadas para execuções individuais da consulta, ErrorReportConfiguration em que um relatório é escrito caso a consulta encontre algum erro e a ScheduledQueryExecutionRoleArn, que é a função usada para realizar operações para a consulta agendada.

```
{
```

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

```
}
```

No exemplo, o ScheduleExpression cron (0/5 * * *? *) implica que a consulta seja executada uma vez a cada 5 minutos nos dias 5, 10, 15,.. minutos de cada hora de cada dia. Esses carimbos de data/hora quando uma instância específica dessa consulta é acionada são o que se traduz no parâmetro @scheduled_runtime usado na consulta. Por exemplo, considere a instância dessa consulta agendada em execução em 01/12/2021 00:00:00. Nesse caso, o parâmetro @scheduled_runtime é inicializado com o timestamp 2021-12-01 00:00:00 ao invocar a consulta. Portanto, essa instância específica será executada no timestamp 2021-12-01 00:00:00 e calculará os agregados por minuto do intervalo de tempo 2021-11-30 23:50:00 a 2021-12-01 00:01:00. Da mesma forma, a próxima instância dessa consulta é acionada no timestamp 2021-12-01 00:05:00 e, nesse caso, a consulta computará agregados por minuto a partir do intervalo de tempo 2021-11-30 23:55:00 a 2021-12-01 00:06:00. Portanto, o parâmetro @scheduled_runtime fornece uma consulta agendada para pré-calcular os agregados para os intervalos de tempo configurados usando o tempo de invocação das consultas.

Observe que duas instâncias subsequentes da consulta se sobrepõem em seus intervalos de tempo. Isso é algo que você pode controlar com base em seus requisitos. Nesse caso, essa sobreposição permite que essas consultas atualizem os agregados com base em qualquer dado cuja chegada tenha sido um pouco atrasada, até 5 minutos neste exemplo. Para garantir a exatidão das consultas materializadas, o Timestream for LiveAnalytics garante que a consulta em 01/12/2021 00:05:00 seja executada somente após a conclusão da consulta em 01/12/2021 00:00:00 e que os resultados das últimas consultas possam atualizar qualquer agregado materializado anteriormente usando se um valor mais novo for gerado. Por exemplo, se alguns dados no timestamp 2021-11-30 23:59:00 chegarem após a consulta de 2021-12-01 00:00:00 ser executada, mas antes da consulta de 2021-12-01 00:05:00, a execução em 2021-12-01 00:05:00 recalculará os agregados do minuto 2021-11-30 23:59:00 e isso resultará na atualização do agregado anterior com o valor recém-calculado. Você pode confiar nessa semântica das consultas agendadas para estabelecer uma compensação entre a rapidez com que você atualiza seus pré-cálculos e a forma como você pode lidar com alguns dados com atraso na chegada. Considerações adicionais são discutidas abaixo sobre como você troca essa cadência de atualização com a atualização dos dados e como você aborda a atualização dos agregados por dados que chegam ainda mais atrasados ou se sua fonte de computação programada tem valores atualizados que exigiriam que os agregados fossem recalculados.

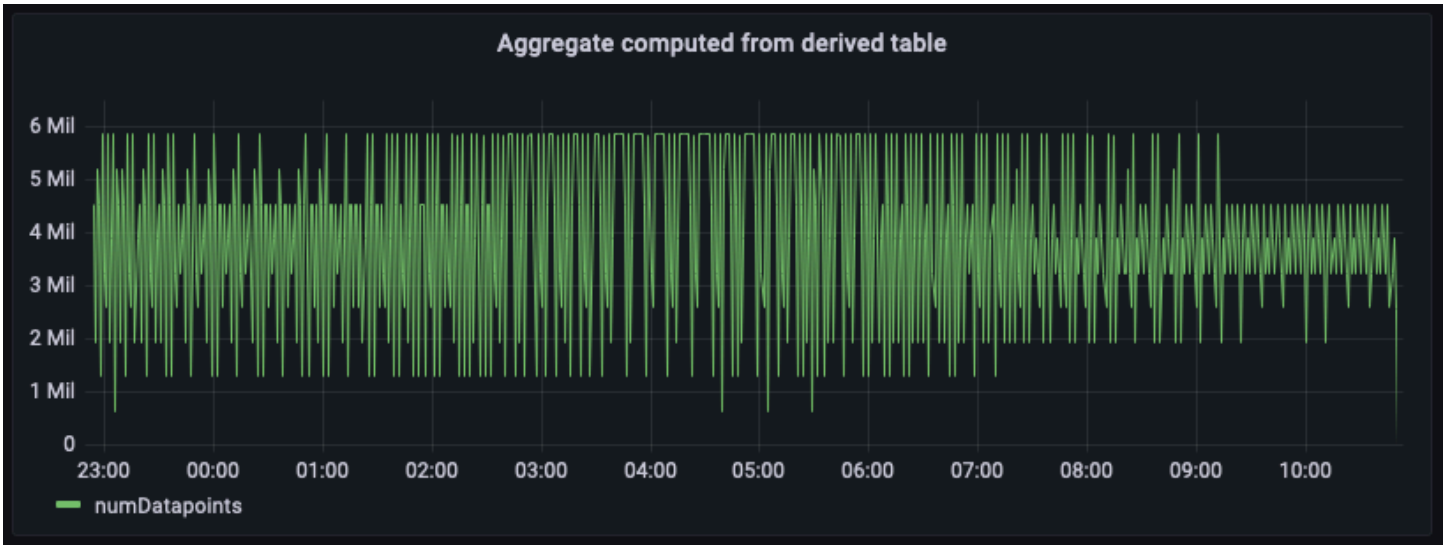
Cada computação programada tem uma configuração de notificação em que o Timestream for LiveAnalytics envia uma notificação de cada execução de uma configuração agendada. Você pode configurar um SNS tópico para receber notificações para cada invocação. Além do status de sucesso

ou falha de uma instância específica, ela também tem várias estatísticas, como o tempo que essa computação levou para ser executada, o número de bytes que a computação verificou e o número de bytes que a computação gravou na tabela de destino. Você pode usar essas estatísticas para ajustar ainda mais sua consulta, agendar a configuração ou monitorar os gastos com suas consultas agendadas. Um aspecto digno de nota é o tempo de execução de uma instância. Neste exemplo, a computação programada é configurada para ser executada a cada 5 minutos. O tempo de execução determinará o atraso com o qual a pré-computação estará disponível, o que também definirá o atraso em seu painel quando você estiver usando os dados pré-computados em seus painéis. Além disso, se esse atraso for consistentemente maior do que o intervalo de atualização, por exemplo, se o tempo de execução for superior a 5 minutos para uma computação configurada para ser atualizada a cada 5 minutos, é importante ajustar sua computação para ser executada mais rapidamente para evitar mais atrasos em seus painéis.

Agregar da tabela derivada

Agora que você configurou as consultas agendadas e os agregados estão pré-computados e materializados em outro Timestream para a LiveAnalytics tabela especificada na configuração de destino da computação agendada, você pode usar os dados dessa tabela para escrever consultas para alimentar seus painéis. SQL Abaixo está um equivalente da consulta que usa os pré-agregados materializados para gerar o agregado de contagem de pontos de dados por minuto para us-east-1.

```
SELECT bin(time, 1m) as minute, SUM(numDataPoints) as numDatapoints
FROM "derived"."per_minute_aggs_pt5m"
WHERE time BETWEEN from_milliseconds(1636699996445) AND
 from_milliseconds(1636743196445)
    AND region = 'us-east-1'
GROUP BY bin(time, 1m)
ORDER BY 1 desc
```



A figura anterior traça o agregado calculado a partir da tabela agregada. Comparando esse painel com o painel calculado a partir dos dados brutos da fonte, você notará que eles coincidem exatamente, embora esses agregados sejam atrasados em alguns minutos, controlados pelo intervalo de atualização que você configurou para o cálculo agendado mais o tempo para executá-lo.

Essa consulta sobre os dados pré-computados digitaliza dados de várias ordens de magnitude menores em comparação com os agregados calculados nos dados de origem bruta. Dependendo da granularidade das agregações, essa redução pode facilmente resultar em 100 vezes menos custo e latência de consulta. A execução dessa computação programada tem um custo. No entanto, dependendo da frequência com que esses painéis são atualizados e de quantos usuários simultâneos os carregam, você acaba reduzindo significativamente seus custos gerais usando esses pré-cálculos. E isso se soma aos tempos de carregamento de 10 a 100 vezes mais rápidos para os painéis.

Combinação agregada de tabelas de origem e derivadas

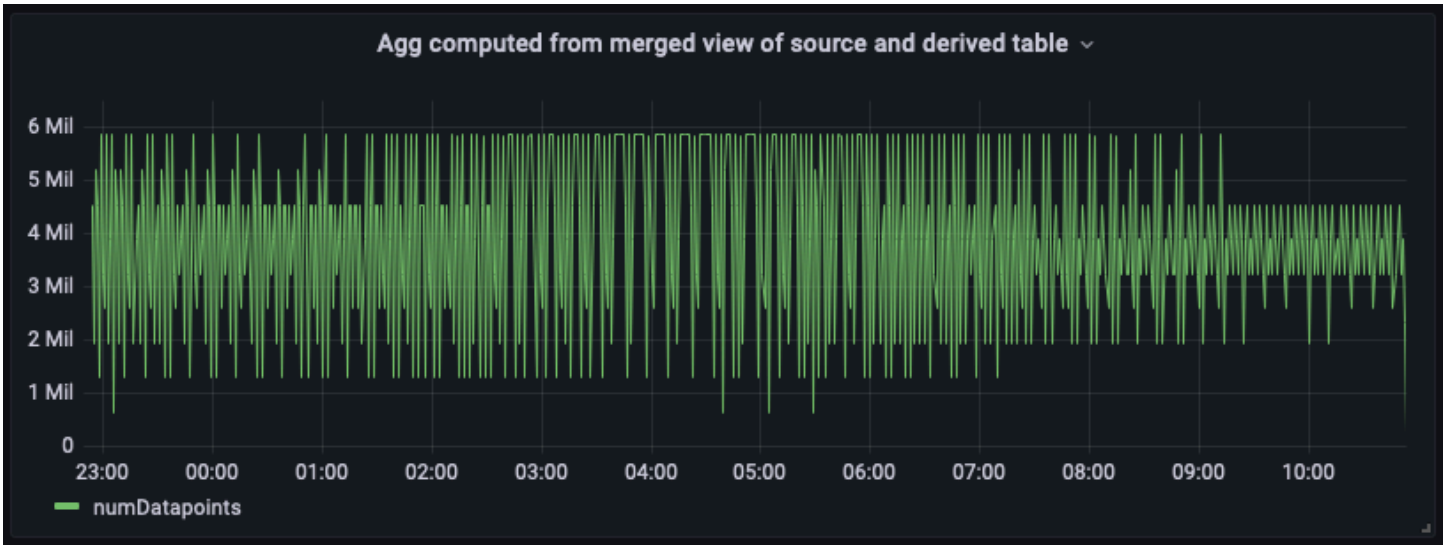
Os painéis criados usando as tabelas derivadas podem ter um atraso. Se o cenário do seu aplicativo exigir que os painéis tenham os dados mais recentes, você poderá usar o poder e a flexibilidade do SQL suporte do Timestream for para LiveAnalytics combinar os dados mais recentes da tabela de origem com os agregados históricos da tabela derivada para formar uma exibição mesclada. Essa exibição mesclada usa a semântica de união SQL e os intervalos de tempo não sobrepostos da tabela de origem e da tabela derivada. No exemplo abaixo, estamos usando o “derivado”.” per_minute_aggs_pt5m” tabela derivada. Como a computação programada para essa tabela derivada é atualizada uma vez a cada 5 minutos (de acordo com a especificação da expressão de agendamento), a consulta abaixo usa os 15 minutos mais recentes de dados da tabela de origem e quaisquer dados anteriores a 15 minutos da tabela derivada e, em seguida, une os resultados para

criar a exibição mesclada que tem o melhor dos dois mundos: a economia e a baixa latência lendo agregados pré-computados mais antigos da tabela derivada e a atualização da agregação da tabela de origem para potencializar seus casos de uso de análise em tempo real.

Observe que essa abordagem de união terá uma latência de consulta um pouco maior em comparação com a consulta apenas da tabela derivada e também terá dados digitalizados um pouco mais altos, pois agrega os dados brutos em tempo real para preencher o intervalo de tempo mais recente. No entanto, essa visualização mesclada ainda será significativamente mais rápida e barata em comparação com a agregação dinâmica da tabela de origem, especialmente para painéis que renderizam dias ou semanas de dados. Você pode ajustar os intervalos de tempo deste exemplo para atender às necessidades de atualização e à tolerância a atrasos do seu aplicativo.

```
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
```

Abaixo está o painel do painel com essa visualização unificada mesclada. Como você pode ver, o painel parece quase idêntico à exibição calculada a partir da tabela derivada, exceto pelo fato de que ele terá a maior quantidade up-to-date agregada na ponta mais à direita.



Agregado a partir de cálculos programados frequentemente atualizados

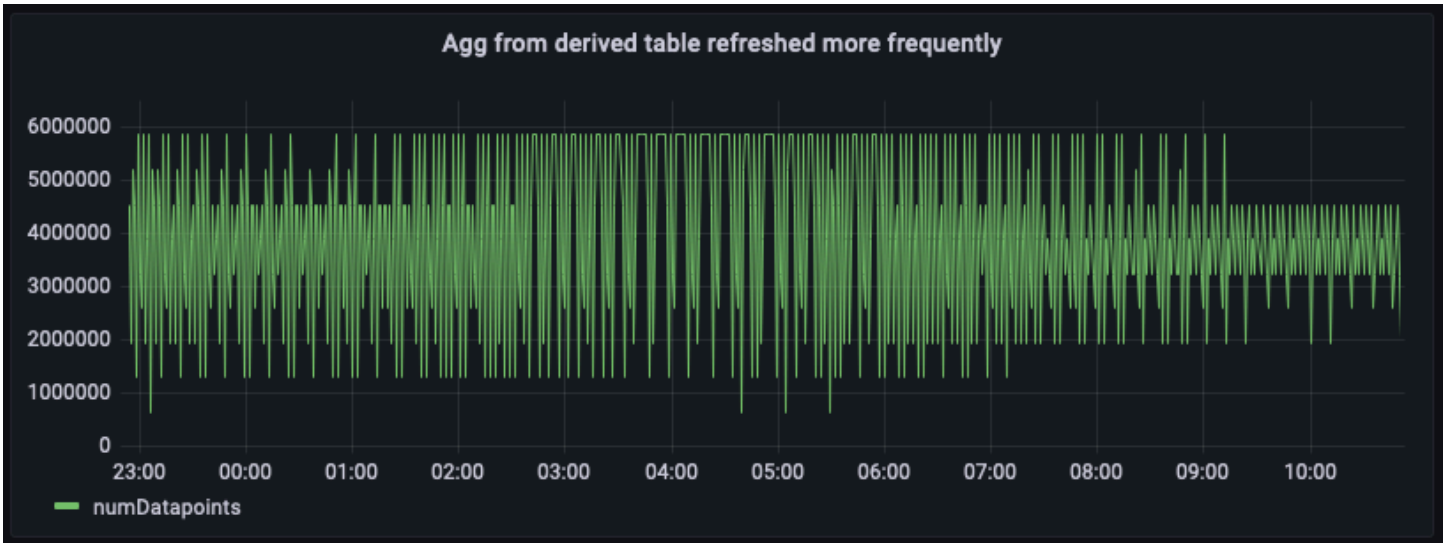
Dependendo da frequência com que seus painéis são carregados e da latência que você deseja para seu painel, há outra abordagem para obter resultados mais recentes em seu painel: fazer com que a computação programada atualize os agregados com mais frequência. Por exemplo, abaixo está a configuração da mesma computação programada, exceto que ela é atualizada uma vez a cada minuto (observe a programação expressa cron (0/1 * * *? *)). Com essa configuração, a tabela derivada per_minute_aggs_pt1m terá agregados muito mais recentes em comparação com o cenário em que o cálculo especificou um cronograma de atualização de uma vez a cada 5 minutos.

```
{
    "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
```

Como a tabela derivada tem agregados mais recentes, agora você pode consultar diretamente a tabela derivada per_minute_aggs_pt1m para obter agregados mais recentes, como pode ser visto na consulta anterior e no instantâneo do painel abaixo.



Observe que atualizar a computação programada em uma programação mais rápida (digamos, 1 minuto em comparação com 5 minutos) aumentará os custos de manutenção da computação programada. A mensagem de notificação para a execução de cada computação fornece estatísticas de quantos dados foram digitalizados e quantos foram gravados na tabela derivada. Da mesma forma, se você usar a exibição mesclada para unir a tabela derivada, você consulta os custos na exibição mesclada e a latência de carregamento do painel será maior em comparação com a consulta somente da tabela derivada. Portanto, a abordagem escolhida dependerá da frequência com que seus painéis são atualizados e dos custos de manutenção das consultas agendadas. Se você tiver dezenas de usuários atualizando os painéis aproximadamente uma vez a cada minuto, ter uma atualização mais frequente da tabela derivada provavelmente resultará em custos gerais mais baixos.

Último ponto de cada dispositivo

Seu aplicativo pode exigir que você leia a última medição emitida por um dispositivo. Pode haver casos de uso mais gerais para obter a última medição de um dispositivo antes de uma determinadadate/time or the first measurement for a device after a given date/time. Quando você tem milhões de dispositivos e anos de dados, essa pesquisa pode exigir a digitalização de grandes quantidades de dados.

Abaixo, você verá um exemplo de como você pode usar consultas agendadas para otimizar a busca pelo último ponto emitido por um dispositivo. Você também pode usar o mesmo padrão para otimizar a consulta do primeiro ponto, se seu aplicativo precisar dela.

Tópicos

- Calculado a partir da tabela de origem
- Tabela derivada para pré-computar com granularidade diária
- Calculado a partir da tabela derivada
- Combinando a partir da fonte e da tabela derivada

Calculado a partir da tabela de origem

Abaixo está um exemplo de consulta para encontrar a última medição emitida pelos serviços em uma implantação específica (por exemplo, servidores para um determinado microsserviço em uma determinada região, célula, silo e zona de disponibilidade). No aplicativo de exemplo, essa consulta retornará a última medição para centenas de servidores. Observe também que essa consulta tem um predicado de tempo ilimitado e procura dados anteriores a um determinado carimbo de data/hora.

> **Note**
>
> Para obter informações sobre as `max_by` funções `max` e, consulteFunções agregadas.

```
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
```

Tabela derivada para pré-computar com granularidade diária

Você pode converter o caso de uso anterior em um cálculo programado. Se os requisitos de seu aplicativo exigirem que você precise obter esses valores para toda a sua frota em várias regiões, células, silos, zonas de disponibilidade e microsserviços, você pode usar um cálculo de agendamento para pré-computar os valores de toda a sua frota. Esse é o poder do Timestream

para LiveAnalytics as consultas agendadas sem servidor, que permite que essas consultas sejam escalonadas de acordo com os requisitos de escalabilidade do seu aplicativo.

Abaixo está uma consulta para pré-calcular o último ponto em todos os servidores em um determinado dia. Observe que a consulta tem apenas um predicado de tempo e não um predicado nas dimensões. O predicado de hora limita a consulta ao dia anterior a partir do momento em que o cálculo é acionado com base na expressão de agendamento especificada.

```
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
```

Abaixo está uma configuração para o cálculo agendado usando a consulta anterior, que executa essa consulta às 01:00 horas UTC todos os dias para calcular o agregado do dia anterior. A expressão do cronograma cron (0 1 * *? *) controla esse comportamento e funciona uma hora após o término do dia para considerar quaisquer dados que cheguem com até um dia de atraso.

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

Calculado a partir da tabela derivada

Depois de definir a tabela derivada usando a configuração anterior e pelo menos uma instância da consulta agendada ter materializado os dados na tabela derivada, agora você pode consultar a tabela derivada para obter a medição mais recente. Abaixo está um exemplo de consulta na tabela derivada.

```
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
```

Combinando a partir da fonte e da tabela derivada

Semelhante ao exemplo anterior, nenhum dado da tabela derivada terá as gravações mais recentes. Portanto, você pode usar novamente um padrão semelhante ao anterior para mesclar os dados da tabela derivada para os dados mais antigos e usar os dados de origem para a dica restante. Abaixo está um exemplo dessa consulta usando a UNION abordagem similar. Como o requisito do aplicativo é encontrar a medição mais recente antes de um período de tempo, e essa hora de início pode estar no passado, a maneira de escrever essa consulta é usar a hora fornecida, usar os dados de origem de até um dia a partir da hora especificada e, em seguida, usar a tabela derivada nos dados mais antigos. Como você pode ver no exemplo de consulta abaixo, o predicado de tempo nos dados de

origem é limitado. Isso garante um processamento eficiente na tabela de origem, que tem um volume significativamente maior de dados, e então o predicado de tempo ilimitado está na tabela derivada.

```
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
```

O texto anterior é apenas uma ilustração de como você pode estruturar as tabelas derivadas. Se você tiver anos de dados, poderá usar mais níveis de agregações. Por exemplo, você pode ter agregados mensais em cima dos agregados diários e pode ter agregados por hora antes do diário. Assim, você pode mesclar o mais recente para preencher a última hora, o horário para preencher o

último dia, o diário para preencher o último mês e o mensal para preencher o antigo. O número de níveis que você configura em relação ao cronograma de atualização dependerá de seus requisitos de frequência com que essas consultas são emitidas e de quantos usuários as estão emitindo simultaneamente.

Valores de dimensão exclusivos

Você pode ter um caso de uso em que tenha painéis nos quais deseja usar os valores exclusivos das dimensões como variáveis para detalhar as métricas correspondentes a uma fatia específica dos dados. O instantâneo abaixo é um exemplo em que o painel pré-preenche os valores exclusivos de várias dimensões, como região, célula, silo, microsserviço e zona de disponibilidade. Aqui, mostramos um exemplo de como você pode usar consultas agendadas para acelerar significativamente o cálculo desses valores distintos dessas variáveis a partir das métricas que você está monitorando.

Tópicos

- Em dados brutos
- Pré-calcule valores de dimensão exclusivos
- Calculando as variáveis da tabela derivada

Em dados brutos

Você pode usar SELECT DISTINCT para calcular os valores distintos vistos em seus dados. Por exemplo, se você quiser obter os valores distintos da região, você pode usar a consulta deste formulário.

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

Você pode estar rastreando milhões de dispositivos e bilhões de séries temporais. No entanto, na maioria dos casos, essas variáveis interessantes são para dimensões de cardinalidade mais baixas, nas quais você tem de alguns a dezenas de valores. DISTINCTA computação a partir de dados brutos pode exigir a digitalização de grandes volumes de dados.

Pré-calcule valores de dimensão exclusivos

Você quer que essas variáveis sejam carregadas rapidamente para que seus painéis sejam interativos. Além disso, essas variáveis geralmente são calculadas em cada carga do painel, então você quer que elas também sejam econômicas. Você pode otimizar a localização dessas variáveis usando consultas agendadas e materializando-as em uma tabela derivada.

Primeiro, você precisa identificar as dimensões para as quais você precisa calcular os DISTINCT valores ou colunas que você usará nos predicados ao calcular o DISTINCT valor.

Neste exemplo, você pode ver que o painel está preenchendo valores distintos para as dimensões região, célula, silo, zona de disponibilidade e microsserviço. Portanto, você pode usar a consulta abaixo para pré-calcular esses valores exclusivos.

```
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
```

Há algumas coisas importantes a serem observadas aqui.

- Você pode usar um cálculo programado para pré-computar valores para várias consultas diferentes. Por exemplo, você está usando a consulta anterior para pré-computar valores para cinco variáveis diferentes. Portanto, você não precisa de um para cada variável. Você pode usar esse mesmo padrão para identificar a computação compartilhada em vários painéis para otimizar o número de consultas agendadas que você precisa manter.
- Os valores exclusivos das dimensões não são inerentemente dados de séries temporais. Então você converte isso em séries temporais usando o @scheduled_runtime. Ao associar esses dados ao parâmetro @scheduled_runtime, você também pode rastrear quais valores exclusivos apareceram em um determinado momento, criando dados de séries temporais a partir deles.
- No exemplo anterior, você verá um valor métrico sendo rastreado. Este exemplo usa COUNT (*). Você pode calcular outros agregados significativos se quiser rastreá-los em seus painéis.

Abaixo está uma configuração para um cálculo programado usando a consulta anterior. Neste exemplo, ele é configurado para ser atualizado uma vez a cada 15 minutos usando a expressão de agendamento 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": "******"
}
```

Calculando as variáveis da tabela derivada

Depois que o cálculo programado pré-materializa os valores exclusivos na tabela derivada hc_unique_dimensions_pt15m, você pode usar a tabela derivada para calcular com eficiência os valores exclusivos das dimensões. Abaixo estão exemplos de consultas sobre como calcular os valores exclusivos e como você pode usar outras variáveis como predicados nessas consultas de valores exclusivos.

Region

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

Célula

```
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
```

Microsserviço

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

Zona de disponibilidade

```
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
```

Lidando com dados que chegam tardiamente

Você pode ter cenários em que os dados chegam com um atraso significativo, por exemplo, o horário em que os dados foram ingeridos no Timestream LiveAnalytics está significativamente atrasado em comparação com o timestamp associado às linhas que são ingeridas. Nos exemplos anteriores, você viu como usar os intervalos de tempo definidos pelo parâmetro @scheduled_runtime para contabilizar alguns dados que chegam tarde. No entanto, se você tiver casos de uso em que os dados podem ser atrasados em horas ou dias, talvez seja necessário um padrão diferente para garantir que seus pré-cálculos na tabela derivada sejam atualizados adequadamente para refletir esses dados que chegam tardiamente. Para obter informações gerais sobre dados de chegada tardia, consulte. Gravando dados (inserções e acréscimos)

A seguir, você verá duas maneiras diferentes de lidar com esses dados que chegam tardiamente.

- Se você tiver atrasos previsíveis na chegada dos dados, poderá usar outra computação programada “atualizada” para atualizar seus agregados para dados que chegam tarde.
- Se você tiver atrasos imprevisíveis ou dados ocasionais de chegada tardia, poderá usar execuções manuais para atualizar as tabelas derivadas.

Esta discussão aborda cenários de chegada tardia de dados. No entanto, os mesmos princípios se aplicam às correções de dados, nas quais você modificou os dados na tabela de origem e deseja atualizar os agregados nas tabelas derivadas.

Tópicos

- Consultas de atualização programadas
- Execuções manuais para dados imprevisíveis que chegam tardiamente

Consultas de atualização programadas

Consulta agregando dados que chegaram a tempo

Abaixo está um padrão: você verá como usar uma forma automatizada de atualizar seus agregados se houver atrasos previsíveis na chegada dos dados. Considere um dos exemplos anteriores de um cálculo programado em dados em tempo real abaixo. Esse cálculo programado atualiza a tabela derivada uma vez a cada 30 minutos e já contabiliza dados com até uma hora de atraso.

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

Consulta de atualização atualizando os agregados para dados que chegam tardiamente

Agora, se você considerar o caso, seus dados podem ser atrasados em cerca de 12 horas. Abaixo está uma variante da mesma consulta. No entanto, a diferença é que ele calcula os agregados em dados que estão atrasados em até 12 horas em comparação com quando a computação programada está sendo acionada. Por exemplo, você vê a consulta no exemplo abaixo. O intervalo de tempo que essa consulta tem como alvo é entre 2h e 14h antes de ser acionada. Além disso, se você observar a expressão de cronograma cron (0 0,12 * *? *), ele acionará o cálculo às 00:00 UTC e 12:00 todos os diasUTC. Portanto, quando a consulta é acionada em 01/12/2021 00:00:00, a consulta atualiza os agregados no intervalo de tempo 2021-11-30 10:00:00 a 2021-11-30 22:00:00. As consultas agendadas usam uma semântica ascendente semelhante à Timestream para LiveAnalytics gravações, em que essa consulta de recuperação atualizará os valores agregados com valores mais novos se houver dados atrasados na janela ou se novos agregados forem encontrados (por exemplo, um novo agrupamento aparecerá nesse agregado que não estava presente quando a computação programada original foi acionada) e, em seguida, o novo agregado será inserido na tabela derivada. Da mesma forma, quando a próxima instância for acionada em 01/12/2021 12:00:00, essa instância atualizará os agregados no intervalo 2021-11-30 22:00:00 a 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": "******"
}
```

O exemplo anterior é uma ilustração, supondo que sua chegada tardia esteja limitada a 12 horas e que não há problema em atualizar a tabela derivada uma vez a cada 12 horas para que os dados cheguem depois da janela em tempo real. Você pode adaptar esse padrão para atualizar sua tabela derivada uma vez a cada hora, para que ela reflita os dados que chegam tardiamente mais cedo. Da mesma forma, você pode adaptar o intervalo de tempo para ser superior a 12 horas, por exemplo, um dia ou até uma semana ou mais, para lidar com dados previsíveis que chegam tarde.

Execuções manuais para dados imprevisíveis que chegam tardiamente

Pode haver casos em que você tenha dados imprevisíveis chegando atrasados ou tenha feito alterações nos dados de origem e atualizado alguns valores após o fato. Em todos esses casos, você pode acionar manualmente consultas agendadas para atualizar a tabela derivada. Abaixo está um exemplo de como você pode conseguir isso.

Suponha que você tenha o caso de uso em que você tenha a computação gravada na tabela derivada dp_per_timeseries_per_hr. Seus dados básicos na tabela devops foram atualizados no intervalo de tempo 2021-11-30 23:00:00 - 2021-12-01 00:00:00. Há duas consultas agendadas diferentes que podem ser usadas para atualizar essa tabela derivada: Multi PT3 0 mPerHr PerTimeseries DPCount e Multi. PT12HPerHrPerTimeseriesDPCountCatchUp Cada cálculo agendado para o qual você cria no Timestream LiveAnalytics tem um único ARN que você obtém ao criar o cálculo ou ao realizar uma operação de lista. Você pode usar o ARN para o cálculo e um valor para o parâmetro @scheduled_runtime obtido pela consulta para realizar essa operação.

Suponha que o cálculo para Multi PT3 0 mPerHr PerTimeseries DPCount tenha um ARN arn_1 e você queira usar esse cálculo para atualizar a tabela derivada. Como o cálculo agendado anterior atualiza os agregados 1 hora antes e 1 hora depois do valor @scheduled_runtime, você pode cobrir o intervalo de tempo da atualização (2021-11-30 23:00:00 - 2021-12-01 00:00:00) usando um valor de 2021-12-01 00:00:00 para o parâmetro @scheduled_runtime. Você pode usar o ExecuteScheduledQuery API para transmitir esse cálculo e o ARN valor do parâmetro de tempo em segundos de época (inUTC) para fazer isso. Abaixo está um exemplo usando o AWS CLI e você pode seguir o mesmo padrão usando qualquer um dos SDKs suportados pelo Timestream for. LiveAnalytics

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

No exemplo anterior, profile é o AWS perfil que tem os privilégios apropriados para fazer essa API chamada e 1638316800 corresponde à segunda época de 2021-12-01 00:00:00. Esse gatilho manual se comporta quase como o gatilho automatizado, supondo que o sistema tenha acionado essa invocação no período de tempo desejado.

Se você teve uma atualização em um período mais longo, digamos que os dados básicos foram atualizados para 2021-11-30 23:00:00 - 2021-12-01 11:00:00, então você pode acionar as consultas anteriores várias vezes para cobrir todo esse intervalo de tempo. Por exemplo, você pode fazer seis execuções diferentes da seguinte maneira.

```
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
```

Os seis comandos anteriores correspondem à computação programada invocada em 2021-12-01 00:00:00, 2021-12-01 02:00:00, 2021-12-01 04:0:00, 2021-12-01 06:00:00, 2021-12-01 08:00:00e 2021-12-01 10:00:

Como alternativa, você pode usar o cálculo Multi PT12HPerHrPerTimeseriesDPCountCatchUp acionado às 13:00:00 de 01/12/2021 para uma execução para atualizar os agregados em todo o intervalo de tempo de 12 horas. Por exemplo, se arn_2 for o ARN para esse cálculo, você poderá executar o seguinte comando a partir de. CLI

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

É importante notar que, para um gatilho manual, você pode usar um carimbo de data/hora para o parâmetro de tempo de invocação que não precisa estar alinhado com os carimbos de data/hora do gatilho automatizado. Por exemplo, no exemplo anterior, você acionou o cálculo no horário 2021-12-01 13:00:00 embora a programação automatizada só seja acionada nos timestamps 2021-12-01 10:00:00, 2021-12-01 12:00:00e 2021-12-02 00:00:00. O Timestream for LiveAnalytics fornece a flexibilidade de acioná-lo com valores apropriados, conforme necessário para suas operações manuais.

A seguir estão algumas considerações importantes ao usar o. ExecuteScheduledQuery API

- Se você estiver acionando várias dessas invocações, precisará garantir que essas invocações não gerem resultados em intervalos de tempo sobrepostos. Por exemplo, nos exemplos anteriores, havia seis invocações. Cada invocação abrange um intervalo de tempo de 2 horas e, portanto, os timestamps de invocação foram distribuídos em duas horas cada para evitar qualquer sobreposição nas atualizações. Isso garante que os dados na tabela derivada terminem em um estado que corresponda aos agregados da tabela de origem. Se você não puder garantir intervalos de tempo não sobrepostos, certifique-se de que essas execuções sejam acionadas sequencialmente uma após a outra. Se você acionar várias execuções simultaneamente que se sobrepõem em seus intervalos de tempo, poderá ver falhas de gatilho nas quais poderá ver conflitos de versão nos relatórios de erros dessas execuções. Os resultados gerados por uma invocação de consulta agendada recebem uma versão com base em quando a invocação foi acionada. Portanto, as linhas geradas por invocações mais recentes têm versões superiores. Um registro de versão superior pode sobrescrever um registro de versão inferior. Para consultas agendadas acionadas automaticamente, o Timestream for gerencia LiveAnalytics automaticamente os agendamentos para que você não veja esses problemas, mesmo que as invocações subsequentes tenham intervalos de tempo sobrepostos.
- observado anteriormente, você pode acionar as invocações com qualquer valor de timestamp para @scheduled_runtime. Portanto, é sua responsabilidade definir adequadamente os valores para que os intervalos de tempo apropriados sejam atualizados na tabela derivada correspondente aos intervalos em que os dados foram atualizados na tabela de origem.
- Você também pode usar esses gatilhos manuais para consultas agendadas que estão no DISABLED estado. Isso permite que você defina consultas especiais que não são executadas em uma programação automatizada, pois estão no DISABLED estado. Em vez disso, você pode usar os acionadores manuais neles para gerenciar correções de dados ou casos de uso de chegada tardia.

Preenchendo pré-cálculos históricos

Quando você cria um cálculo agendado, o Timestream for LiveAnalytics gerencia as execuções das consultas em andamento, onde a atualização é governada pela expressão de agendamento que você fornece. Dependendo da quantidade de dados históricos de sua tabela de origem, talvez você queira atualizar sua tabela derivada com agregados correspondentes aos dados históricos. Você pode usar a lógica anterior para acionadores manuais para preencher os agregados históricos.

Por exemplo, se considerarmos a tabela derivada per_timeseries_lastpoint_pt1d, o cálculo agendado será atualizado uma vez por dia em relação ao dia anterior. Se sua tabela de origem tiver um ano de dados, você poderá usar o ARN para esse cálculo agendado e acioná-lo manualmente para todos

os dias de até um ano, para que a tabela derivada tenha todas as consultas históricas preenchidas. Observa que todas as advertências para acionadores manuais se aplicam aqui. Além disso, se a tabela derivada for configurada de forma que a ingestão histórica seja gravada no armazenamento magnético da tabela derivada, esteja ciente das melhores práticas e limites para gravações no armazenamento magnético.

Exemplos de consultas agendadas

Esta seção contém exemplos de como você pode usar as consultas agendadas LiveAnalytics do Timestream for para otimizar os custos e os tempos de carregamento do painel ao visualizar estatísticas de toda a frota e monitorar com eficácia sua frota de dispositivos. As consultas agendadas no Timestream for LiveAnalytics permitem que você expresse suas consultas usando toda a área de SQL superfície do Timestream for. LiveAnalytics Sua consulta pode incluir uma ou mais tabelas de origem, realizar agregações ou qualquer outra consulta permitida pela SQL linguagem LiveAnalytics do Timestream for e, em seguida, armazenar os resultados da consulta em outra tabela de destino no Timestream for. LiveAnalytics

Esta seção se refere à tabela de destino de uma consulta agendada como uma tabela derivada.

Como exemplo, usaremos um DevOps aplicativo em que você monitora uma grande frota de servidores implantados em várias implantações (como regiões, células e silos), vários microsserviços e rastreia as estatísticas de toda a frota usando o Timestream for. LiveAnalytics O esquema de exemplo que usaremos está descrito em Scheduled Queries Sample Schema.

Os cenários a seguir serão descritos.

- Como converter um painel, traçar estatísticas agregadas dos dados brutos que você ingere no Timestream em uma consulta agendada e, LiveAnalytics em seguida, como usar seus agregados pré-computados para criar um novo painel mostrando estatísticas agregadas.
- Como combinar consultas agendadas para obter uma visão agregada e os dados granulares brutos, para detalhar os detalhes. Isso permite que você armazene e analise os dados brutos enquanto otimiza suas operações comuns em toda a frota usando consultas programadas.
- Como otimizar custos usando consultas agendadas descobrindo quais agregados são usados em vários painéis e fazer com que a mesma consulta agendada preencha vários painéis no mesmo painel ou em vários painéis.

Tópicos

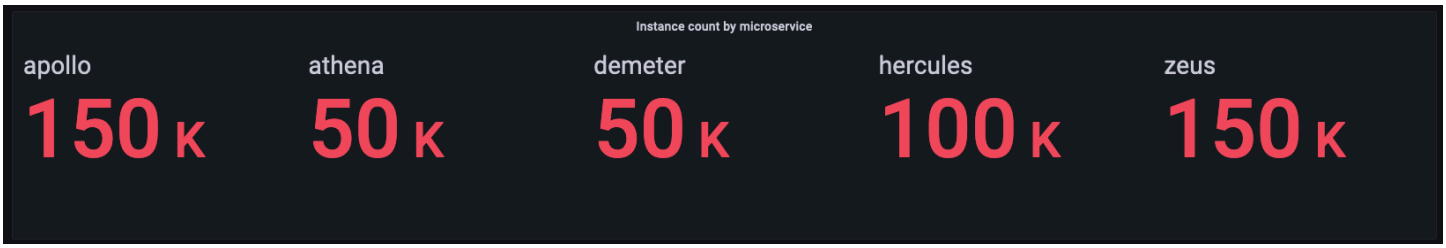
- Convertendo um painel agregado em uma consulta agendada

- Usando consultas programadas e dados brutos para detalhamentos
- Otimizando custos compartilhando consultas agendadas entre painéis
- Comparando uma consulta em uma tabela base com uma consulta de resultados de consultas agendadas

Convertendo um painel agregado em uma consulta agendada

Suponha que você esteja computando as estatísticas de toda a frota, como a contagem de hosts na frota, pelos cinco microsserviços e pelas seis regiões em que seu serviço está implantado. No instantâneo abaixo, você pode ver que há 500 mil servidores emitindo métricas, e algumas das maiores regiões (por exemplo, us-east-1) têm mais de 200 mil servidores.

A computação desses agregados, em que você está computando nomes de instância distintos em centenas de gigabytes de dados, pode resultar em latência de consulta de dezenas de segundos, além do custo de digitalização dos dados.



Consulta original do painel

O agregado mostrado no painel do painel é calculado, a partir de dados brutos, usando a consulta abaixo. A consulta usa várias SQL construções, como contagens distintas e várias funções de agregação.

```
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
)
```

Convertendo em uma consulta agendada

A consulta anterior pode ser convertida em uma consulta agendada da seguinte maneira. Primeiro, você calcula os nomes de host distintos em uma determinada implantação em uma região, célula, silo, zona de disponibilidade e microsserviço. Em seguida, você soma os hosts para calcular uma contagem de hosts por hora por microsserviço. Usando o `@scheduled_runtime` parâmetro suportado pelas consultas agendadas, você pode recalculá-lo na última hora em que a consulta é invocada. A `WHERE` cláusula `bin(@scheduled_runtime, 1h)` in the da consulta interna garante que, mesmo que a consulta seja agendada em um horário no meio da hora, você ainda receba os dados da hora inteira.

Embora a consulta calcule agregados por hora, como você verá na configuração de computação agendada, ela é configurada para ser atualizada a cada meia hora para que você receba atualizações na tabela derivada mais cedo. Você pode ajustar isso com base em seus requisitos de atualização, por exemplo, recalcular os agregados a cada 15 minutos ou recalculá-los nos limites de horas.

```
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
```

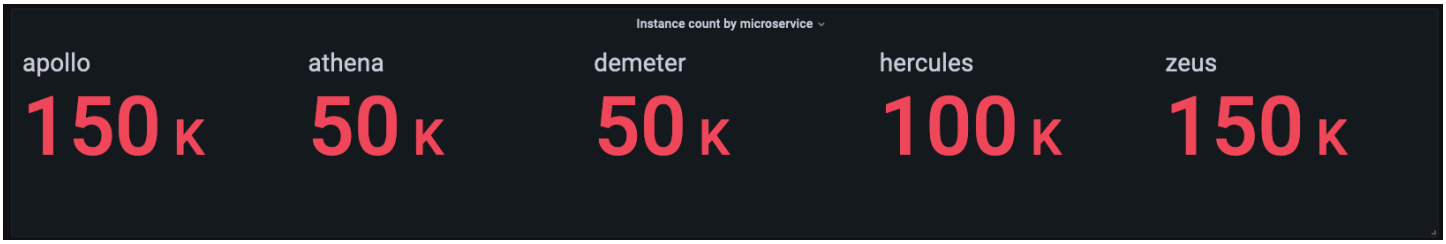
```
{
    "Name": "MultiPT30mHostCountMicroservicePerHr",
```

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

```
}
```

Usando os resultados pré-computados em um novo painel

Agora você verá como criar seu painel de visualização agregada usando a tabela derivada da consulta agendada que você criou. A partir do instantâneo do painel, você também poderá validar se os agregados calculados a partir da tabela derivada e da tabela base também coincidem. Depois de criar os painéis usando as tabelas derivadas, você notará o tempo de carregamento significativamente mais rápido e os custos mais baixos do uso das tabelas derivadas em comparação com o cálculo desses agregados a partir dos dados brutos. Abaixo está um instantâneo do painel usando dados pré-computados e a consulta usada para renderizar esse painel usando dados pré-computados armazenados na tabela “derivada”.” host_count_pt1h”. Observe que a estrutura da consulta é muito semelhante à consulta usada no painel de dados brutos, exceto que ela usa a tabela derivada que já calcula as contagens distintas que essa consulta está agregando.




```
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
)
```

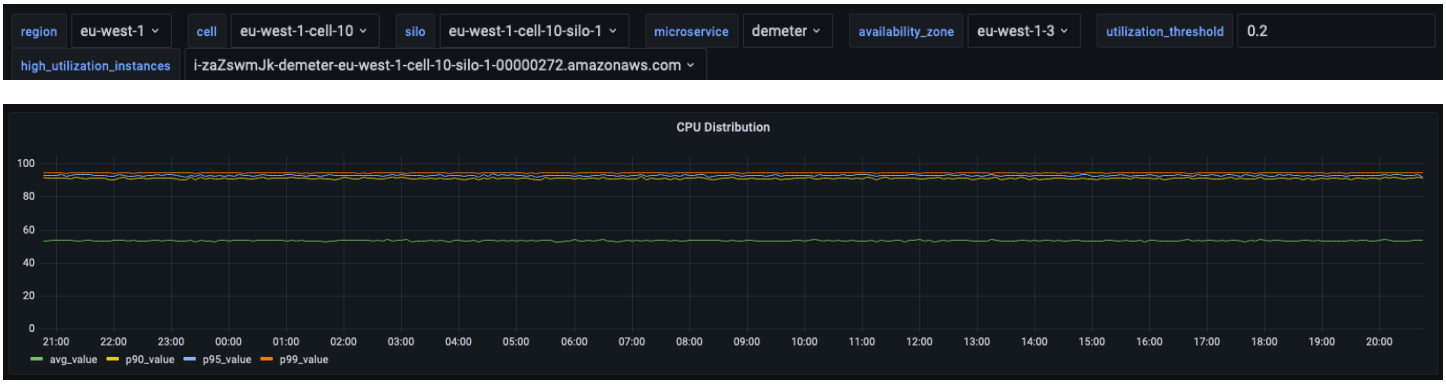
Usando consultas programadas e dados brutos para detalhamentos

Você pode usar as estatísticas agregadas em toda a sua frota para identificar áreas que precisam de detalhamento e, em seguida, usar os dados brutos para detalhar dados granulares e obter informações mais detalhadas.

Neste exemplo, você verá como usar o painel agregado para identificar qualquer implantação (uma implantação é para um determinado microsserviço em uma determinada região, célula, silo e zona de disponibilidade) que parece ter maior CPU utilização em comparação com outras implantações. Em seguida, você pode detalhar para entender melhor o uso dos dados brutos. Como esses detalhamentos podem ser pouco frequentes e acessar apenas dados relevantes para a implantação, você pode usar os dados brutos para essa análise e não precisa usar consultas agendadas.

Detalhamento por implantação

O painel abaixo fornece informações detalhadas sobre estatísticas mais granulares e em nível de servidor em uma determinada implantação. Para ajudá-lo a detalhar as diferentes partes da sua frota, esse painel usa variáveis como região, célula, silo, microsserviço e zona de disponibilidade. Em seguida, mostra algumas estatísticas agregadas dessa implantação.



Na consulta abaixo, você pode ver que os valores escolhidos na lista suspensa das variáveis são usados como predicados na `WHERE` cláusula da consulta, o que permite que você se concentre apenas nos dados da implantação. Em seguida, o painel traça as CPU métricas agregadas das instâncias dessa implantação. Você pode usar os dados brutos para realizar esse detalhamento com latência de consulta interativa para obter insights mais profundos.

```
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
```

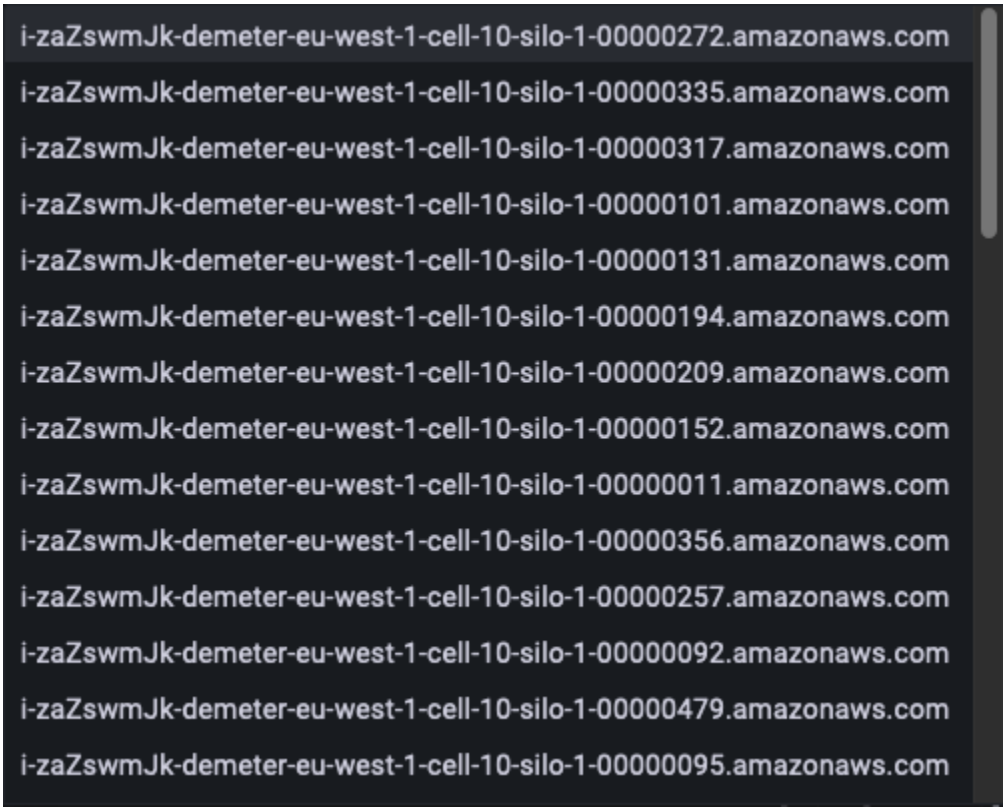
Estatísticas em nível de instância

Esse painel calcula ainda outra variável que também lista os servidores/instâncias com alta CPU utilização, classificados em ordem decrescente de utilização. A consulta usada para calcular essa variável é exibida abaixo.

```
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
```

Na consulta anterior, a variável é recalculada dinamicamente, dependendo dos valores escolhidos para as outras variáveis. Depois que a variável é preenchida para uma implantação, você pode escolher instâncias individuais da lista para visualizar melhor as métricas dessa instância. Você pode escolher as diferentes instâncias no menu suspenso dos nomes das instâncias, conforme visto no instantâneo abaixo.



Os painéis anteriores mostram as estatísticas da instância selecionada e abaixo estão as consultas usadas para obter essas estatísticas.

```
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
```

Otimizando custos compartilhando consultas agendadas entre painéis

Neste exemplo, veremos um cenário em que vários painéis exibem variações de informações semelhantes (encontrando altos CPU hosts e uma fração da frota com alta CPU utilização) e como você pode usar a mesma consulta agendada para pré-computar resultados que são usados para preencher vários painéis. Essa reutilização otimiza ainda mais seus custos, pois, em vez de usar consultas agendadas diferentes, uma para cada painel, você usa apenas o proprietário.

Painéis de painel com dados brutos

CPUutilização por região por microsserviço

O primeiro painel calcula as instâncias cuja CPU utilização média é um limite abaixo ou acima da CPU utilização acima para determinada implantação em uma região, célula, silo, zona de disponibilidade e microsserviço. Em seguida, ele classifica a região e o microserviço que tem a maior porcentagem de hosts com alta utilização. Ele ajuda a identificar a temperatura dos servidores de uma implantação específica e, posteriormente, detalhar para entender melhor os problemas.

A consulta do painel demonstra a flexibilidade do SQL suporte do Timestream for para realizar tarefas analíticas complexas com expressões de tabela, funções de janela, junções e LiveAnalytics assim por diante comuns.

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

Consulta:

```
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
```

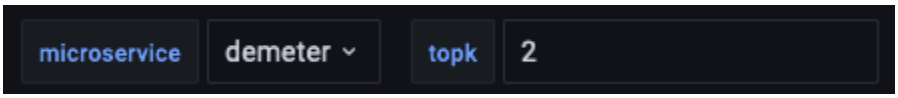
Pesquise um microsserviço para encontrar pontos quentes

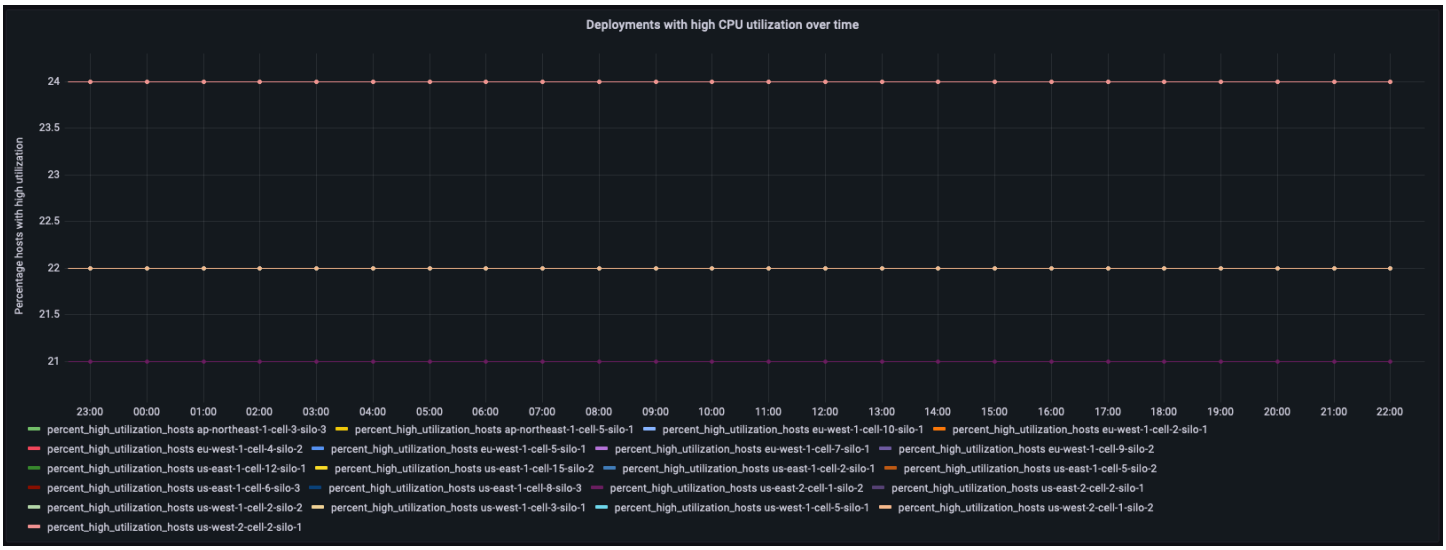
O próximo painel permite que você se aprofunde em um dos microsserviços para descobrir qual região, célula e silo específicos desse microsserviço está executando a fração de sua frota com

maior utilização. CPU Por exemplo, no painel de controle de toda a frota, você viu o demeter de microsserviços aparecer nas primeiras posições do ranking, então, neste painel, você quer se aprofundar nesse microsserviço.

Esse painel usa uma variável para escolher o microsserviço para detalhamento, e os valores da variável são preenchidos usando valores exclusivos da dimensão. Depois de escolher o microsserviço, o restante do painel é atualizado.

Como você pode ver abaixo, o primeiro painel traça a porcentagem de hosts em uma implantação (uma região, célula e silo para um microsserviço) ao longo do tempo e a consulta correspondente que é usada para traçar o painel. Esse gráfico em si identifica uma implantação específica com maior porcentagem de hosts com altaCPU.





Consulta:

```
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
```

Conversão em uma única consulta agendada, permitindo a reutilização

É importante observar que um cálculo semelhante é feito nos diferentes painéis dos dois painéis. Você pode definir uma consulta agendada separada para cada painel. Aqui você verá como otimizar ainda mais seus custos definindo uma consulta agendada cujos resultados podem ser usados para renderizar todos os três painéis.

A seguir está a consulta que captura os agregados que são calculados e usados em todos os diferentes painéis. Você observará vários aspectos importantes na definição dessa consulta agendada.

- A flexibilidade e a potência da área de SQL superfície são suportadas por consultas programadas, nas quais você pode usar expressões comuns de tabela, junções, declarações de caso etc.
- Você pode usar uma consulta agendada para calcular as estatísticas em uma granularidade mais precisa do que um painel específico pode precisar e para todos os valores que um painel pode usar para diferentes variáveis. Por exemplo, você verá que os agregados são computados em uma região, célula, silo e microsserviço. Portanto, você pode combiná-los para criar agregados em nível de região ou região e microsserviços. Da mesma forma, a mesma consulta calcula os agregados para todas as regiões, células, silos e microsserviços. Ele permite que você aplique filtros nessas colunas para obter os agregados de um subconjunto dos valores. Por exemplo, você pode calcular os agregados para qualquer região, digamos us-east-1, ou qualquer microsserviço, digamos, demeter ou detalhar uma implantação específica em uma região, célula, silo e microsserviço. Essa abordagem otimiza ainda mais seus custos de manutenção dos agregados pré-computados.

```
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
```

Veja a seguir uma definição de consulta agendada para a consulta anterior. A expressão de agendamento é configurada para ser atualizada a cada 30 minutos e atualiza os dados por até uma hora atrás, novamente usando a construção bin (@scheduled_runtime, 1h) para obter os eventos de uma hora inteira. Dependendo dos requisitos de atualização do seu aplicativo, você pode configurá-lo para ser atualizado com mais ou menos frequência. Ao usar WHERE time BETWEEN bin (@scheduled_runtime, 1h) - 1h AND bin (@scheduled_runtime, 1h) + 1h, podemos garantir que, mesmo se você estiver atualizando uma vez a cada 15 minutos, você receberá os dados completos da hora atual e da hora anterior.

Posteriormente, você verá como os três painéis usam esses agregados gravados na tabela deployment_cpu_stats_per_hr para visualizar as métricas que são relevantes para o painel.

```
{
    "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)
```

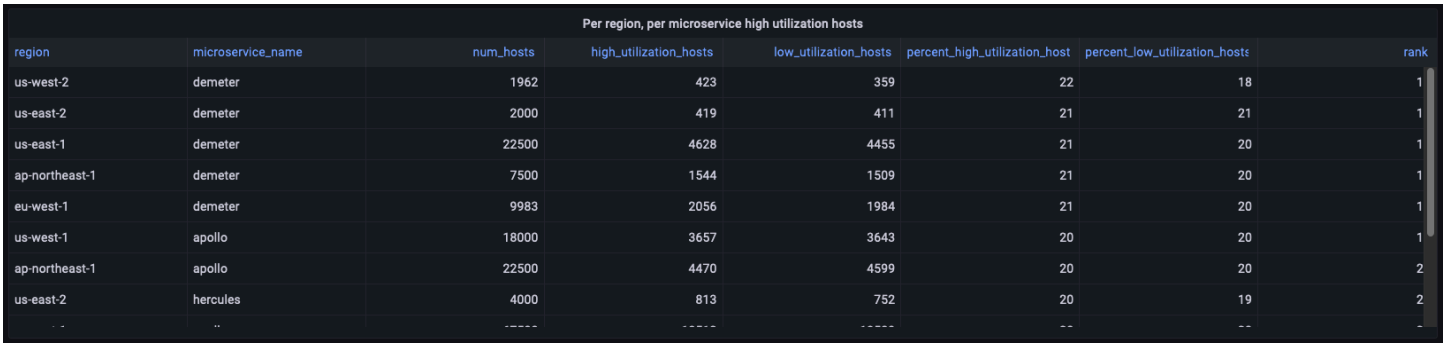
```
as hour,    AVG(cpu_user) AS instance_avg_metric    FROM raw_data.devops
WHERE time BETWEEN bin(@scheduled_runtime, 1h) - 1h AND bin(@scheduled_runtime,
1h) + 1h    AND measure_name = 'metrics'    GROUP BY region, cell, silo,
availability_zone, microservice_name, instance_name, bin(time, 1h)    ),
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": "******"
}
```

Painel de resultados pré-computados

Hosts CPU de alta utilização

Para os hosts de alta utilização, você verá como os diferentes painéis usam os dados de deployment_cpu_stats_per_hr para calcular os diferentes agregados necessários para os painéis. Por exemplo, esses painéis fornecem informações em nível de região, portanto, relatam agregados agrupados por região e microsserviço, sem filtrar nenhuma região ou microsserviço.



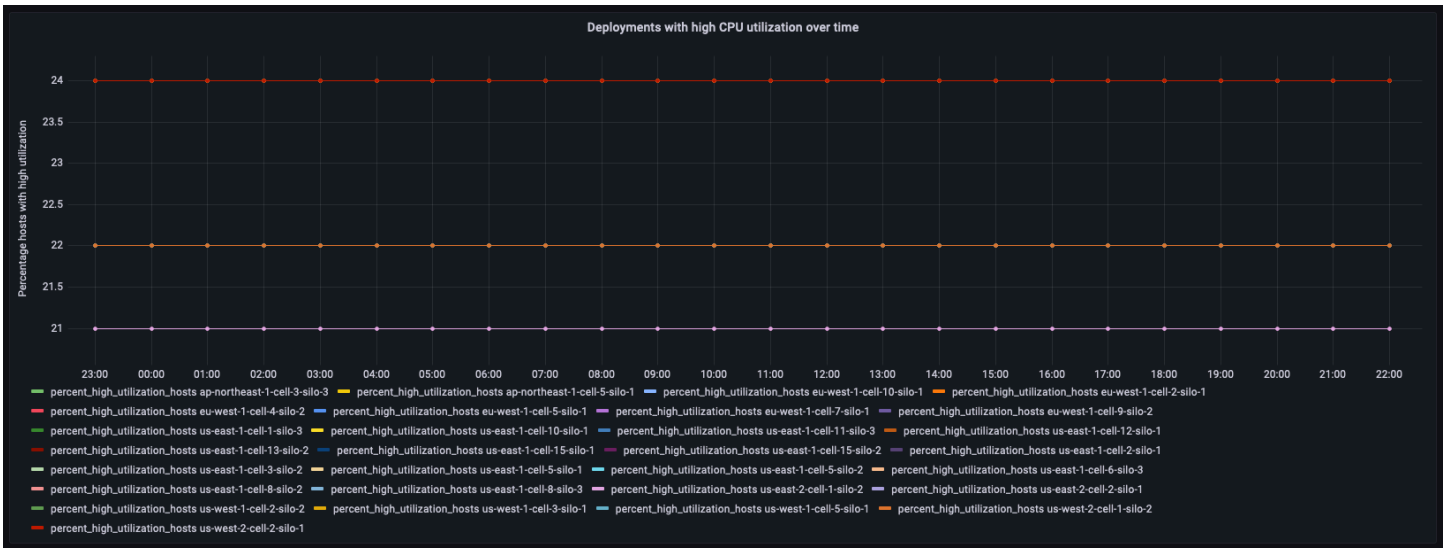
Per region, per microservice high utilization hosts

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
```

Aprofunde-se em um microsserviço para encontrar CPU implantações de alto uso

O próximo exemplo usa novamente a tabela derivada deployment_cpu_stats_per_hr, mas agora aplica um filtro para um microsserviço específico (demeter neste exemplo, pois relatou hosts de alta utilização no painel agregado). Esse painel rastreia a porcentagem de hosts de alta CPU utilização ao longo do tempo.




```
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
```

Comparando uma consulta em uma tabela base com uma consulta de resultados de consultas agendadas

Neste exemplo de consulta Timestream, usamos o esquema, exemplos de consultas e saídas a seguir para comparar uma consulta em uma tabela base com uma consulta em uma tabela derivada de resultados de consultas agendadas. Com uma consulta agendada bem planejada, você pode obter uma tabela derivada com menos linhas e outras características que podem levar a consultas mais rápidas do que seria possível na tabela base original.

Para ver um vídeo que descreve esse cenário, consulte Melhore o desempenho das consultas e reduza os custos usando consultas programadas no Amazon Timestream para. LiveAnalytics

Neste exemplo, usamos o seguinte cenário:

- Região — us-east-1
- Tabela base — `"clickstream"."shopping"`
- Tabela derivada — `"clickstream"."aggregate"`

Tabela base

O seguinte descreve o esquema da tabela base.

Coluna	Tipo	Fluxo de tempo para o tipo de LiveAnalytics atributo
channel	varchar	MULTI
description	varchar	MULTI
evento	varchar	DIMENSION
ip_address	varchar	DIMENSION
nome_medida	varchar	MEASURE_NAME

Coluna	Tipo	Fluxo de tempo para o tipo de LiveAnalytics atributo
product	varchar	MULTI
product_id	varchar	MULTI
quantity	double	MULTI
consulta	varchar	MULTI
session_id	varchar	DIMENSION
grupo_usuário	varchar	DIMENSION
user_id	varchar	DIMENSION

A seguir estão descritas as medidas da tabela base. Uma tabela base se refere a uma tabela no Timestream na qual a consulta agendada é executada.

- nome_medida — `metrics`
- dados — vários
- dimensões:

```
[ ( user_group, varchar ),( user_id, varchar ),( session_id, varchar ),( ip_address,
 varchar ),( event, varchar ) ]
```

Consulta em uma tabela base

Veja a seguir uma consulta ad-hoc que reúne contagens por um agregado de 5 minutos em um determinado intervalo de tempo.

```
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
```

Saída:

```
duration:1.745 sec
Bytes scanned: 29.89 MB
Query Id: AEBQEANMHG7MHHBHCKJ3BSOE3QUGIDBGWCCP5I6J6YUW5CVJZ2M3JCJ27QRMM7A
Row count:5
```

Consulta agendada

Veja a seguir uma consulta agendada que é executada a cada 5 minutos.

```
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
```

Consulta em uma tabela derivada

Veja a seguir uma consulta ad-hoc em uma tabela derivada. Uma tabela derivada se refere a uma tabela Timestream que contém os resultados de uma consulta agendada.

```
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'
```

Saída:

```
duration: 0.2960 sec
Bytes scanned: 235.00 B
QueryID: AEBQEANMHHAAQU4FFTT6CFM6UYXTL4SMLZV22MFP4KV2Z7IRVOPLOMLDD6BR33Q
Row count: 5
```

Comparação

Veja a seguir uma comparação dos resultados de uma consulta em uma tabela base com uma consulta em uma tabela derivada. A mesma consulta em uma tabela derivada que tem resultados agregados feitos por meio de uma consulta agendada é concluída mais rapidamente com menos bytes digitalizados.

Esses resultados mostram o valor de usar consultas agendadas para agregar dados para consultas mais rápidas.

	Consulta na tabela base	Consulta na tabela derivada
Duração	1,745 seg	0,2960 seg
Bytes digitalizados	29,89 MB	235 bytes
Contagem de linhas	5	5

Usando UNLOAD para exportar os resultados da consulta para o S3 do Timestream for LiveAnalytics

LiveAnalytics Por enquanto, o Amazon Timestream permite que você exporte os resultados da consulta para o Amazon S3 de forma econômica e segura usando a declaração. `UNLOAD` Usando a `UNLOAD` instrução, agora você pode exportar dados de séries temporais para buckets S3 selecionados no formato Apache Parquet ou Comma Separated Values (CSV), que oferece flexibilidade para armazenar, combinar e analisar seus dados de séries temporais com outros serviços. A `UNLOAD` declaração permite que você exporte os dados de forma compactada, o que reduz os dados transferidos e o espaço de armazenamento necessário. `UNLOAD`também oferece suporte ao particionamento com base em atributos selecionados ao exportar os dados, melhorando o desempenho e reduzindo o tempo de processamento dos serviços downstream que acessam os dados. Além disso, você pode usar as chaves gerenciadas do Amazon S3 (SSE-S3) ou as chaves gerenciadas do AWS Key Management Service (AWS KMS) (SSE-KMS) para criptografar seus dados exportados.

Benefícios UNLOAD do Timestream para LiveAnalytics

Os principais benefícios de usar a `UNLOAD` declaração são os seguintes.

- Facilidade operacional — Com a UNLOAD declaração, você pode exportar gigabytes de dados em uma única solicitação de consulta no formato Apache Parquet ou no CSV formato, oferecendo flexibilidade para selecionar o formato mais adequado às suas necessidades de processamento posterior e facilitando a criação de data lakes.
- Seguro e econômico — o UNLOAD statement fornece a capacidade de exportar seus dados para um bucket S3 de forma compactada e criptografar (SSE- KMS ou SSE _S3) seus dados usando chaves gerenciadas pelo cliente, reduzindo os custos de armazenamento de dados e protegendo contra acesso não autorizado.
- Desempenho — Usando a UNLOAD instrução, você pode particionar os dados ao exportar para um bucket do S3. O particionamento dos dados permite que os serviços downstream processem os dados em paralelo, reduzindo o tempo de processamento. Além disso, os serviços downstream podem processar somente os dados de que precisam, reduzindo os recursos de processamento necessários e, portanto, os custos associados.

Casos de uso UNLOAD do Timestream for LiveAnalytics

Você pode usar a UNLOAD instrução para gravar dados em seu bucket do S3 da seguinte forma.

- Crie um data warehouse — você pode exportar gigabytes de resultados de consultas para o bucket do S3 e adicionar mais facilmente dados de séries temporais ao seu data lake. Você pode usar serviços como o Amazon Athena e o Amazon Redshift para combinar seus dados de séries temporais com outros dados relevantes para obter insights comerciais complexos.
- Crie pipelines de dados de IA e ML — A UNLOAD declaração permite que você crie facilmente pipelines de dados para seus modelos de aprendizado de máquina que acessam dados de séries temporais, facilitando o uso de dados de séries temporais com serviços como Amazon e SageMaker Amazon. EMR
- Simplifique o ETL processamento — a exportação de dados para buckets do S3 pode simplificar o processo de execução das operações Extract, Transform, Load (ETL) nos dados, permitindo que você use facilmente ferramentas ou serviços de terceiros, AWS como o AWS Glue, para processar e transformar os dados.

UNLOADConceitos

Sintaxe

```
UNLOAD (SELECT statement)
```

```
 TO 's3://bucket-name/folder'
 WITH ( option = expression [, ...] )
```

onde option está

```
{ 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>'
 | }
```

Parâmetros

SELECTdeclaração

A instrução de consulta usada para selecionar e recuperar dados de um ou mais Timestream para tabelas. LiveAnalytics

```
(SELECT column 1, column 2, column 3 from database.table
      where measure_name = "ABC" and timestamp between ago (1d) and now() )
```

Cláusula TO

```
TO 's3://bucket-name/folder'
```

ou

```
TO 's3://access-point-alias/folder'
```

A TO cláusula na UNLOAD instrução especifica o destino para a saída dos resultados da consulta. Você precisa fornecer o caminho completo, incluindo o nome do bucket do Amazon S3 ou o Amazon S3 com a access-point-alias localização da pasta no Amazon S3, onde o Timestream grava os objetos do arquivo de saída. LiveAnalytics O bucket do S3 deve pertencer à mesma conta e estar na mesma região. Além do conjunto de resultados da consulta, o Timestream for LiveAnalytics grava os arquivos de manifesto e metadados na pasta de destino especificada.

PARTITIONEDCláusula _BY

```
partitioned_by = ARRAY [col_name[,…] , (default: none)
```

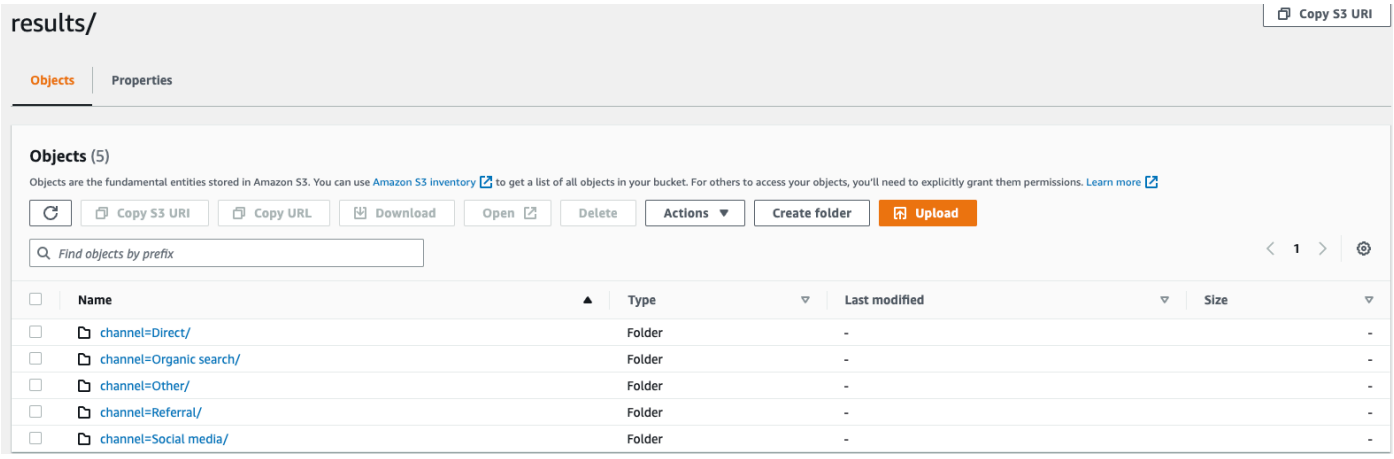
A `partitioned_by` cláusula é usada em consultas para agrupar e analisar dados em um nível granular. Ao exportar os resultados da consulta para o bucket do S3, você pode optar por particionar os dados com base em uma ou mais colunas na consulta selecionada. Ao particionar os dados, os dados exportados são divididos em subconjuntos com base na coluna de partição e cada subconjunto é armazenado em uma pasta separada. Na pasta de resultados que contém os dados exportados, uma subpasta `folder/results/partition column = partition value/` é criada automaticamente. No entanto, observe que as colunas particionadas não estão incluídas no arquivo de saída.

`partitioned_by`não é uma cláusula obrigatória na sintaxe. Se você optar por exportar os dados sem nenhum particionamento, poderá excluir a cláusula na sintaxe.

Example

Supondo que você esteja monitorando os dados do fluxo de cliques do seu site e tenha 5 canais de tráfego`direct`, a saber,`Social Media`,`Organic Search`, e. `Other` `Referral` Ao exportar os dados, você pode optar por particioná-los usando a coluna`Channel`. Em sua pasta de dados`s3://bucketname/results`, você terá cinco pastas, cada uma com o nome do respectivo canal. Por exemplo, `s3://bucketname/results/channel=Social Media/`. nessa pasta, você encontrará os dados de todos os clientes que acessaram seu site por meio do `Social Media` canal. Da mesma forma, você terá outras pastas para os canais restantes.

Dados exportados particionados pela coluna Canal



FORMAT

```
format = [ '{ CSV | PARQUET }' , default: CSV
```

As palavras-chave para especificar o formato dos resultados da consulta gravados em seu bucket do S3. Você pode exportar os dados como um valor separado por vírgula (CSV) usando uma vírgula (,) como delimitador padrão ou no formato Apache Parquet, um formato de armazenamento em colunas aberto eficiente para análise.

COMPRESSION

```
compression = [ '{ GZIP | NONE }' ], default: GZIP
```

Você pode compactar os dados exportados usando o algoritmo de compactação GZIP ou descompactá-los especificando a opção. NONE

ENCRYPTION

```
encryption = [ '{ SSE_KMS | SSE_S3 }' ], default: SSE_S3
```

Os arquivos de saída no Amazon S3 são criptografados usando a opção de criptografia selecionada. Além dos seus dados, os arquivos de manifesto e metadados também são criptografados com base na opção de criptografia selecionada. Atualmente, oferecemos suporte à SSE criptografia _S3 e SSE _KMS. SSE_S3 é uma criptografia do lado do servidor com o Amazon S3 criptografando os dados usando criptografia padrão de criptografia avançada () de 256 bits. AES SSE_ KMS é uma criptografia do lado do servidor para criptografar dados usando chaves gerenciadas pelo cliente.

KMS_KEY

```
kms_key = '<string>'
```

KMSA chave é uma chave definida pelo cliente para criptografar os resultados da consulta exportada. KMSA chave é gerenciada com segurança pelo AWS Key Management Service (AWS KMS) e usada para criptografar arquivos de dados no Amazon S3.

FIELD_DELIMITER

```
field_delimiter ='<character>' , default: (,)
```

Ao exportar os dados em CSV formato, esse campo especifica um único ASCII caractere usado para separar campos no arquivo de saída, como caractere vertical (|), vírgula (,) ou tabulação (/t). O delimitador padrão para CSV arquivos é um caractere de vírgula. Se um valor em seus dados contiver o delimitador escolhido, o delimitador será citado com um caractere de aspa. Por exemplo, se o valor em seus dados contiver`Time,stream`, esse valor será cotado como `"Time,stream"` nos dados exportados. O caractere de aspas usado pelo Timestream para LiveAnalytics são aspas duplas (“).

Evite especificar o caractere de retorno do carro (ASCII13, hexadecimal`0D`, texto '\ r') ou o caractere de quebra de linha (ASCII10, hexadecimal 0A, texto'\n') como `FIELD_DELIMITER` se você quiser incluir cabeçalhos no`CSV`, pois isso impedirá que muitos analisadores consigam analisar os cabeçalhos corretamente na saída resultante. CSV

ESCAPED_POR

```
escaped_by = '<character>', default: (\)
```

Ao exportar os dados em CSV formato, esse campo especifica o caractere que deve ser tratado como um caractere de escape no arquivo de dados gravado no bucket do S3. A fuga acontece nos seguintes cenários:

1. Se o valor em si contiver o caractere de aspa (“), ele será escapado usando um caractere de escape. Por exemplo, se o valor for`Time"stream`, onde (\) é o caractere de escape configurado, ele será escapado como`Time\"stream`.
2. Se o valor contiver o caractere de escape configurado, ele será escapado. Por exemplo, se o valor for`Time\stream`, ele será escapado como`Time\\stream`.

> **Note**
>
> Se a saída exportada contiver tipos de dados complexos, como matrizes, linhas ou séries temporais, ela será serializada como uma string. JSON Veja um exemplo a seguir.

Tipo de dados	Valor real	Como o valor é escapado no CSV formato [string serializa daJSON]
Array	`[ 23,24,25 ]`	`"[23,24,25]"`

Tipo de dados	Valor real	Como o valor é escapado no CSV formato [string serializadaJSON]
Linha	`( x=23.0, y=hello )`	`"{\"x\":23.0,\"y\":\"hello\"}"`
Série temporal	`[ ( 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'
```

Ao exportar os dados em CSV formato, esse campo permite incluir nomes de colunas como a primeira linha dos arquivos de CSV dados exportados.

Os valores aceitos são 'verdadeiro' e 'falso' e o valor padrão é 'falso'. Opções de transformação de texto, como `escaped_by` e também `field_delimiter` se aplicam aos cabeçalhos.

> **Note**
>
> Ao incluir cabeçalhos, é importante que você não selecione um caractere de retorno de carro (ASCII13, hexadecimal 0D, texto '\ r') ou um caractere de quebra de linha (ASCII10, hexadecimal 0A, texto'\n') como o`FIELD_DELIMITER`, pois isso impedirá que muitos analisadores consigam analisar os cabeçalhos corretamente na saída resultante. CSV

MAX_FILE_SIZE

```
max_file_size = 'X[MB|GB]' , default: '78GB'
```

Esse campo especifica o tamanho máximo dos arquivos que a UNLOAD declaração cria no Amazon S3. A UNLOAD declaração pode criar vários arquivos, mas o tamanho máximo de cada arquivo gravado no Amazon S3 será aproximadamente o especificado nesse campo.

O valor do campo deve estar entre 16 MB e 78 GB, inclusive. Você pode especificá-lo em números inteiros12GB, como, ou em decimais, como ou. 0.5GB 24.7MB O valor padrão é 78 GB.

O tamanho real do arquivo é aproximado quando o arquivo está sendo gravado, portanto, o tamanho máximo real pode não ser exatamente igual ao número especificado.

O que está gravado no meu bucket do S3?

Para cada UNLOAD consulta executada com sucesso, o Timestream for LiveAnalytics grava os resultados da consulta, o arquivo de metadados e o arquivo de manifesto no bucket do S3. Se você particionou os dados, você tem todas as pastas de partição na pasta de resultados. O arquivo de manifesto contém uma lista dos arquivos que foram gravados pelo UNLOAD comando. O arquivo de metadados contém informações que descrevem as características, propriedades e atributos dos dados gravados.

Qual é o nome do arquivo exportado?

O nome do arquivo exportado contém dois componentes, o primeiro componente é o QueryID e o segundo componente é um identificador exclusivo.

CSVarquivos

```
S3://bucket_name/results/<queryid>_<UUID>.csv
S3://bucket_name/results/<partitioncolumn>=<partitionvalue>/<queryid>_<UUID>.csv
```

Arquivo compactado CSV

```
S3://bucket_name/results/<partitioncolumn>=<partitionvalue>/<queryid>_<UUID>.gz
```

Arquivo de parquet

```
S3://bucket_name/results/<partitioncolumn>=<partitionvalue>/<queryid>_<UUID>.parquet
```

Arquivos de metadados e manifestos

```
S3://bucket_name/<queryid>_<UUID>_manifest.json
```

```
S3://bucket_name/<queryid>_<UUID>_metadata.json
```

Como os dados no CSV formato são armazenados em um nível de arquivo, quando você compacta os dados ao exportar para o S3, o arquivo terá uma extensão “.gz”. No entanto, os dados no Parquet são compactados no nível da coluna, portanto, mesmo quando você compacta os dados durante a exportação, o arquivo ainda terá a extensão.parquet.

Quais informações cada arquivo contém?

Arquivo manifesto

O arquivo de manifesto fornece informações sobre a lista de arquivos que são exportados com a UNLOAD execução. O arquivo de manifesto está disponível no bucket S3 fornecido com um nome de arquivo:`s3://<bucket_name>/<queryid>_<UUID>_manifest.json`. O arquivo de manifesto conterá o URL dos arquivos na pasta de resultados, o número de registros e o tamanho dos respectivos arquivos e os metadados da consulta (que são o total de bytes e o total de linhas exportadas para o S3 para a consulta).

```
{
  "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"
  }
}
```

Metadados

O arquivo de metadados fornece informações adicionais sobre o conjunto de dados, como nome da coluna, tipo de coluna e esquema. <queryid>O arquivo de metadados está disponível no bucket S3 fornecido com um nome de arquivo: S3: //bucket_name/ _< >_metadata.json UUID

Veja a seguir um exemplo de um arquivo de metadados.

```
{
    "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"
  }
}
```

As informações da coluna compartilhadas no arquivo de metadados têm a mesma estrutura `ColumnInfo` enviada na API Resposta da consulta para `SELECT` consultas.

Resultados

A pasta de resultados contém seus dados exportados no formato Apache Parquet ou no formato. CSV

Exemplo

Quando você envia uma `UNLOAD` consulta como a abaixo via QueryAPI,

```
UNLOAD(SELECT user_id, ip_address, event, session_id, measure_name, time, query,
 quantity, product_id, channel
                    FROM sample_clickstream.sample_shopping WHERE time BETWEEN ago(2d)
 AND now())
                TO 's3://my_timestream_unloads/withoutpartition/' WITH ( format='CSV',
 compression='GZIP')
```

`UNLOAD`a resposta da consulta terá 1 linha * 3 colunas. Essas 3 colunas são:

- linhas do tipo BIGINT - indicando o número de linhas exportadas
- metadataFile do tipo VARCHAR - que é o S3 do arquivo URI de metadados exportado
- manifestFile do tipo VARCHAR - que é o S3 do arquivo URI de manifesto exportado

Você receberá a seguinte resposta do 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
    }
```

```
}
```

Tipos de dados

A UNLOAD instrução suporta todos os tipos de dados da linguagem de consulta Timestream LiveAnalytics for descrita emTipos de dados compatíveis, exceto e. `time` `unknown`

Pré-requisitos do Timestream para UNLOAD LiveAnalytics

A seguir estão os pré-requisitos para gravar dados no S3 usando UNLOAD o Timestream for. LiveAnalytics

- Você deve ter permissão para ler dados do Timestream para que as LiveAnalytics tabelas sejam usadas em um UNLOAD comando.
- Você deve ter um bucket do Amazon S3 na mesma AWS região do seu Timestream para obter recursos. LiveAnalytics
- Para o bucket do S3 selecionado, certifique-se de que a política do bucket do S3 também tenha permissões para permitir que o Timestream exporte LiveAnalytics os dados.
- As credenciais usadas para executar a UNLOAD consulta devem ter as permissões necessárias de AWS Identity and Access Management (IAM) que permitam ao Timestream gravar os dados no S3. LiveAnalytics Um exemplo de política seria o seguinte:

```
{
    "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>/*"
            ]
        }
    ]
}
```

Para obter mais informações sobre essas permissões de gravação do S3, consulte o guia do Amazon Simple Storage Service. Se você estiver usando uma KMS chave para criptografar os dados exportados, consulte a seguir as IAM políticas adicionais necessárias.

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

Melhores práticas UNLOAD do Timestream for LiveAnalytics

A seguir estão as melhores práticas relacionadas ao UNLOAD comando.

- A quantidade de dados que pode ser exportada para o bucket do S3 usando o `UNLOAD` comando não é limitada. No entanto, o tempo limite da consulta é de 60 minutos e recomendamos exportar no máximo 60 GB de dados em uma única consulta. Se você precisar exportar mais de 60 GB de dados, divida o trabalho em várias consultas.
- Embora você possa enviar milhares de solicitações ao S3 para carregar os dados, é recomendável paralelizar as operações de gravação em vários prefixos do S3. Consulte a documentação aqui. A taxa de API chamadas do S3 pode ser reduzida quando vários leitores/gravadores acessam a mesma pasta.
- Dado o limite do tamanho da chave do S3 para definir um prefixo, recomendamos ter nomes de bucket e pasta dentro de 10 a 15 caracteres, especialmente ao usar `partitioned_by` a cláusula.
- Quando você recebe um 4XX ou 5XX para consultas contendo a `UNLOAD` instrução, é possível que resultados parciais sejam gravados no bucket do S3. O Timestream for LiveAnalytics não exclui nenhum dado do seu bucket. Antes de executar outra `UNLOAD` consulta com o mesmo destino do S3, recomendamos excluir manualmente os arquivos criados pela consulta com falha. Você pode identificar os arquivos gravados por uma consulta com falha com o correspondente`QueryExecutionId`. Para consultas com falha, o Timestream for LiveAnalytics não exporta um arquivo de manifesto para o bucket do S3.
- O Timestream for LiveAnalytics usa o upload de várias partes para exportar os resultados da consulta para o S3. Quando você recebe um 4XX ou 5XX do Timestream for LiveAnalytics para consultas contendo uma UNLOAD declaração, o Timestream for LiveAnalytics faz o melhor possível para interromper o upload de várias partes, mas é possível que algumas

partes incompletas sejam deixadas para trás. Portanto, recomendamos configurar uma limpeza automática de uploads incompletos de várias partes em seu bucket do S3 seguindo as diretrizes aqui.

Recomendações para acessar os dados em CSV formato usando o CSV analisador

- CSVos analisadores não permitem que você tenha o mesmo caractere em caracteres delimitadores, de escape e de aspas.
- Alguns CSV analisadores não conseguem interpretar tipos de dados complexos, como matrizes. Recomendamos interpretá-los por meio do desserializador. JSON

Recomendações para acessar os dados no formato Parquet

1. Se seu caso de uso exigir suporte a UTF -8 caracteres no esquema, também conhecido como nome da coluna, recomendamos usar a biblioteca Parquet-MR.
2. O timestamp em seus resultados é representado como um inteiro de 12 bytes () INT96
3. As séries temporais serão representadas como `array<row<time, value>>` outras estruturas aninhadas usarão os tipos de dados correspondentes suportados no formato Parquet

Usando a cláusula partition_by

- A coluna usada no `partitioned_by` campo deve ser a última coluna na consulta selecionada. Se mais de uma coluna for usada no `partitioned_by` campo, as colunas deverão ser as últimas colunas na consulta de seleção e na mesma ordem em que foram usadas no `partition_by` campo.
- Os valores da coluna usados para particionar os dados (`partitioned_by`campo) podem conter somente ASCII caracteres. Enquanto o Timestream for LiveAnalytics permite UTF -8 caracteres nos valores, o S3 suporta somente ASCII caracteres como chaves de objeto.

Exemplo de caso de uso UNLOAD do Timestream for LiveAnalytics

Suponha que você esteja monitorando as métricas da sessão do usuário, as fontes de tráfego e as compras de produtos do seu site de comércio eletrônico. Você está usando o Timestream LiveAnalytics para obter informações em tempo real sobre o comportamento do usuário, vendas de

produtos e realizar análises de marketing em canais de tráfego (pesquisa orgânica, mídia social, tráfego direto, campanhas pagas e outros) que direcionam os clientes ao site.

Tópicos

- Exportando os dados sem partições
- Particionamento de dados por canal
- Particionamento de dados por evento
- Particionamento de dados por canal e evento
- Arquivos de manifesto e metadados
- Usando rastreadores Glue para criar o Glue Data Catalog

Exportando os dados sem partições

Você deseja exportar os últimos dois dias de seus dados em CSV formato.

```
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')
```

Particionamento de dados por canal

Você deseja exportar os últimos dois dias de dados em CSV formato, mas gostaria de ter os dados de cada canal de tráfego em uma pasta separada. Para fazer isso, você precisa particionar os dados usando a `channel` coluna, conforme mostrado a seguir.

```
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')
```

Particionamento de dados por evento

Você deseja exportar os últimos dois dias de dados em CSV formato, mas gostaria de ter os dados de cada evento em uma pasta separada. Para fazer isso, você precisa particionar os dados usando a `event` coluna, conforme mostrado a seguir.

```
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')
```

Particionamento de dados por canal e evento

Você deseja exportar os últimos dois dias de dados em CSV formato, mas gostaria que os dados de cada canal e dentro do canal armazenassem cada evento em uma pasta separada. Para fazer isso, você precisa particionar os dados usando a `event` coluna `channel` e, conforme mostrado a seguir.

```
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')
```

Arquivos de manifesto e metadados

Arquivo manifesto

O arquivo de manifesto fornece informações sobre a lista de arquivos que são exportados com a UNLOAD execução. O arquivo de manifesto está disponível no bucket S3 fornecido com um nome de arquivo:`S3://bucket_name/<queryid>_<UUID>_manifest.json`. O arquivo de manifesto conterá o URL dos arquivos na pasta de resultados, o número de registros e o tamanho

dos respectivos arquivos e os metadados da consulta (que são o total de bytes e o total de linhas exportadas para o S3 para a consulta).

```
{
  "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"
  }
}
```

Metadados

O arquivo de metadados fornece informações adicionais sobre o conjunto de dados, como nome da coluna, tipo de coluna e esquema. <queryid>O arquivo de metadados está disponível no bucket S3 fornecido com um nome de arquivo: S3: //bucket_name/ _< >_metadata.json UUID

Veja a seguir um exemplo de um arquivo de metadados.

```
{
    "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"
  }
}
```

As informações da coluna compartilhadas no arquivo de metadados têm a mesma estrutura `ColumnInfo` enviada na API Resposta da consulta para `SELECT` consultas.

Usando rastreadores Glue para criar o Glue Data Catalog

1. Faça login em sua conta com credenciais de administrador para a seguinte validação.
2. Crie um Crawler for Glue Database usando as diretrizes fornecidas aqui. Observe que a pasta S3 a ser fornecida na fonte de dados deve ser a pasta de `UNLOAD` resultados, como. `s3://my_timestream_unloads/results`
3. Execute o rastreador seguindo as diretrizes aqui.
4. Veja a tabela Glue.
   - Vá para AWS Glue → Tables.
   - Você verá uma nova tabela criada com o prefixo de tabela fornecido ao criar o rastreador.
   - Você pode ver as informações do esquema e da partição clicando na exibição de detalhes da tabela.

A seguir estão outros AWS serviços e projetos de código aberto que usam o AWS Glue Data Catalog.

- Amazon Athena — Para obter mais informações, consulte Entendendo tabelas, bancos de dados e catálogos de dados no Guia do usuário do Amazon Athena.
- Amazon Redshift Spectrum — Para obter mais informações, consulte Consulta de dados externos usando o Amazon Redshift Spectrum no Guia do desenvolvedor do banco de dados Amazon Redshift.
- Amazon EMR — Para obter mais informações, consulte Usar políticas baseadas em recursos para EMR acesso da Amazon ao AWS Glue Data Catalog no Guia de EMR Gerenciamento da Amazon.
- AWS Cliente Glue Data Catalog para Apache Hive metastore — Para obter mais informações sobre esse projeto GitHub, consulte AWS Glue Data Catalog Client para Apache Hive Metastore.

Limites UNLOAD de Timestream para LiveAnalytics

A seguir estão os limites relacionados ao `UNLOAD` comando.

- A simultaneidade para consultas usando a `UNLOAD` instrução é de 1 consulta por segundo ()QPS. Exceder a taxa de consulta pode resultar em limitação.
- As consultas contendo instruções `UNLOAD` podem exportar no máximo 100 partições por consulta. Recomendamos verificar a contagem distinta da coluna selecionada antes de usá-la para particionar os dados exportados.

- Consultas contendo o tempo `UNLOAD` limite da declaração após 60 minutos.
- O tamanho máximo dos arquivos que a `UNLOAD` declaração cria no Amazon S3 é 78 GB.

Para outros limites do Timestream for LiveAnalytics, consulte Cotas

Usando insights de consulta para otimizar consultas no Amazon Timestream

O Query Insights é um recurso de ajuste de desempenho que ajuda você a otimizar suas consultas, melhorar seu desempenho e reduzir custos. Com os insights de consulta, você pode avaliar a eficiência de poda de partições temporais, temporais e espaciais de suas consultas. Usando insights de consulta, você também pode identificar áreas de melhoria para aprimorar o desempenho da consulta. Além disso, com os insights de consulta, você pode avaliar a eficiência com que suas consultas usam a indexação baseada em tempo e em chave de partição para otimizar a recuperação de dados. Para otimizar o desempenho da consulta, é essencial ajustar os parâmetros temporais e espaciais que governam a execução da consulta.

Tópicos

- Benefícios dos insights de consulta
- Otimizando o acesso aos dados no Amazon Timestream
- Habilitando insights de consulta no Amazon Timestream
- Otimizando consultas usando a resposta de insights de consulta

Benefícios dos insights de consulta

A seguir estão os principais benefícios do uso de insights de consulta:

- Identificação de consultas ineficientes — O Query Insights fornece informações sobre a remoção baseada em tempo e atributo das tabelas acessadas pela consulta. Essas informações ajudam a identificar as tabelas que não são acessadas de forma ideal.
- Otimizando seu modelo de dados e particionamento — Você pode usar as informações de insights de consulta para acessar e ajustar seu modelo de dados e sua estratégia de particionamento.
- Ajuste de consultas — os insights de consulta destacam oportunidades de usar índices com mais eficiência.

Otimizando o acesso aos dados no Amazon Timestream

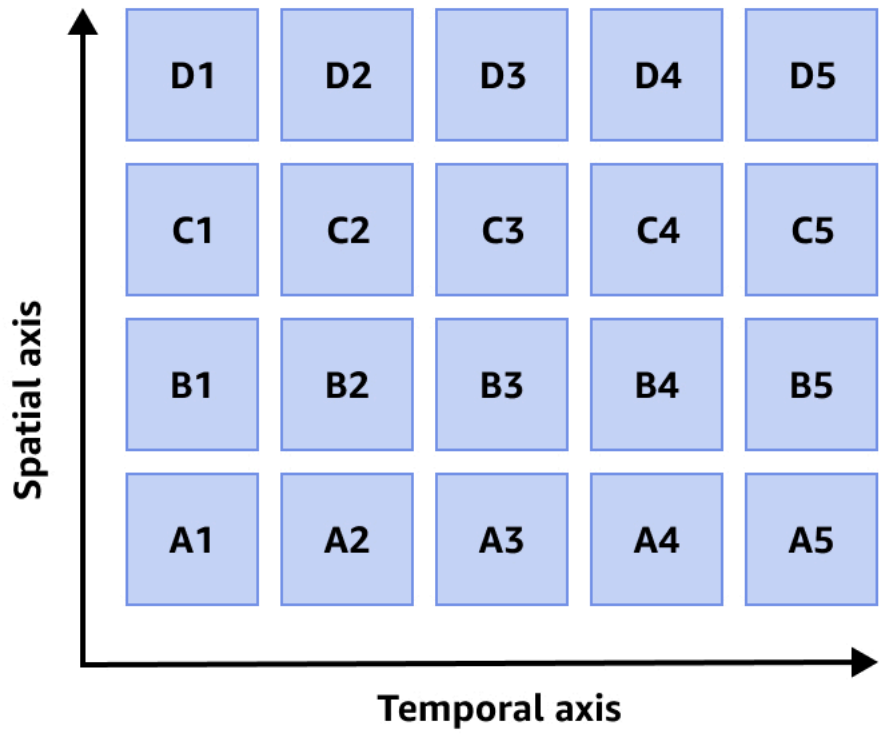
Você pode otimizar os padrões de acesso aos dados no Amazon Timestream usando o esquema de particionamento Timestream ou técnicas de organização de dados.

Tópicos

- Esquema de particionamento de Timestream
- Organização de dados

Esquema de particionamento de Timestream

O Amazon Timestream usa um esquema de particionamento altamente escalável em que cada tabela do Timestream pode ter centenas, milhares ou até milhões de partições independentes. Um serviço de indexação e rastreamento de partições altamente disponível gerencia o particionamento, minimizando o impacto das falhas e tornando o sistema mais resiliente.



Organização de dados

O Timestream armazena cada ponto de dados que ingere em uma única partição. Conforme você ingere dados em uma tabela do Timestream, o Timestream cria automaticamente partições com base nos carimbos de data e hora, na chave de partição e em outros atributos de contexto nos dados. Além de particionar os dados no tempo (particionamento temporal), o Timestream também particiona os dados com base na chave de particionamento selecionada e em outras dimensões (particionamento espacial). Essa abordagem foi projetada para distribuir o tráfego de gravação e permitir a remoção eficaz dos dados para consultas.

O recurso de insights de consulta fornece informações valiosas sobre a eficiência de remoção da consulta, que inclui cobertura espacial da consulta e cobertura temporal da consulta.

Tópicos

- QuerySpatialCoverage
- QueryTemporalCoverage

QuerySpatialCoverage

A QuerySpatialCoveragemétrica fornece informações sobre a cobertura espacial da consulta executada e a tabela com a redução espacial mais ineficiente. Essas informações podem ajudá-lo a identificar áreas de melhoria na estratégia de particionamento para aprimorar a poda espacial. O valor da `QuerySpatialCoverage` métrica varia entre 0 e 1. Quanto menor o valor da métrica, melhor será a poda da consulta no eixo espacial. Por exemplo, um valor de 0,1 indica que a consulta varre 10% do eixo espacial. Um valor de 1 indica que a consulta varre 100% do eixo espacial.

Example Usando insights de consulta para analisar a cobertura espacial de uma consulta

Digamos que você tenha um banco de dados Timestream que armazena dados meteorológicos. Suponha que a temperatura seja registrada a cada hora em estações meteorológicas localizadas em diferentes estados dos Estados Unidos. Imagine que você escolha `State` a chave de particionamento definida pelo cliente (CDPK) para particionar os dados por estado.

Suponha que você execute uma consulta para recuperar a temperatura média de todas as estações meteorológicas na Califórnia entre 14h e 16h em um dia específico. O exemplo a seguir mostra a consulta para esse cenário.

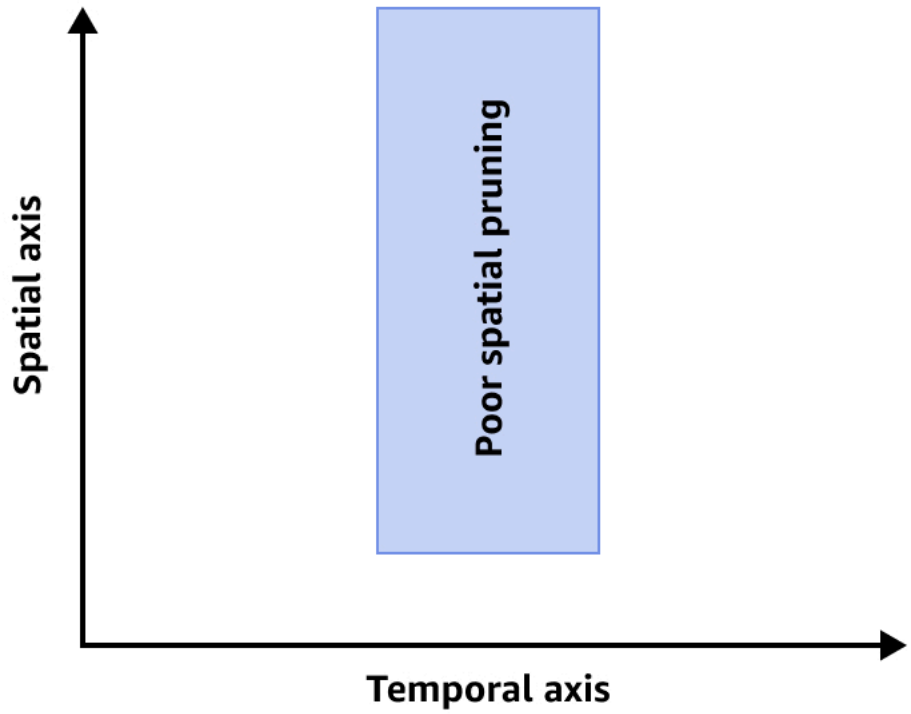
```
SELECT AVG(temperature)
```

```
FROM "weather_data"."hourly_weather"
WHERE time >= '2024-10-01 14:00:00' AND time < '2024-10-01 16:00:00'
  AND state = 'CA';
```

Usando o recurso de insights de consulta, você pode analisar a cobertura espacial da consulta. Imagine que a `QuerySpatialCoverage` métrica retorne um valor de 0,02. Isso significa que a consulta digitalizou apenas 2% do eixo espacial, o que é eficiente. Nesse caso, a consulta conseguiu reduzir efetivamente a faixa espacial, recuperando apenas dados da Califórnia e ignorando dados de outros estados.

Ao contrário, se a `QuerySpatialCoverage` métrica retornasse um valor de 0,8, isso indicaria que a consulta escaneou 80% do eixo espacial, o que é menos eficiente. Isso pode sugerir que a estratégia de particionamento precisa ser refinada para melhorar a poda espacial. Por exemplo, você pode selecionar a chave de partição como cidade ou região em vez de estado. Ao analisar a `QuerySpatialCoverage` métrica, você pode identificar oportunidades para otimizar sua estratégia de particionamento e melhorar o desempenho de suas consultas.

A imagem a seguir mostra uma poda espacial deficiente.



Para melhorar a eficiência da poda espacial, você pode fazer um ou ambos os procedimentos a seguir:

- Adicione `measure_name` a chave de particionamento padrão ou use os CDPK predicados na sua consulta.
- Se você já adicionou os atributos mencionados no ponto anterior, remova as funções em torno desses atributos ou cláusulas, como`LIKE`.

QueryTemporalCoverage

A `QueryTemporalCoverage` métrica fornece informações sobre o intervalo temporal verificado pela consulta executada, incluindo a tabela com o maior intervalo de tempo verificado. O valor da `QueryTemporalCoverage` métrica é o intervalo de tempo representado em nanossegundos. Quanto menor o valor dessa métrica, melhor será a redução da consulta no intervalo temporal. Por exemplo, uma consulta que verifica os últimos minutos de dados tem mais desempenho do que uma consulta que verifica todo o intervalo de tempo da tabela.

Example

Digamos que você tenha um banco de dados Timestream que armazena dados de sensores de IoT, com medições feitas a cada minuto em dispositivos localizados em uma fábrica. Suponha que você tenha particionado seus dados por`device_ID`.

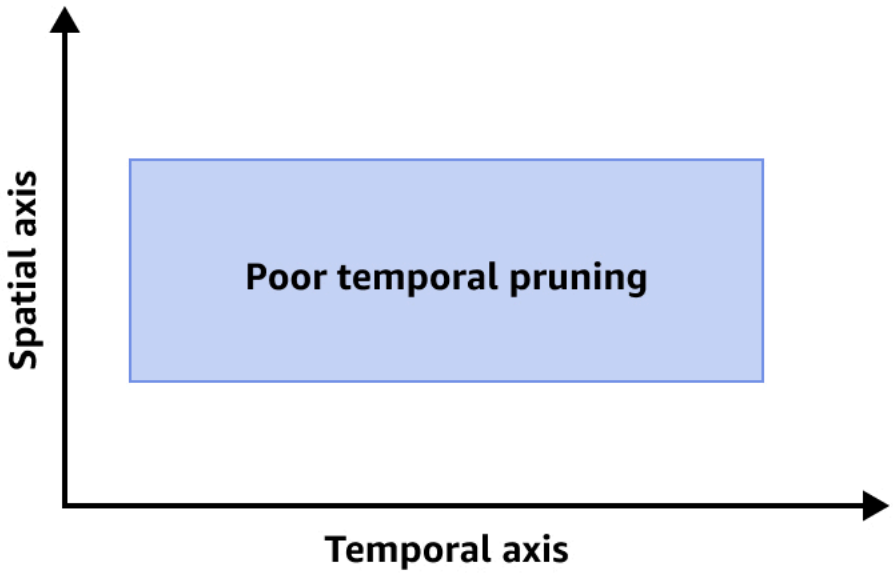
Suponha que você execute uma consulta para recuperar a leitura média do sensor de um dispositivo específico nos últimos 30 minutos. O exemplo a seguir mostra a consulta para esse cenário.

```
SELECT AVG(sensor_reading)
FROM "sensor_data"."factory_1"
WHERE device_id = 'DEV_123'
  AND time >= NOW() - INTERVAL 30 MINUTE and time < NOW();
```

Usando o recurso de insights de consulta, você pode analisar o intervalo temporal verificado pela consulta. Imagine que a `QueryTemporalCoverage` métrica retorne um valor de 1800000000000 nanossegundos (30 minutos). Isso significa que a consulta digitalizou apenas os últimos 30 minutos de dados, o que é um intervalo temporal relativamente estreito. Isso é um bom sinal porque indica que a consulta conseguiu eliminar efetivamente o particionamento temporal e recuperou somente os dados solicitados.

Por outro lado, se a `QueryTemporalCoverage` métrica retornou um valor de 1 ano em nanossegundos, isso indica que a consulta examinou um ano do intervalo de tempo na tabela, o que é menos eficiente. Isso pode sugerir que a consulta não está otimizada para remoção temporal, e você pode melhorá-la adicionando filtros de tempo.

A imagem a seguir mostra uma poda temporal deficiente.



Para melhorar a poda temporal, recomendamos que você faça um ou todos os procedimentos a seguir:

- Adicione os predicados de tempo ausentes na consulta e certifique-se de que os predicados de tempo estejam eliminando a janela de tempo desejada.
- Remova funções, como`MAX()`, por exemplo, predicados de tempo.
- Adicione predicados de tempo a todas as subconsultas. Isso é importante se suas subconsultas estiverem unindo tabelas grandes ou executando operações complexas.

Habilitando insights de consulta no Amazon Timestream

Você pode habilitar insights de consulta para suas consultas com insights fornecidos diretamente por meio da resposta da consulta. Habilitar insights de consulta não requer infraestrutura adicional nem incorre em custos adicionais. Quando você ativa os insights da consulta, ele retorna campos de metadados relacionados ao desempenho da consulta, além dos resultados da consulta, como parte

da resposta da consulta. Você pode usar essas informações para ajustar suas consultas a fim de melhorar o desempenho e reduzir o custo da consulta.

Para obter informações sobre como habilitar insights de consulta, consulteExecute uma consulta.

Para ver exemplos das respostas retornadas ao ativar os insights de consulta, consulte Exemplos de consultas agendadas.

> **Note**
>
> - Quando você ativa os insights de consulta, a taxa limita a consulta a 1 consulta por segundo (QPS). Para evitar impactos no desempenho, é altamente recomendável que você ative os insights de consulta somente durante a fase de avaliação de suas consultas, antes de implantá-las na produção.
> - Os insights fornecidos nos insights de consulta acabam sendo consistentes, o que significa que podem mudar à medida que novos dados são continuamente ingeridos nas tabelas.

Otimizando consultas usando a resposta de insights de consulta

Digamos que você esteja usando o Amazon Timestream LiveAnalytics para monitorar o consumo de energia em vários locais. Imagine que você tenha duas tabelas em seu banco de dados chamadas `raw-metrics` `aggregate-metrics` e.

A `raw-metrics` tabela armazena dados de energia detalhados no nível do dispositivo e contém as seguintes colunas:

- Timestamp
- Estado, por exemplo, Washington
- ID do dispositivo
- Consumo de energia

Os dados dessa tabela são coletados e armazenados em uma minute-by-minute granularidade. A tabela usa `State` como CDPK o.

A `aggregate-metrics` tabela armazena o resultado de uma consulta agendada para agregar os dados de consumo de energia em todos os dispositivos de hora em hora. Essa tabela contém as seguintes colunas:

- Timestamp
- Estado, por exemplo, Washington
- Consumo total de energia

A `aggregate-metrics` tabela armazena esses dados em uma granularidade horária. A tabela usa `State` como CDPK o.

Tópicos

- Consultando o consumo de energia nas últimas 24 horas
- Otimizando a consulta para intervalo temporal
- Otimizando a consulta para cobertura espacial
- Desempenho aprimorado de consultas

Consultando o consumo de energia nas últimas 24 horas

Digamos que você queira extrair a energia total consumida em Washington nas últimas 24 horas. Para encontrar esses dados, você pode aproveitar os pontos fortes de ambas as tabelas: `raw-metrics` e. `aggregate-metrics` A `aggregate-metrics` tabela fornece dados de consumo de energia por hora nas últimas 23 horas, enquanto a `raw-metrics` tabela oferece dados granulares em minutos para a última hora. Ao consultar as duas tabelas, você pode obter uma visão completa e precisa do consumo de energia em Washington nas últimas 24 horas.

```
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()
```

Esse exemplo de consulta é fornecido apenas para fins ilustrativos e pode não funcionar como está. O objetivo é demonstrar o conceito, mas talvez seja necessário modificá-lo para se adequar ao seu caso de uso ou ambiente específico.

Depois de executar essa consulta, você pode perceber que o tempo de resposta da consulta está mais lento do que o esperado. Para identificar a causa raiz desse problema de desempenho, você pode usar o recurso de insights de consulta para analisar o desempenho da consulta e otimizar sua execução.

O exemplo a seguir mostra a resposta dos insights da consulta.

```
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
```

A resposta do query insights fornece as seguintes informações:

- Intervalo temporal: a consulta examinou um intervalo temporal excessivo de 365 dias para a tabela. `aggregate-metrics` Isso indica um uso ineficiente da filtragem temporal.
- Cobertura espacial: a consulta examinou toda a faixa espacial (100%) da `raw-metrics` tabela. Isso sugere que a filtragem espacial não está sendo utilizada de forma eficaz.

Se sua consulta acessar mais de uma tabela, os insights de consulta fornecerão as métricas da tabela com o padrão de acesso mais abaixo do ideal.

Otimizando a consulta para intervalo temporal

Com base na resposta dos insights da consulta, você pode otimizar a consulta para o intervalo temporal, conforme mostrado no exemplo a seguir.

```
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'
```

Se você executar o `QueryInsights` comando novamente, ele retornará a seguinte resposta.

```
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
```

Essa resposta mostra que a cobertura espacial da `aggregate-metrics` tabela ainda é de 100%, o que é ineficiente. A seção a seguir mostra como otimizar a consulta para cobertura espacial.

Otimizando a consulta para cobertura espacial

Com base na resposta dos insights da consulta, você pode otimizar a consulta para cobertura espacial, conforme mostrado no exemplo a seguir.

```
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'
```

Se você executar o `QueryInsights` comando novamente, ele retornará a seguinte resposta.

```
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
```

Desempenho aprimorado de consultas

Depois de otimizar a consulta, os insights de consulta fornecem as seguintes informações:

- A poda temporal da `aggregate-metrics` mesa é de 23 horas. Isso indica que somente 23 horas do intervalo temporal são escaneadas.
- A poda espacial da `aggregate-metrics` mesa é de 0,02. Isso indica que apenas 2% dos dados do intervalo espacial da tabela estão sendo digitalizados. A consulta verifica uma parte muito pequena das tabelas, levando a um desempenho rápido e à redução da utilização de recursos. A eficiência aprimorada da remoção indica que a consulta agora está otimizada para desempenho.

Trabalhando com AWS Backup

A funcionalidade de proteção de dados no Amazon Timestream LiveAnalytics for é uma solução totalmente gerenciada para ajudar você a atender aos requisitos de conformidade regulatória e continuidade de negócios. A funcionalidade é ativada por meio da integração nativa com AWS Backup um serviço de backup unificado projetado para simplificar a criação, migração, restauração e exclusão de backups, ao mesmo tempo em que fornece relatórios e auditoria aprimorados.
Por meio da integração com AWS Backup, você pode usar uma solução de proteção de dados centralizada totalmente gerenciada e orientada por políticas para criar backups imutáveis e gerenciar centralmente a proteção dos dados de seu aplicativo, abrangendo o Timestream e outros serviços suportados pela. AWS AWS Backup

Para usar a funcionalidade, você deve optar por permitir AWS Backup a proteção de seus recursos do Timestream. As opções de aceitação se aplicam à conta e à AWS região específicas, portanto, talvez você precise optar por várias regiões usando a mesma conta. Para obter mais informações sobre AWS Backup, consulte o Guia do AWS Backup desenvolvedor.

A funcionalidade de proteção de dados disponível por meio AWS Backup inclui o seguinte.

Backups agendados — você pode configurar backups regulares do seu Timestream para LiveAnalytics tabelas usando planos de backup.

Cópia entre contas e entre regiões — você pode copiar automaticamente seus backups para outro cofre de backup em uma AWS região ou conta diferente, o que permite que você atenda aos seus requisitos de proteção de dados.

Classificação por níveis de armazenamento frio — você pode configurar seus backups para implementar regras de ciclo de vida para excluir ou fazer a transição de backups para um armazenamento mais frio. Isso pode ajudar você a otimizar seus custos de backup.

Tags — Você pode marcar automaticamente seus backups para fins de cobrança e alocação de custos.

Criptografia — Seus dados de backup são armazenados no AWS Backup cofre. Isso permite que você criptografe e proteja seus backups usando uma AWS KMS chave independente do Timestream para a chave de criptografia de LiveAnalytics tabelas.

Backups seguros usando o WORM modelo — Você pode usar o AWS Backup Vault Lock para ativar a configuração write-once-read-many (WORM) para seus backups. Com o AWS Backup Vault Lock,

você pode adicionar uma camada adicional de defesa que protege os backups contra operações de exclusão inadvertidas ou mal-intencionadas, alterações nos períodos de retenção de backup e atualizações nas configurações do ciclo de vida. Saiba mais: AWS Backup Vault Lock.

A funcionalidade de proteção de dados está disponível em todas as regiões. Para saber mais sobre a funcionalidade, consulte o Guia do AWS Backup desenvolvedor.

Backup e restauração de tabelas do Timestream: como funciona

Você pode criar backups de suas tabelas do Amazon Timestream. Esta seção oferece uma visão geral do que ocorre durante o processo de backup e restauração.

Tópicos

- Backups
- Restaurações

Backups

Você pode usar o recurso de backup sob demanda para criar backups completos do Amazon LiveAnalytics Timestream para tabelas. Esta seção oferece uma visão geral do que ocorre durante o processo de backup e restauração.

Você pode criar um backup dos dados do Timestream em uma granularidade de tabela. Você pode iniciar um backup da tabela selecionada usando o console Timestream, o console, ou AWS Backup . SDK CLI O backup é criado de forma assíncrona e todos os dados na tabela até o horário de início do backup são incluídos no backup. No entanto, existe a possibilidade de que alguns dos dados ingeridos na tabela enquanto o backup está em andamento também possam ser incluídos no backup. Para proteger seus dados, você pode criar um backup único sob demanda ou agendar um backup recorrente da sua tabela.

Enquanto um backup estiver em andamento, você não poderá fazer o seguinte.

- Pausar ou cancelar a operação de backup.
- Excluir a tabela de origem do backup.
- Desativar backups em uma tabela se houver um backup em andamento para essa tabela.

Depois de configurado, AWS Backup fornece agendamentos automatizados de backup, gerenciamento de retenção e gerenciamento do ciclo de vida, eliminando a necessidade de scripts

personalizados e processos manuais. Para obter mais informações, consulte o Guia do AWS Backup desenvolvedor

Todo o Timestream para LiveAnalytics backups é de natureza incremental, o que significa que o primeiro backup de uma tabela é um backup completo e cada backup subsequente da mesma tabela é um backup incremental, copiando somente as alterações nos dados desde o último backup. Como os dados no Timestream LiveAnalytics for são armazenados em uma coleção de partições, todas as partições que foram alteradas devido à ingestão de novos dados ou às atualizações dos dados existentes desde o último backup são copiadas durante os backups subsequentes.

Se você estiver usando o Timestream para LiveAnalytics console, os backups criados para todos os recursos da conta serão listados na guia Backups. Além disso, os backups também estão listados nos detalhes da tabela.

Restaurações

Você pode restaurar uma tabela do Timestream para LiveAnalytics consoleSDK, AWS Backup console ou. AWS CLI Você pode restaurar todos os dados do seu backup ou definir as configurações de retenção da tabela para restaurar dados selecionados. Ao iniciar uma restauração, você pode definir as seguintes configurações da tabela.

- Database Name
- Nome da tabela
- Retenção do armazenamento de memória
- Retenção magnética da loja
- Ativar gravações de armazenamento magnético
- Localização dos registros de erros do S3 (opcional)
- IAMfunção que AWS Backup assumirá ao restaurar o backup

As configurações anteriores são independentes da tabela de origem. Para restaurar todos os dados em seu backup, recomendamos que você defina as novas configurações da tabela de forma que a soma do período de retenção do armazenamento de memória e do período de retenção do armazenamento magnético seja maior do que a diferença entre o carimbo de data/hora mais antigo e o atual. Quando você seleciona um backup incremental para restauração, todos os dados (incrementais + dados completos subjacentes) são restaurados. Após a restauração bem-sucedida, a tabela fica ativa e você pode realizar operações de ingestão e/ou consulta na tabela restaurada.

No entanto, você não pode realizar essas operações enquanto a restauração estiver em andamento. Depois de restaurada, a tabela é semelhante a qualquer outra tabela em sua conta.

Example Restaure todos os dados de um backup

Este exemplo tem as seguintes suposições.

Carimbo de data/hora mais antigo — `August 1, 2021 0:00:00`

- Agora — `November 9, 2022 0:00:00`

Para restaurar todos os dados de um backup, insira e compare os valores da seguinte forma.

1. Insira Retenção de armazenamento de memória e Retenção de armazenamento magnético. Por exemplo, suponha esses valores.
   - Retenção do armazenamento de memória — 12 horas
   - Retenção magnética da loja — 500 dias
2. Encontre a soma de retenção de armazenamento de memória e retenção de armazenamento magnético.

```
12 hours + (500 * 24 hours) =
12 hours + 12,000 hours =
12,012 hours
```

3. Descubra a diferença entre o carimbo de data/hora mais antigo e agora.

```
November 9, 2022 0:00:00 - August 1, 2021 0:00:00 =
465 days =
465 * 24 hours =
11,160 hours
```

4. Certifique-se de que a soma dos valores de retenção na segunda etapa seja maior do que a diferença de tempos na terceira etapa. Ajuste os tempos de retenção, se necessário.

```
12,012 > 11,160
true
```

Example Restaurar dados selecionados de um backup

Este exemplo tem a seguinte suposição.

- Agora — `November 9, 2022 0:00:00`

Para restaurar somente dados selecionados de um backup, insira e compare valores da seguinte forma.

1. Determine o primeiro carimbo de data/hora necessário. Por exemplo, suponha`December 4, 2021 0:00:00`.
2. Descubra a diferença entre o primeiro registro de data e hora necessário e agora.

```
November 9, 2022 0:00:00 - December 4, 2021 0:00:00 =
340 days =
340 * 24 hours =
8,160 hours
```

3. Insira o valor desejado para Retenção do armazenamento de memória. Por exemplo, insira 12 horas.
4. Subtraia o valor da diferença na segunda etapa.

```
8,160 hours - 12 hours =
8148 hours
```

5. Insira esse valor para Retenção magnética da loja.

Você pode copiar um backup do seu Timestream para dados da LiveAnalytics tabela em uma AWS região diferente e depois restaurá-lo nessa nova região. Você pode copiar e depois restaurar backups entre regiões AWS comerciais e regiões AWS GovCloud (EUA). Pague somente pelos dados transferidos para fora da região de origem e pelos dados restaurados na nova tabela na região de destino.

Depois que a tabela for restaurada, você deverá configurar manualmente o seguinte na tabela restaurada.

- AWS Políticas de Identity and Access Management (IAM)
- Tags
- Consultas agendadas

Os tempos de restauração estão diretamente relacionados à configuração de suas tabelas. Isso inclui o tamanho das tabelas, o número de partições subjacentes, a quantidade de dados restaurados no armazenamento de memória e outras variáveis. Uma prática recomendada ao planejar a recuperação de desastres é documentar regularmente os tempos médios de conclusão da restauração e estabelecer como esses tempos afetam seu objetivo geral de tempo de recuperação (RTO).

Todas as API ações e consoles de backup e restauração são capturados e registrados AWS CloudTrail para registro, monitoramento contínuo e auditoria.

Criação de backups das tabelas do Amazon Timestream

Esta seção descreve como habilitar AWS Backup e criar backups sob demanda e programados para o Amazon Timestream.

Tópicos

- Permitindo AWS Backup proteger o Timestream para dados LiveAnalytics
- Criar backups sob demanda
- Backups agendados

Permitindo AWS Backup proteger o Timestream para dados LiveAnalytics

Você deve habilitá-lo AWS Backup para usá-lo com o Timestream for. LiveAnalytics

Para ativar AWS Backup o Timestream para LiveAnalytics console, execute as etapas a seguir.

1. Faça login no Console de gerenciamento da AWS.
2. Um banner pop-up aparece na parte superior da página Timestream for LiveAnalytics Dashboard para permitir o suporte ao Timestream AWS Backup para dados. LiveAnalytics Caso contrário, no painel de navegação, escolha Backups.
3. Na janela Backup, você verá o banner para ativar AWS Backup. Escolha Habilitar.

   A proteção de dados por meio de agora AWS Backup está disponível para seu Timestream para LiveAnalytics tabelas.

Para habilitar AWS Backup, consulte a AWS Backup documentação para habilitar via console e programaticamente.

Se você optar por desativar a proteção AWS Backup do Timestream para LiveAnalytics dados após a ativação, faça login pelo AWS Backup console e mova o botão para a esquerda.

Se você não conseguir ativar ou desativar os AWS Backup recursos, talvez seu AWS administrador precise realizar essas ações.

Criar backups sob demanda

Para criar um backup sob demanda de um Timestream para LiveAnalytics tabela, siga estas etapas.

1. Faça login no Console de gerenciamento da AWS.
2. No painel de navegação, no lado esquerdo do console, selecione Backups.
3. Escolha Criar backup sob demanda.
4. Continue selecionando as configurações na janela de backup.
5. Você pode criar um backup agora, iniciar um backup imediatamente ou selecionar uma janela de backup para iniciar o backup.
6. Selecione a política de gerenciamento do ciclo de vida do seu backup. Você pode fazer a transição dos dados de backup para um armazenamento frio, onde precisa reter o backup por no mínimo 90 dias. Você pode definir o período de retenção necessário para seu backup. Você pode selecionar um cofre existente ou selecionar criar um novo cofre de backup para navegar até o AWS Backup console e criar um novo cofre de backup. <documentation link on creating a new backup vault here>
7. Selecione a IAM função apropriada.
8. Se você deseja atribuir uma ou mais tags ao seu backup sob demanda, insira uma chave e um valor opcional, e escolha Adicionar tag.
9. Escolha criar um backup sob demanda. Isso leva você à página Backup, onde você verá uma lista de trabalhos.
10. Escolha o ID do trabalho de backup do recurso para o qual você optou por fazer backup para ver os detalhes desse trabalho.

Backups agendados

Para agendar um backup, consulte Criar um backup agendado.

Restaurando um backup de uma tabela do Amazon Timestream

Esta seção descreve como restaurar um backup de uma tabela do Amazon Timestream.

Tópicos

- Restaurando um Timestream para uma tabela a partir de LiveAnalytics AWS Backup
- Restaurando um Timestream de uma LiveAnalytics tabela em outra região ou conta

Restaurando um Timestream para uma tabela a partir de LiveAnalytics AWS Backup

Para restaurar o Timestream da LiveAnalytics tabela AWS Backup usando o Timestream para LiveAnalytics console, siga estas etapas.

1. Faça login no Console de gerenciamento da AWS.
2. No painel de navegação, no lado esquerdo do console, selecione Backups.
3. Para restaurar um recurso, escolha o botão de rádio ao lado da ID do ponto de recuperação do recurso. No canto superior direito do painel, escolha Restaurar.
4. Insira as configurações da tabela, ou seja, Nome do banco de dados e Nome da tabela. Observe que o nome da tabela restaurada deve ser diferente do nome da tabela de origem original.
5. Defina as configurações de retenção de memória e armazenamento magnético.
6. Em Função de restauração, escolha a IAM função que AWS Backup será assumida para essa restauração.
7. Escolha Restaurar backup. Uma mensagem na parte superior da página fornece informações sobre o trabalho de restauração.

> **Note**
>
> Você é cobrado pela restauração de todo o backup, independentemente da memória configurada e dos períodos de retenção do armazenamento magnético. No entanto, quando a restauração for concluída, sua tabela restaurada conterá somente os dados dentro dos períodos de retenção configurados.

Restaurando um Timestream de uma LiveAnalytics tabela em outra região ou conta

Para restaurar um Timestream da LiveAnalytics tabela em outra região ou conta, primeiro você precisará copiar o backup para essa nova região ou conta. Para copiar para outra conta, essa conta deve primeiro conceder permissão. Depois de copiar seu Timestream para LiveAnalytics backup na nova região ou conta, ele pode ser restaurado com o processo na seção anterior.

Copiar um backup de uma tabela do Amazon Timestream

Você pode fazer uma cópia de um backup atual. Você pode copiar backups para várias AWS contas ou AWS regiões sob demanda ou automaticamente como parte de um plano de backup agendado. A replicação entre regiões é particularmente vantajosa se você tiver requisitos de continuidade de negócios ou de conformidade para armazenar backups a uma distância mínima dos dados de produção.

Backups entre contas são úteis para copiar com segurança seus backups para uma ou mais contas do AWS na sua organização por motivos operacionais ou de segurança. Se o backup original for excluído acidentalmente, você poderá copiar o backup da conta de destino para a conta de origem e, em seguida, iniciar a restauração. Antes de fazer isso, você deve ter duas contas que pertençam à mesma organização no serviço Organizations e as permissões necessárias para as contas. Quando você copia um backup incremental em outra conta ou região, o backup completo associado também é copiado.

As cópias herdam a configuração do backup de origem, a menos que você especifique o contrário. Há uma exceção. Se você especificar sua nova cópia como “Nunca”, expire. Com essa configuração, a nova cópia ainda herda a data de validade da fonte. Se você quiser que sua nova cópia de backup seja permanente, defina seus backups de origem para nunca expirar ou então determine que sua nova cópia expirará 100 anos após a criação.

Para copiar um backup do console Timestream, siga estas etapas.

1. Faça login no Console de gerenciamento da AWS.
2. No painel de navegação, no lado esquerdo do console, selecione Backups.
3. Escolha o botão de rádio ao lado do ID do ponto de recuperação do recurso. No canto superior direito do painel, selecione Ações e escolha Copiar.
4. Selecione Continuar com o AWS Backup e siga as etapas para o backup entre contas.

Atualmente, a cópia de backups sob demanda e programados entre contas e regiões não é suportada nativamente no Timestream for LiveAnalytics console e você precisa navegar até ela para realizar a AWS Backup operação.

Excluir backups

Esta seção descreve como excluir um backup de um Timestream para LiveAnalytics tabela.

Para excluir um backup do console Timestream, siga estas etapas.

1. Faça login no Console de gerenciamento da AWS.
2. No painel de navegação, no lado esquerdo do console, selecione Backups.
3. Escolha o botão de rádio ao lado do ID do ponto de recuperação do recurso. No canto superior direito do painel, selecione Ações e escolha Excluir.
4. Selecione Continuar com o AWS Backup e siga as etapas para excluir backups em Excluindo backups.

> **Note**
>
> Quando você exclui um backup incremental, somente o backup incremental é excluído e o backup completo subjacente não é excluído.

Cota e limites

AWS Backup limita os backups a um backup simultâneo por recurso. Portanto, solicitações adicionais de backup agendadas ou sob demanda para o recurso estão na fila e serão iniciadas somente após a conclusão da tarefa de backup existente. Se a tarefa de backup não for iniciada ou concluída dentro da janela de backup, a solicitação falhará. Para obter mais informações sobre AWS Backup limites, consulte Limites de AWS backup no Guia do desenvolvedor de AWS backup.

Ao criar um backup, você pode executar até quatro backups simultâneos por conta. Da mesma forma, você pode executar uma restauração simultânea por conta. Quando você inicia mais de quatro tarefas de backup simultaneamente, somente quatro tarefas de backup são iniciadas e as tarefas restantes serão repetidas periodicamente. Depois de iniciada, se a tarefa de backup não for concluída dentro da duração da janela de backup configurada, a tarefa de backup falhará. Se a tarefa de backup com falha for um backup sob demanda, você poderá repetir o backup e, para backups agendados, a tarefa será executada na programação a seguir.

Chaves de partição definidas pelo cliente

O Amazon Timestream LiveAnalytics para chaves de partição definidas pelo cliente é um recurso do Timestream que permite que os clientes definam suas LiveAnalytics próprias chaves de partição para suas tabelas. O particionamento é uma técnica usada para distribuir dados em várias unidades físicas de armazenamento, permitindo uma recuperação de dados mais rápida e eficiente. Com as

chaves de partição definidas pelo cliente, os clientes podem criar um esquema de particionamento que se alinhe melhor com seus padrões de consulta e casos de uso.

Com o Timestream para chaves de partição LiveAnalytics definidas pelo cliente, os clientes podem escolher um nome de dimensão como chave de partição para suas tabelas. Isso permite mais flexibilidade na definição do esquema de particionamento para seus dados. Ao selecionar a chave de partição correta, os clientes podem otimizar seu modelo de dados, melhorando o desempenho da consulta e reduzindo a latência da consulta.

Tópicos

- Usando chaves de partição definidas pelo cliente
- Introdução às chaves de partição definidas pelo cliente
- Verificando a configuração do esquema de particionamento
- Atualizando a configuração do esquema de particionamento
- Vantagens das chaves de partição definidas pelo cliente
- Limitações das chaves de partição definidas pelo cliente
- Chaves de partição definidas pelo cliente e dimensões de baixa cardinalidade
- Criação de chaves de partição para tabelas existentes
- Cronograma para validação de LiveAnalytics esquema com chaves de partição compostas personalizadas

Usando chaves de partição definidas pelo cliente

Se você tiver um padrão de consulta bem definido com dimensões de alta cardinalidade e precisar de baixa latência de consulta, um Timestream para uma chave de partição LiveAnalytics definida pelo cliente pode ser uma ferramenta útil para aprimorar seu modelo de dados. Por exemplo, se você for uma empresa de varejo que acompanha as interações dos clientes em seu site, os principais padrões de acesso provavelmente seriam por ID do cliente e registro de data e hora. Ao definir a ID do cliente como a chave de partição, seus dados podem ser distribuídos uniformemente, permitindo uma latência reduzida e, em última análise, melhorando a experiência do usuário.

Outro exemplo está no setor de saúde, onde dispositivos vestíveis coletam dados de sensores para rastrear os sinais vitais dos pacientes. O principal padrão de acesso seria por ID do dispositivo e registro de data e hora, com alta cardinalidade em ambas as dimensões. Ao definir o ID do dispositivo como a chave de partição, você pode otimizar a execução da consulta e garantir um desempenho sustentado da consulta a longo prazo.

Em resumo, o Timestream para chaves de partição LiveAnalytics definidas pelo cliente é mais útil quando você tem um padrão de consulta claro, dimensões de alta cardinalidade e precisa de baixa latência para suas consultas. Ao definir uma chave de partição que se alinha ao seu padrão de consulta, você pode otimizar a execução da consulta e garantir um desempenho de consulta sustentado a longo prazo.

Introdução às chaves de partição definidas pelo cliente

No console, escolha Tabelas e crie uma nova tabela. Você também pode usar um SDK para acessar a `CreateTable` ação para criar novas tabelas que podem incluir uma chave de partição definida pelo cliente.

Crie uma tabela com uma chave de partição do tipo dimensão

Você pode usar os seguintes trechos de código para criar uma tabela com uma chave de partição do tipo dimensão.

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)
```

Verificando a configuração do esquema de particionamento

Você pode verificar a configuração de uma tabela para o esquema de particionamento de algumas maneiras. No console, escolha Bancos de dados e escolha a tabela a ser verificada. Você também pode usar um SDK para acessar a `DescribeTable` ação.

Descrever uma tabela com uma chave de partição

Você pode usar os seguintes trechos de código para descrever uma tabela com uma chave de partição.

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;
        }
    }
```

Veja a seguir um exemplo de saída.

1. A tabela tem uma chave de partição do tipo de dimensão

```
[{Type: DIMENSION,Name: hostId,EnforcementInRecord: OPTIONAL}]
```

2. A tabela tem nome, tipo de medida, chave de partição

```
[{Type: MEASURE,}]
```

3. Obtendo a chave de partição composta de uma tabela criada sem especificar a chave de partição composta

```
[{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;
        }
    }
```

Veja a seguir um exemplo de saída.

1. A tabela tem uma chave de partição do tipo de dimensão

```
[PartitionKey(Type=DIMENSION, Name=hostId, EnforcementInRecord=OPTIONAL)]
```

2. A tabela tem nome, tipo de medida, chave de partição

```
[PartitionKey(Type=MEASURE)]
```

3. Obter a chave de partição composta de uma tabela criada sem especificar a chave de partição composta retornará

```
[PartitionKey(Type=MEASURE)]
```

Go v1

```
    <tablistentry>
     <tabname> Go </tabname>
     <tabcontent>
      <programlisting language="go"></programlisting>
     </tabcontent>
    </tablistentry>
```

Veja a seguir um exemplo de saída.

```
{
  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
    }
```

Veja a seguir um exemplo de saída.

```
{
  "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)
```

Veja a seguir um exemplo de saída.

1. A tabela tem uma chave de partição do tipo de dimensão

```
[{'CompositePartitionKey': [{'Type': 'DIMENSION', 'Name': 'hostId',
 'EnforcementInRecord': 'OPTIONAL'}]}]
```

2. A tabela tem nome, tipo de medida, chave de partição

```
[{'CompositePartitionKey': [{'Type': 'MEASURE'}]}]
```

3. Obtendo a chave de partição composta de uma tabela criada sem especificar a chave de partição composta

```
[{'CompositePartitionKey': [{'Type': 'MEASURE'}]}]
```

Atualizando a configuração do esquema de particionamento

Você pode atualizar a configuração da tabela para o esquema de particionamento com um SDK com acesso à ação. `UpdateTable`

Atualizar uma tabela com uma chave de partição

Você pode usar os seguintes trechos de código para atualizar uma tabela com uma chave de partição.

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)
```

Vantagens das chaves de partição definidas pelo cliente

Desempenho aprimorado da consulta: as chaves de partição definidas pelo cliente permitem que você otimize a execução da consulta e melhore o desempenho geral da consulta. Ao definir chaves de partição que se alinham aos seus padrões de consulta, você pode minimizar a varredura de dados e otimizar a remoção de dados, resultando em menor latência de consulta.

Melhor previsibilidade do desempenho a longo prazo: as chaves de partição definidas pelo cliente permitem que os clientes distribuam os dados uniformemente entre as partições, melhorando a eficiência do gerenciamento de dados. Isso garantirá que o desempenho da consulta permaneça estável à medida que seus dados armazenados aumentam com o tempo.

Limitações das chaves de partição definidas pelo cliente

Como Timestream para o LiveAnalytics usuário, é importante ter em mente as limitações relacionadas à chave de partição de um cliente. Em primeiro lugar, é necessário um bom entendimento da carga de trabalho e dos padrões de consulta. Isso significa que você deve ter uma ideia clara de quais dimensões são usadas com mais frequência como principais condições de filtragem em consultas e têm alta cardinalidade para fazer o uso mais eficaz das chaves de partição.

Em segundo lugar, as chaves de partição precisam ser definidas no momento da criação da tabela e não podem ser adicionadas às tabelas existentes. Isso significa que você deve considerar cuidadosamente sua estratégia de particionamento antes de criar uma tabela para garantir que ela esteja alinhada às necessidades da sua empresa.

Por fim, é importante observar que, depois que a tabela for criada, você não poderá alterar a chave de partição posteriormente. Isso significa que você deve testar e avaliar minuciosamente sua estratégia de particionamento antes de se comprometer com ela. Com essas limitações em mente, a chave de partição definida pelo cliente do Timestream pode melhorar muito o desempenho da consulta e a satisfação a longo prazo.

Chaves de partição definidas pelo cliente e dimensões de baixa cardinalidade

Se você decidir usar uma chave de partição com cardinalidade muito baixa, como uma região ou estado específico, é importante observar que os dados de outras entidades`customerID`, como,, e outras`ProductCategory`, podem acabar espalhados por muitas partições, às vezes com poucos ou nenhum dado presente. Isso pode levar à execução ineficiente de consultas e à diminuição do desempenho.

Para evitar isso, recomendamos que você escolha dimensões que não sejam apenas parte da condição de filtragem da chave, mas tenham maior cardinalidade. Isso ajudará a garantir que os dados sejam distribuídos uniformemente pelas partições e melhorará o desempenho da consulta.

Criação de chaves de partição para tabelas existentes

Se você já tiver tabelas no Timestream para LiveAnalytics e quiser usar chaves de partição definidas pelo cliente, precisará migrar seus dados para uma nova tabela com a definição de esquema de particionamento desejada. Isso pode ser feito usando a exportação para o S3 e o carregamento em lote juntos, o que envolve exportar os dados da tabela existente para o S3, modificar os dados para incluir a chave de partição (se necessário) e adicionar cabeçalhos aos seus CSV arquivos e, em seguida, importar os dados para uma nova tabela com o esquema de particionamento desejado definido. Lembre-se de que esse método pode ser demorado e caro, especialmente para mesas grandes.

Como alternativa, você pode usar consultas agendadas para migrar seus dados para uma nova tabela com o esquema de particionamento desejado. Esse método envolve a criação de uma consulta agendada que lê a tabela existente e grava na nova tabela. A consulta agendada pode ser configurada para ser executada regularmente até que todos os dados tenham sido migrados. Lembre-se de que você será cobrado pela leitura e gravação dos dados durante o processo de migração.

Cronograma para validação de LiveAnalytics esquema com chaves de partição compostas personalizadas

A validação do esquema no Timestream for LiveAnalytics ajuda a garantir que os dados ingeridos no banco de dados estejam em conformidade com o esquema especificado, minimizando os erros de ingestão e melhorando a qualidade dos dados. Em particular, a validação do esquema é especialmente útil ao adotar uma chave de partição definida pelo cliente com o objetivo de otimizar o desempenho da consulta.

O que é Timestream para validação de LiveAnalytics esquema com chaves de partição definidas pelo cliente?

O Timestream para validação do LiveAnalytics esquema é um recurso que valida os dados que estão sendo ingeridos em um Timestream for LiveAnalytics table com base em um esquema predefinido. Esse esquema define o modelo de dados, incluindo chave de partição, tipos de dados e restrições para os registros que estão sendo inseridos.

Ao usar uma chave de partição definida pelo cliente, a validação do esquema se torna ainda mais crucial. As chaves de partição permitem que você especifique uma chave de partição, que determina como seus dados são armazenados no Timestream for. LiveAnalytics Ao validar os dados recebidos em relação ao esquema com uma chave de partição personalizada, você pode impor a consistência dos dados, detectar erros com antecedência e melhorar a qualidade geral dos dados armazenados no Timestream for. LiveAnalytics

Como usar o Timestream para validação de LiveAnalytics esquema com chaves de partição compostas personalizadas

Para usar o Timestream para validação de LiveAnalytics esquema com chaves de partição compostas personalizadas, siga estas etapas:

Pense em como serão seus padrões de consulta: para escolher e definir adequadamente o esquema do seu Timestream para a LiveAnalytics tabela, você deve começar com os requisitos de consulta.

Especifique chaves de partição compostas personalizadas: ao criar a tabela, especifique uma chave de partição personalizada. Essa chave determina o atributo que será usado para particionar os dados da tabela. Você pode escolher entre teclas de dimensão e teclas de medida para particionamento. Uma chave de dimensão particiona os dados com base no nome de uma dimensão, enquanto uma chave de medida particiona os dados com base no nome da medida.

Defina níveis de imposição: para garantir o particionamento de dados adequado e os benefícios que vêm com ele, o Amazon Timestream LiveAnalytics for permite que você defina níveis de imposição para cada chave de partição em seu esquema. O nível de imposição determina se a dimensão da chave de partição é necessária ou opcional ao ingerir registros. Você pode escolher entre duas opções:`REQUIRED`, o que significa que a chave de partição deve estar presente no registro ingerido e`OPTIONAL`, o que significa que a chave de partição não precisa estar presente. É recomendável usar o nível de `REQUIRED` imposição ao usar uma partição definida pelo cliente para garantir que seus dados sejam particionados adequadamente e que você obtenha todos os benefícios desse recurso. Além disso, você pode alterar a configuração do nível de imposição a qualquer momento após a criação do esquema para se ajustar aos requisitos de ingestão de dados.

Ingerir dados: ao ingerir dados no Timestream for LiveAnalytics table, o processo de validação do esquema verificará os registros em relação ao esquema definido com chaves de partição compostas personalizadas. Se os registros não seguirem o esquema, o Timestream for LiveAnalytics retornará um erro de validação.

Lidar com erros de validação: em caso de erros de validação, o Timestream for LiveAnalytics retornará a `ValidationException` ou a`RejectedRecordsException`, dependendo do tipo de erro. Certifique-se de lidar com essas exceções em seu aplicativo e tomar as medidas apropriadas, como corrigir os registros incorretos e tentar novamente a ingestão.

Atualizar níveis de imposição: se necessário, você pode atualizar o nível de imposição das chaves de partição após a criação da tabela usando a `UpdateTable` ação. No entanto, é importante observar que alguns aspectos da configuração da chave de partição, como nome e tipo, não podem ser alterados após a criação da tabela. Se você alterar o nível de imposição de `REQUIRED` para`OPTIONAL`, todos os registros serão aceitos independentemente da presença do atributo selecionado como a chave de partição definida pelo cliente. Por outro lado, se você alterar o nível de fiscalização de `OPTIONAL` para`REQUIRED`, poderá começar a ver erros de gravação 4xx em registros que não atendem a essa condição. Portanto, é essencial escolher o nível de fiscalização adequado para seu caso de uso ao criar sua tabela, com base nos requisitos de particionamento de seus dados.

Quando usar o Timestream para validação de LiveAnalytics esquema com chaves de partição compostas personalizadas

O cronograma para validação do LiveAnalytics esquema com chaves de partição compostas personalizadas deve ser usado em cenários em que a consistência, a qualidade e o particionamento otimizado dos dados são cruciais. Ao aplicar um esquema durante a ingestão de dados, você pode evitar erros e inconsistências que podem levar à análise incorreta ou à perda de informações valiosas.

Interação com trabalhos de carregamento em lote

Ao configurar um trabalho de carregamento em lote para importar dados em uma tabela com uma chave de partição definida pelo cliente, há alguns cenários que podem afetar o processo:

1. Se o nível de imposição estiver definido como`OPTIONAL`, um alerta será exibido no console durante o fluxo de criação se a chave de partição não for mapeada durante a configuração do trabalho. Esse alerta não aparecerá ao usar o API ou`CLI`.
2. Se o nível de imposição for definido como`REQUIRED`, a criação do trabalho será rejeitada, a menos que a chave de partição seja mapeada para uma coluna de dados de origem.
3. Se o nível de imposição for alterado para `REQUIRED` após a criação da tarefa, a tarefa continuará sendo executada, mas qualquer registro que não tenha o mapeamento adequado para a chave de partição será rejeitado com um erro 4xx.

Interação com consulta agendada

Ao configurar um trabalho de consulta agendado para calcular e armazenar agregados, pacotes cumulativos e outras formas de dados pré-processados em uma tabela com uma chave de partição definida pelo cliente, há alguns cenários que podem afetar o processo:

1. Se o nível de imposição estiver definido como`OPTIONAL`, um alerta será exibido se a chave de partição não for mapeada durante a configuração do trabalho. Esse alerta não aparecerá ao usar o API ouCLI.
2. Se o nível de imposição for definido como`REQUIRED`, a criação do trabalho será rejeitada, a menos que a chave de partição seja mapeada para uma coluna de dados de origem.
3. Se o nível de imposição for alterado para `REQUIRED` após a criação do trabalho e os resultados da consulta agendada não contiverem a dimensão da chave de partição, todas as próximas iterações do trabalho falharão.

Adicionar tags e rótulos a recursos

Você pode rotular o Amazon Timestream LiveAnalytics para recursos usando tags. As tags permitem categorizar recursos de diferentes maneiras, por exemplo, por finalidade, proprietário, ambiente ou outros critérios. As tags podem ajudar a fazer o seguinte:

- Identificar rapidamente um recurso com base nas tags que você atribuiu a ele.
- Veja AWS as contas divididas por etiquetas.

A marcação é suportada por AWS serviços como Amazon Elastic Compute Cloud (AmazonEC2), Amazon Simple Storage Service (Amazon S3), Timestream for e muito mais. LiveAnalytics Uma marcação eficiente é capaz de fornecer insights de custos, permitindo criar relatórios entre serviços que possuem uma tag específica.

Para começar a usar tags, faça o seguinte:

1. Entenda as restrições de marcação.
2. Crie tags usando operações de marcação.

Por fim, é recomendável seguir estratégias de marcação ideais. Para obter informações, consulte Estratégias de marcação da AWS.

Restrições de marcação

Cada tag consiste em uma chave e um valor, ambos definidos por você. As seguintes restrições são aplicáveis:

- Cada Timestream da LiveAnalytics tabela só pode ter uma tag com a mesma chave. Se você tentar adicionar uma tag existente, o valor da tag existente será atualizado para o novo valor.
- Um valor atua como um descritor dentro de uma categoria de tag. No Timestream, LiveAnalytics o valor não pode ser vazio ou nulo.
- As chaves e valores das tags diferenciam maiúsculas de minúsculas.
- O comprimento máximo da chave é 128 caracteres Unicode.
- O número máximo de tags que você pode atribuir a um recurso é 50.
- Os caracteres permitidos são letras, espaço em branco e números, além dos seguintes caracteres especiais: `+ - = . _ : /`
- O número máximo de tags por recurso é 50.
- AWS- os nomes e valores das tags atribuídos recebem automaticamente o `aws:` prefixo, que você não pode atribuir. AWS-os nomes de tag atribuídos não contam para o limite de 50. Nomes de tags atribuídos pelo usuário têm o prefixo `user:` no relatório de alocação de custos.
- Não é possível colocar uma data retroativa na aplicação de uma tag.

Operações de marcação

Você pode adicionar, listar, editar ou excluir tags para bancos de dados e tabelas usando o Amazon Timestream LiveAnalytics para console, linguagem de consulta ou AWS Command Line Interface o ().AWS CLI

Tópicos

- Adicionar tags a bancos de dados e tabelas novos ou existentes usando o console

Adicionar tags a bancos de dados e tabelas novos ou existentes usando o console

Você pode usar o Timestream for LiveAnalytics console para adicionar tags a novos bancos de dados, tabelas e consultas agendadas ao criá-los. Você também pode adicionar, editar ou excluir tags para tabelas existentes.

Para marcar bancos de dados ao criá-los (console)

1. Abra o console Timestream em https://console.aws.amazon.com /timestream.
2. No painel de navegação, selecione Bancos de dados e, em seguida, selecione Criar banco de dados.
3. Na página Criar banco de dados, forneça um nome para o banco de dados. Insira uma chave e um valor para a tag e selecione Adicionar nova tag.
4. Selecione Criar banco de dados.

Para marcar tabelas ao criá-las (console)

1. Abra o console Timestream em https://console.aws.amazon.com /timestream.
2. No painel de navegação, selecione Tables (Tabelas) e Create table (Criar tabela).
3. Na página Criar fluxo de tempo para LiveAnalytics tabela, forneça um nome para a tabela. Insira uma chave e um valor para a tag e escolha Adicionar nova tag.
4. Escolha Create table.

Para marcar consultas agendadas ao criá-las (console)

1. Abra o console Timestream em https://console.aws.amazon.com /timestream.
2. No painel de navegação, escolha Consultas agendadas e, em seguida, escolha Criar consulta agendada.
3. Na Etapa 3. Defina a página de configurações de consulta, escolha Adicionar nova tag. Insira uma chave e um valor para a tag. Selecione Adicionar nova tag para adicionar mais tags.
4. Escolha Próximo.

Para marcar recursos existentes (console)

1. Abra o console Timestream em https://console.aws.amazon.com /timestream.
2. No painel de navegação, escolha Bancos de dados, tabelas ou consultas agendadas.
3. Escolha um banco de dados ou tabela na lista. Em seguida, selecione Gerenciar tags para adicionar, editar ou excluir tags.

Para obter informações sobre a estrutura da tag, consulte Restrições de marcação.

Segurança no Timestream para LiveAnalytics

A segurança na nuvem AWS é a maior prioridade. Como AWS cliente, você se beneficia de uma arquitetura de data center e rede criada para atender aos requisitos das organizações mais sensíveis à segurança.

A segurança é uma responsabilidade compartilhada entre você AWS e você. O modelo de responsabilidade compartilhada descreve isto como segurança da nuvem e segurança na nuvem:

- Segurança da nuvem — AWS é responsável por proteger a infraestrutura que executa AWS os serviços na AWS nuvem. AWS também fornece serviços que você pode usar com segurança. A eficácia da nossa segurança é regularmente testada e verificada por auditores de terceiros como parte dos Programas de conformidade da AWS. Para saber mais sobre os programas de conformidade que se aplicam ao Timestream for LiveAnalytics, consulte AWS Serviços no escopo por programa de conformidade.
- Segurança na nuvem — Sua responsabilidade é determinada pelo AWS serviço que você usa. Você também é responsável por outros fatores, inclusive a confidencialidade dos dados, os requisitos da organização, as leis e as regulamentações vigentes.

Esta documentação ajudará você a entender como aplicar o modelo de responsabilidade compartilhada ao usar o Timestream para. LiveAnalytics Os tópicos a seguir mostram como configurar o Timestream para atender LiveAnalytics aos seus objetivos de segurança e conformidade. Você também aprenderá a usar outros AWS serviços que podem ajudá-lo a monitorar e proteger seu Timestream para obter LiveAnalytics recursos.

Tópicos

- Proteção de dados no Timestream para LiveAnalytics
- Gerenciamento de identidade e acesso para o Amazon Timestream para LiveAnalytics
- Registro e monitoramento no Timestream para LiveAnalytics
- Resiliência no Amazon Timestream Live Analytics
- Segurança da infraestrutura no Amazon Timestream Live Analytics
- Análise de configuração e vulnerabilidade no Timestream
- Resposta a incidentes no Timestream para LiveAnalytics
- VPCpontos finais ()AWS PrivateLink
- Melhores práticas de segurança para o Amazon Timestream for LiveAnalytics

Proteção de dados no Timestream para LiveAnalytics

O modelo de responsabilidade AWS compartilhada se aplica à proteção de dados no Amazon Timestream Live Analytics. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa todos os Nuvem AWS. Você é responsável por manter o controle sobre seu conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para obter mais informações sobre privacidade de dados, consulte Privacidade de dados FAQ. Para obter informações sobre proteção de dados na Europa, consulte o Modelo de Responsabilidade AWS Compartilhada e GDPR a postagem no blog AWS de segurança.

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure usuários individuais com AWS IAM Identity Center ou AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use a autenticação multifator (MFA) com cada conta.
- UseSSL/TLSpara se comunicar com AWS os recursos. Exigimos TLS 1,2 e recomendamos TLS 1,3.
- Configure API e registre as atividades do usuário com AWS CloudTrail. Para obter informações sobre o uso de CloudTrail trilhas para capturar AWS atividades, consulte Como trabalhar com CloudTrail trilhas no Guia AWS CloudTrail do usuário.
- Use soluções de AWS criptografia, juntamente com todos os controles de segurança padrão Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sigilosos armazenados no Amazon S3.
- Se você precisar de FIPS 140-3 módulos criptográficos validados ao acessar AWS por meio de uma interface de linha de comando ou umaAPI, use um endpoint. FIPS Para obter mais informações sobre os FIPS endpoints disponíveis, consulte Federal Information Processing Standard (FIPS) 140-3.

É altamente recomendável que nunca sejam colocadas informações de identificação confidenciais, como endereços de e-mail dos seus clientes, em marcações ou campos de formato livre, como um campo Nome. Isso inclui quando você trabalha com o Timestream Live Analytics ou outro Serviços da AWS usando o console,, API AWS CLI, ou. AWS SDKs Quaisquer dados inseridos em tags ou

campos de texto de formato livre usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, é altamente recomendável que você não inclua informações de credenciais no URL para validar sua solicitação para esse servidor.

Para obter informações mais detalhadas sobre o Timestream para tópicos de proteção de LiveAnalytics dados, como criptografia em repouso e gerenciamento de chaves, selecione qualquer um dos tópicos disponíveis abaixo.

Tópicos

- Criptografia inativa
- Criptografia em trânsito
- Gerenciamento de chaves

Criptografia inativa

O Timestream para LiveAnalytics criptografia em repouso fornece segurança aprimorada ao criptografar todos os seus dados em repouso usando chaves de criptografia armazenadas em AWS Key Management Service ().AWS KMS Essa funcionalidade ajuda a reduzir a carga e complexidade operacionais necessárias para proteger dados confidenciais. Com a criptografia de dados em repouso, você pode criar aplicativos confidenciais que atendem a requisitos rigorosos de conformidade e regulamentação de criptografia.

- A criptografia está ativada por padrão no Timestream para LiveAnalytics banco de dados e não pode ser desativada. O algoritmo de criptografia AES -256 padrão do setor é o algoritmo de criptografia padrão usado.
- AWS KMS é necessário para criptografia em repouso no Timestream for. LiveAnalytics
- Não é possível criptografar apenas um subconjunto de itens em uma tabela.
- Você não precisa modificar suas aplicações cliente de banco de dados para usar a criptografia.

Se você não fornecer uma chave, o Timestream for LiveAnalytics cria e usa uma AWS KMS chave nomeada `alias/aws/timestream` na sua conta.

Você pode usar sua própria chave gerenciada pelo cliente KMS para criptografar seu Timestream para dados. LiveAnalytics Para obter mais informações sobre chaves no Timestream for LiveAnalytics, consulte. Gerenciamento de chaves

Timestream para LiveAnalytics armazenar seus dados em dois níveis de armazenamento, armazenamento de memória e armazenamento magnético. Os dados do armazenamento de memória são criptografados usando um Timestream para chave LiveAnalytics de serviço. Os dados do armazenamento magnético são criptografados usando sua AWS KMS chave.

O serviço Timestream Query requer credenciais para acessar seus dados. Essas credenciais são criptografadas usando sua KMS chave.

> **Note**
>
> O Timestream for LiveAnalytics não exige todas as operações AWS KMS de descriptografia. Em vez disso, ele mantém um cache local de chaves por 5 minutos com tráfego ativo. Quaisquer alterações de permissão são propagadas pelo Timestream para o LiveAnalytics sistema com consistência eventual em no máximo 5 minutos.

Criptografia em trânsito

Todos os seus dados do Timestream Live Analytics são criptografados em trânsito. Por padrão, todas as comunicações de e para o Timestream para LiveAnalytics são protegidas usando a criptografia Transport Layer Security (TLS).

Gerenciamento de chaves

Você pode gerenciar chaves para o Amazon Timestream Live Analytics usando AWS o Key Management Service ().AWS KMS O Timestream Live Analytics requer o uso de KMS para criptografar seus dados. Você tem as seguintes opções para gerenciamento de chaves, dependendo de quanto controle você precisa sobre suas chaves:

Recursos de banco de dados e tabelas

- Chave gerenciada pelo Timestream Live Analytics: se você não fornecer uma chave, o Timestream Live Analytics criará uma chave usando. `alias/aws/timestream` KMS
- Chave gerenciada pelo KMS cliente: as chaves gerenciadas pelo cliente são suportadas. Escolha essa opção se precisar de mais controle sobre as permissões e o ciclo de vida de suas chaves, incluindo a capacidade de alterná-las automaticamente anualmente.

Recurso de consulta agendada

- Chave de propriedade do Timestream Live Analytics: se você não fornecer uma chave, o Timestream Live Analytics usará sua própria chave para criptografar o recurso Query. KMS Essa chave estará presente na conta do Timestream. Consulte as chaves AWS próprias no guia do KMS desenvolvedor para obter mais detalhes.
- Chave gerenciada pelo KMS cliente: as chaves gerenciadas pelo cliente são suportadas. Escolha essa opção se precisar de mais controle sobre as permissões e o ciclo de vida de suas chaves, incluindo a capacidade de alterná-las automaticamente anualmente.

KMSchaves em um armazenamento de chaves externo (XKS) não são suportadas.

Gerenciamento de identidade e acesso para o Amazon Timestream para LiveAnalytics

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. IAMos administradores controlam quem pode ser autenticado (conectado) e autorizado (tem permissões) a usar o Timestream para obter recursos. LiveAnalytics IAMé um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Tópicos

- Público
- Autenticando com identidades
- Gerenciando acesso usando políticas
- Como o Amazon LiveAnalytics Timestream for funciona com IAM
- AWS políticas gerenciadas para o Amazon Timestream Live Analytics
- Amazon Timestream LiveAnalytics para exemplos de políticas baseadas em identidade
- Solução de problemas do Amazon Timestream LiveAnalytics para identidade e acesso

Público

A forma como você usa AWS Identity and Access Management (IAM) difere, dependendo do trabalho que você faz no Timestream. LiveAnalytics

Usuário do serviço — Se você usar o Timestream for LiveAnalytics Service para realizar seu trabalho, seu administrador fornecerá as credenciais e as permissões de que você precisa. À medida que você usa mais Timestream para que os LiveAnalytics recursos façam seu trabalho, talvez você precise de permissões adicionais. Entender como o acesso é gerenciado pode ajudar você a solicitar as permissões corretas ao seu administrador. Se você não conseguir acessar um recurso no Timestream para LiveAnalytics, consulte. Solução de problemas do Amazon Timestream LiveAnalytics para identidade e acesso

Administrador de serviços — Se você é responsável pelo Timestream pelos LiveAnalytics recursos da sua empresa, provavelmente tem acesso total ao Timestream for. LiveAnalytics É seu trabalho determinar qual Timestream para LiveAnalytics recursos e recursos seus usuários do serviço devem acessar. Em seguida, você deve enviar solicitações ao IAM administrador para alterar as permissões dos usuários do serviço. Revise as informações nesta página para entender os conceitos básicos doIAM. Para saber mais sobre como sua empresa pode usar o IAM Timestream for LiveAnalytics, consulte. Como o Amazon LiveAnalytics Timestream for funciona com IAM

IAMadministrador — Se você for IAM administrador, talvez queira saber detalhes sobre como criar políticas para gerenciar o acesso ao Timestream para. LiveAnalytics Para ver um exemplo de Timestream para políticas LiveAnalytics baseadas em identidade que você pode usar em, consulte. IAM Amazon Timestream LiveAnalytics para exemplos de políticas baseadas em identidade

Autenticando com identidades

A autenticação é a forma como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado (conectado AWS) como IAM usuário ou assumindo uma IAM função. Usuário raiz da conta da AWS

Você pode entrar AWS como uma identidade federada usando credenciais fornecidas por meio de uma fonte de identidade. AWS IAM Identity Center Os usuários (do IAM Identity Center), a autenticação de login único da sua empresa e suas credenciais do Google ou do Facebook são exemplos de identidades federadas. Quando você entra como uma identidade federada, seu administrador configurou previamente a federação de identidades usando IAM funções. Ao acessar AWS usando a federação, você está assumindo indiretamente uma função.

Dependendo do tipo de usuário que você é, você pode entrar no AWS Management Console ou no portal de AWS acesso. Para obter mais informações sobre como fazer login em AWS, consulte Como fazer login Conta da AWS no Guia do Início de Sessão da AWS usuário.

Se você acessar AWS programaticamente, AWS fornece um kit de desenvolvimento de software (SDK) e uma interface de linha de comando (CLI) para assinar criptograficamente suas solicitações

usando suas credenciais. Se você não usa AWS ferramentas, você mesmo deve assinar as solicitações. Para obter mais informações sobre como usar o método recomendado para você mesmo assinar solicitações, consulte AWS Signature versão 4 para API solicitações no Guia IAM do usuário.

Independente do método de autenticação usado, também pode ser exigido que você forneça informações adicionais de segurança. Por exemplo, AWS recomenda que você use a autenticação multifator (MFA) para aumentar a segurança da sua conta. Para saber mais, consulte Autenticação multifator no Guia do AWS IAM Identity Center usuário e Autenticação AWS multifator IAM no Guia do IAMusuário.

Grupos e usuários do IAM

Um IAMusuário é uma identidade dentro da sua Conta da AWS que tem permissões específicas para uma única pessoa ou aplicativo. Sempre que possível, recomendamos confiar em credenciais temporárias em vez de criar IAM usuários que tenham credenciais de longo prazo, como senhas e chaves de acesso. No entanto, se você tiver casos de uso específicos que exijam credenciais de longo prazo com IAM os usuários, recomendamos que você alterne as chaves de acesso. Para obter mais informações, consulte Alterne as chaves de acesso regularmente para casos de uso que exigem credenciais de longo prazo no Guia do IAMusuário.

Um IAMgrupo é uma identidade que especifica uma coleção de IAM usuários. Não é possível fazer login como um grupo. É possível usar grupos para especificar permissões para vários usuários de uma vez. Os grupos facilitam o gerenciamento de permissões para grandes conjuntos de usuários. Por exemplo, você pode ter um grupo chamado IAMAdminse conceder a esse grupo permissões para administrar IAM recursos.

Usuários são diferentes de perfis. Um usuário é exclusivamente associado a uma pessoa ou a uma aplicação, mas um perfil pode ser assumido por qualquer pessoa que precisar dele. Os usuários têm credenciais permanentes de longo prazo, mas os perfis fornecem credenciais temporárias. Para saber mais, consulte Casos de uso para IAM usuários no Guia IAM do usuário.

IAMfunções

Uma IAMfunção é uma identidade dentro da sua Conta da AWS que tem permissões específicas. É semelhante a um IAM usuário, mas não está associado a uma pessoa específica. Para assumir temporariamente uma IAM função no AWS Management Console, você pode alternar de usuário para IAM função (console). Você pode assumir uma função chamando uma AWS API operação AWS CLI or ou usando uma personalizadaURL. Para obter mais informações sobre métodos de uso de funções, consulte Métodos para assumir uma função no Guia IAM do usuário.

IAMfunções com credenciais temporárias são úteis nas seguintes situações:

- Acesso de usuário federado: para atribuir permissões a identidades federadas, você pode criar um perfil e definir permissões para ele. Quando uma identidade federada é autenticada, essa identidade é associada ao perfil e recebe as permissões definidas pelo mesmo. Para obter informações sobre funções para federação, consulte Criação de uma função para um provedor de identidade terceirizado no Guia IAM do usuário. Se você usa o IAM Identity Center, configura um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o IAM Identity Center correlaciona o conjunto de permissões a uma função em. IAM Para obter informações sobre conjuntos de permissões, consulte Conjuntos de Permissões no Manual do Usuário do AWS IAM Identity Center .
- Permissões temporárias IAM de IAM usuário — Um usuário ou função pode assumir uma IAM função para assumir temporariamente permissões diferentes para uma tarefa específica.
- Acesso entre contas — Você pode usar uma IAM função para permitir que alguém (um diretor confiável) em uma conta diferente acesse recursos em sua conta. Os perfis são a principal forma de conceder acesso entre contas. No entanto, com alguns Serviços da AWS, você pode anexar uma política diretamente a um recurso (em vez de usar uma função como proxy). Para saber a diferença entre funções e políticas baseadas em recursos para acesso entre contas, consulte Acesso a recursos entre contas IAM no Guia do IAM usuário.
- Acesso entre serviços — Alguns Serviços da AWS usam recursos em outros Serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicativos na Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso usando as permissões do principal de chamada, usando um perfil de serviço ou um perfil vinculado a serviço.
  - Sessões de acesso direto (FAS) — Quando você usa um IAM usuário ou função para realizar ações em AWS, você é considerado principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. FASusa as permissões do diretor chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) para fazer solicitações aos serviços posteriores. FASas solicitações são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou com recursos para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer FAS solicitações, consulte Encaminhar sessões de acesso.
  - Função de serviço — Uma função de serviço é uma IAMfunção que um serviço assume para realizar ações em seu nome. Um IAM administrador pode criar, modificar e excluir uma função

de serviço internamenteIAM. Para obter mais informações, consulte Criação de uma função para delegar permissões a uma AWS service (Serviço da AWS) no Guia do IAM usuário.

- Função vinculada ao serviço — Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode presumir a função de executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um IAM administrador pode visualizar, mas não editar, as permissões das funções vinculadas ao serviço.

- Aplicativos em execução na Amazon EC2 — Você pode usar uma IAM função para gerenciar credenciais temporárias para aplicativos que estão sendo executados em uma EC2 instância e fazendo AWS CLI AWS API solicitações. Isso é preferível a armazenar chaves de acesso na EC2 instância. Para atribuir uma AWS função a uma EC2 instância e disponibilizá-la para todos os aplicativos, você cria um perfil de instância anexado à instância. Um perfil de instância contém a função e permite que programas em execução na EC2 instância recebam credenciais temporárias. Para obter mais informações, consulte Como usar uma IAM função para conceder permissões a aplicativos executados em EC2 instâncias da Amazon no Guia IAM do usuário.

Gerenciando acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política é um objeto AWS que, quando associada a uma identidade ou recurso, define suas permissões. AWS avalia essas políticas quando um principal (usuário, usuário raiz ou sessão de função) faz uma solicitação. As permissões nas políticas determinam se a solicitação será permitida ou negada. A maioria das políticas é armazenada AWS como JSON documentos. Para obter mais informações sobre a estrutura e o conteúdo dos documentos de JSON política, consulte Visão geral das JSON políticas no Guia IAM do usuário.

Os administradores podem usar AWS JSON políticas para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

Por padrão, usuários e funções não têm permissões. Para conceder permissão aos usuários para realizar ações nos recursos de que precisam, um IAM administrador pode criar IAM políticas. O administrador pode então adicionar as IAM políticas às funções e os usuários podem assumir as funções.

IAMas políticas definem permissões para uma ação, independentemente do método usado para realizar a operação. Por exemplo, suponha que você tenha uma política que permite a

ação `iam:GetRole`. Um usuário com essa política pode obter informações de função do AWS Management Console AWS CLI, do ou do AWS API.

Políticas baseadas em identidade

Políticas baseadas em identidade são documentos de políticas de JSON permissões que você pode anexar a uma identidade, como um IAM usuário, grupo de usuários ou função. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Definir IAM permissões personalizadas com políticas gerenciadas pelo cliente no Guia do IAMusuário.

As políticas baseadas em identidade podem ser categorizadas ainda adicionalmente como políticas em linha ou políticas gerenciadas. As políticas em linha são anexadas diretamente a um único usuário, grupo ou perfil. As políticas gerenciadas são políticas autônomas que você pode associar a vários usuários, grupos e funções em seu Conta da AWS. As políticas AWS gerenciadas incluem políticas gerenciadas e políticas gerenciadas pelo cliente. Para saber como escolher entre uma política gerenciada ou uma política em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do IAMusuário.

Políticas baseadas no recurso

Políticas baseadas em recursos são documentos JSON de política que você anexa a um recurso. Exemplos de políticas baseadas em recursos são políticas de confiança de IAM funções e políticas de bucket do Amazon S3. Em serviços que suportem políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações um principal especificado pode executar nesse recurso e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas de uma política baseada IAM em recursos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLssão semelhantes às políticas baseadas em recursos, embora não usem o formato de documento JSON de política.

Amazon S3, AWS WAF, e Amazon VPC são exemplos de serviços que oferecem suporte. ACLs Para saber maisACLs, consulte a visão geral da lista de controle de acesso (ACL) no Guia do desenvolvedor do Amazon Simple Storage Service.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais menos comuns. Esses tipos de política podem definir o máximo de permissões concedidas a você pelos tipos de política mais comuns.

- Limites de permissões — Um limite de permissões é um recurso avançado no qual você define as permissões máximas que uma política baseada em identidade pode conceder a uma IAM entidade (IAMusuário ou função). É possível definir um limite de permissões para uma entidade. As permissões resultantes são a interseção das políticas baseadas em identidade de uma entidade com seus limites de permissões. As políticas baseadas em recurso que especificam o usuário ou o perfil no campo `Principal` não são limitadas pelo limite de permissões. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações sobre limites de permissões, consulte Limites de permissões para IAM entidades no Guia IAM do usuário.
- Políticas de controle de serviço (SCPs) — SCPs são JSON políticas que especificam as permissões máximas para uma organização ou unidade organizacional (OU) em AWS Organizations. AWS Organizations é um serviço para agrupar e gerenciar centralmente vários Contas da AWS que sua empresa possui. Se você habilitar todos os recursos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as suas contas. Os SCP limites de permissões para entidades nas contas dos membros, incluindo cada uma Usuário raiz da conta da AWS. Para obter mais informações sobre Organizations eSCPs, consulte Políticas de controle de serviços no Guia AWS Organizations do Usuário.
- Políticas de sessão: são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para um perfil ou um usuário federado. As permissões da sessão resultante são a interseção das políticas baseadas em identidade do usuário ou do perfil e das políticas de sessão. As permissões também podem ser provenientes de uma política baseada em atributo. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações, consulte Políticas de sessão no Guia IAM do usuário.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de política estão envolvidos, consulte Lógica de avaliação de políticas no Guia IAM do usuário.

Como o Amazon LiveAnalytics Timestream for funciona com IAM

Antes de gerenciar IAM o acesso ao Timestream for LiveAnalytics, você deve entender quais IAM recursos estão disponíveis para uso com o Timestream for. LiveAnalytics Para obter uma visão geral de como o Timestream for LiveAnalytics e outros AWS serviços funcionamIAM, consulte AWS Serviços que funcionam com IAM no Guia do IAM usuário.

Tópicos

- Cronograma para LiveAnalytics políticas baseadas em identidade
- Cronograma para LiveAnalytics políticas baseadas em recursos
- Autorização baseada no Timestream para tags LiveAnalytics
- Cronograma para funções LiveAnalytics IAM

Cronograma para LiveAnalytics políticas baseadas em identidade

Com políticas IAM baseadas em identidade, você pode especificar ações e recursos permitidos ou negados, bem como as condições sob as quais as ações são permitidas ou negadas. Timestream for LiveAnalytics suporta ações e recursos específicos e chaves de condição. Para saber mais sobre todos os elementos que você usa em uma JSON política, consulte Referência IAM JSON de elementos de política no Guia do IAM usuário.

Ações

Os administradores podem usar AWS JSON políticas para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O `Action` elemento de uma JSON política descreve as ações que você pode usar para permitir ou negar acesso em uma política. As ações de política geralmente têm o mesmo nome da AWS API operação associada. Há algumas exceções, como ações somente com permissão que não têm uma operação correspondente. API Algumas operações também exigem várias ações em uma política. Essas ações adicionais são chamadas de ações dependentes.

Incluem ações em uma política para conceder permissões para executar a operação associada.

Você pode especificar as ações a seguir no elemento Ação de uma declaração de IAM política. Use políticas para conceder permissões para realizar uma operação emAWS. Quando você usa uma ação em uma política, geralmente permite ou nega acesso à API operação, CLI comando ou SQL comando com o mesmo nome.

Em alguns casos, uma única ação controla o acesso a uma API operação e a um SQL comando. Como alternativa, algumas operações exigem várias ações diferentes.

Para obter uma lista de Timestream compatíveis com s LiveAnalytics `Action`, consulte a tabela abaixo:

> **Note**
>
> Para todos os bancos de dados específicos`Actions`, você pode especificar um banco de dados ARN para limitar a ação a um banco de dados específico.

Ações	Descrição	Nível de acesso	Tipos de recursos (*necessários)
DescribeEndpoints	Retorna o endpoint do Timestream para o qual as solicitações subsequentes devem ser feitas.	Todos	*
Selecionar	Execute consultas no Timestream que selecionam dados de uma ou mais tabelas. Consulte esta nota para obter uma explicação detalhada	Leitura	mesa*
CancelQuery	Cancele uma consulta.	Leitura	*

Ações	Descrição	Nível de acesso	Tipos de recursos (*necessários)
ListTables	Obtenha a lista de tabelas.	Listar	banco de dados*
ListDatabases	Obtenha a lista de bancos de dados.	Listar	*
ListMeasures	Veja a lista de medidas.	Leitura	mesa*
DescribeTable	Obtenha a descrição da tabela.	Leitura	mesa*
DescribeDatabase	Obtenha a descrição do banco de dados.	Leitura	banco de dados*
SelectValues	Execute consultas que não exijam a especificação de um recurso específic o. Consulte esta nota para obter uma explicação detalhada.	Leitura	*
WriteRecords	Insira dados no Timestream.	Escrever	mesa*
CreateTable	Crie uma tabela.	Escrever	banco de dados*
CreateDatabase	Crie um banco de dados.	Escrever	*
DeleteDatabase	Exclua um banco de dados.	Escrever	*
UpdateDatabase	Atualize um banco de dados.	Escrever	*

Ações	Descrição	Nível de acesso	Tipos de recursos (*necessários)
DeleteTable	Exclua uma tabela.	Escrever	banco de dados*
UpdateTable	Atualize uma tabela.	Escrever	banco de dados*

SelectValues versus selecionar:

`SelectValues`é um `Action` que é usado para consultas que não exigem um recurso. Um exemplo de consulta que não exige um recurso é o seguinte:

```
SELECT 1
```

Observe que essa consulta não se refere a um determinado Timestream para LiveAnalytics o recurso. Considere outro exemplo:

```
SELECT now()
```

Essa consulta retorna o timestamp atual usando a `now()` função, mas não exige que um recurso seja especificado. `SelectValues`é frequentemente usado para testes, para que o Timestream for LiveAnalytics possa executar consultas sem recursos. Agora, considere uma `Select` consulta:

```
SELECT * FROM database.table
```

Esse tipo de consulta requer um recurso, especificamente um formulário Timestream LiveAnalytics `table`, para que os dados especificados possam ser buscados na tabela.

Recursos

Os administradores podem usar AWS JSON políticas para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento `Resource` JSON de política especifica o objeto ou objetos aos quais a ação se aplica. As instruções devem incluir um elemento `Resource` ou `NotResource`. Como prática recomendada, especifique um recurso usando seu Amazon Resource Name (ARN). Isso pode ser feito para ações que oferecem compatibilidade com um tipo de recurso específico, conhecido como permissões em nível de recurso.

Para ações que não oferecem compatibilidade com permissões em nível de recurso, como operações de listagem, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

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

No Timestream, LiveAnalytics bancos de dados e tabelas podem ser usados no `Resource` elemento de IAM permissões.

O recurso Timestream para LiveAnalytics banco de dados tem o seguinte: ARN

```
arn:${Partition}:timestream:${Region}:${Account}:database/${DatabaseName}
```

O recurso Timestream for LiveAnalytics table tem o seguinte: ARN

```
arn:${Partition}:timestream:${Region}:${Account}:database/${DatabaseName}/table/
${TableName}
```

Para obter mais informações sobre o formato deARNs, consulte Amazon Resource Names (ARNs) e AWS Service Namespaces.

Por exemplo, para especificar o `database` espaço de teclas em sua declaração, use o seguinte: ARN

```
"Resource": "arn:aws:timestream:us-east-1:123456789012:database/mydatabase"
```

Para especificar todos os bancos de dados que pertencem a uma conta específica, use o caractere curinga (*):

```
"Resource": "arn:aws:timestream:us-east-1:123456789012:database/*"
```

Alguns fluxos de tempo para LiveAnalytics ações, como aqueles para criar recursos, não podem ser executados em um recurso específico. Nesses casos, você deve utilizar o caractere curinga (*).

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

Chaves de condição

O Timestream for LiveAnalytics não fornece nenhuma chave de condição específica do serviço, mas oferece suporte ao uso de algumas chaves de condição globais. Para ver todas as chaves

de condição AWS globais, consulte Chaves de contexto de condição AWS global no Guia IAM do usuário.

Exemplos

Para ver exemplos de Timestream para políticas LiveAnalytics baseadas em identidade, consulte. Amazon Timestream LiveAnalytics para exemplos de políticas baseadas em identidade

Cronograma para LiveAnalytics políticas baseadas em recursos

O Timestream for LiveAnalytics não oferece suporte a políticas baseadas em recursos. Para visualizar um exemplo de uma política baseada em recurso detalhada, consulte https://docs.aws.amazon.com/lambda/latest/dg/access-control-resource-based.html.

Autorização baseada no Timestream para tags LiveAnalytics

Você pode gerenciar o acesso ao seu Timestream para LiveAnalytics recursos usando tags. Para gerenciar o acesso a recursos baseado em tags, forneça informações sobre as tags no elemento de condição de uma política usando as chaves de condição `timestream:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou `aws:TagKeys`. Para obter mais informações sobre como marcar Timestream para LiveAnalytics recursos, consulte. the section called “Marcar recursos”

Para visualizar exemplos de políticas baseadas em identidade para limitar o acesso a um recurso baseado em tags desse recurso, consulte Cronograma para acesso a LiveAnalytics recursos com base em tags.

Cronograma para funções LiveAnalytics IAM

Uma IAMfunção é uma entidade dentro da sua AWS conta que tem permissões específicas.

Usando credenciais temporárias com o Timestream para LiveAnalytics

Você pode usar credenciais temporárias para entrar com a federação, assumir uma IAM função ou assumir uma função entre contas. Você obtém credenciais de segurança temporárias ligando para AWS STS API operações como AssumeRoleou GetFederationToken.

Funções vinculadas a serviço

O Timestream for LiveAnalytics não oferece suporte a funções vinculadas a serviços.

Perfis de serviço

O Timestream for LiveAnalytics não oferece suporte a funções de serviço.

AWS políticas gerenciadas para o Amazon Timestream Live Analytics

Uma política AWS gerenciada é uma política autônoma criada e administrada por AWS. AWS as políticas gerenciadas são projetadas para fornecer permissões para muitos casos de uso comuns, para que você possa começar a atribuir permissões a usuários, grupos e funções.

Lembre-se de que as políticas AWS gerenciadas podem não conceder permissões de privilégio mínimo para seus casos de uso específicos porque elas estão disponíveis para uso de todos os AWS clientes. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo cliente da específicas para seus casos de uso.

Você não pode alterar as permissões definidas nas políticas AWS gerenciadas. Se AWS atualizar as permissões definidas em uma política AWS gerenciada, a atualização afetará todas as identidades principais (usuários, grupos e funções) às quais a política está anexada. AWS é mais provável que atualize uma política AWS gerenciada quando uma nova AWS service (Serviço da AWS) é lançada ou novas API operações são disponibilizadas para os serviços existentes.

Para obter mais informações, consulte as políticas AWS gerenciadas no Guia IAM do usuário.

Tópicos

- AWS política gerenciada: AmazonTimestreamReadOnlyAccess
- AWS política gerenciada: AmazonTimestreamConsoleFullAccess
- AWS política gerenciada: AmazonTimestreamFullAccess
- Atualizações do Timestream Live Analytics para AWS políticas gerenciadas

AWS política gerenciada: AmazonTimestreamReadOnlyAccess

Você pode vincular a `AmazonTimestreamReadOnlyAccess` aos seus usuários, grupos e perfis. A política fornece acesso somente para leitura ao Amazon Timestream.

Detalhes de permissões

Esta política inclui a seguinte permissão:

- `Amazon Timestream`— Fornece acesso somente para leitura ao Amazon Timestream. Essa política também concede permissão para cancelar qualquer consulta em execução.

Para revisar esta política em JSON formato, consulte AmazonTimestreamReadOnlyAccess.

AWS política gerenciada: AmazonTimestreamConsoleFullAccess

Você pode vincular a `AmazonTimestreamConsoleFullAccess` aos seus usuários, grupos e perfis.

A política fornece acesso total para gerenciar o Amazon Timestream usando o. AWS Management Console Essa política também concede permissões para determinadas AWS KMS operações e operações para gerenciar suas consultas salvas.

Detalhes de permissões

Esta política inclui as seguintes permissões:

- `Amazon Timestream`— Concede aos diretores acesso total ao Amazon Timestream.
- `AWS KMS`— Permite que os diretores listem aliases e descrevam as chaves.
- `Amazon S3`— Permite que os diretores listem todos os buckets do Amazon S3.
- `Amazon SNS`— Permite que os diretores listem SNS tópicos da Amazon.
- `IAM`— Permite que os diretores listem IAM funções.
- `DBQMS`: permite que as entidades principais acessem, criem, excluam, descrevam e atualizem consultas. O Database Query Metadata Service (dbqms) é um serviço somente interno. Ele fornece suas consultas recentes e salvas para o editor de consultas em várias Serviços da AWS, incluindo o AWS Management Console Amazon Timestream.

Para revisar esta política em JSON formato, consulte AmazonTimestreamConsoleFullAccess.

AWS política gerenciada: AmazonTimestreamFullAccess

Você pode vincular a `AmazonTimestreamFullAccess` aos seus usuários, grupos e perfis.

A política fornece acesso total ao Amazon Timestream. Essa política também concede permissões para determinadas AWS KMS operações.

Detalhes de permissões

Esta política inclui as seguintes permissões:

- `Amazon Timestream`— Concede aos diretores acesso total ao Amazon Timestream.
- `AWS KMS`— Permite que os diretores listem aliases e descrevam as chaves.
- `Amazon S3`— Permite que os diretores listem todos os buckets do Amazon S3.

Para revisar esta política em JSON formato, consulte AmazonTimestreamFullAccess.

Atualizações do Timestream Live Analytics para AWS políticas gerenciadas

Veja detalhes sobre as atualizações das políticas AWS gerenciadas do Timestream Live Analytics desde que esse serviço começou a rastrear essas alterações. Para receber alertas automáticos sobre alterações nessa página, assine o RSS feed na página de histórico de documentos do Timestream Live Analytics.

Alteração	Descrição	Data
AmazonTimestreamReadOnlyAccess: atualizar para uma política existente	A `timestream:DescribeAccountSettings` ação foi adicionada à política `AmazonTimestreamReadOnlyAccess` gerenciada existente. Essa ação é usada para descrever Conta da AWS as configurações. O Timestream Live Analytics também atualizou essa política gerenciada adicionando um `Sid` campo. A atualização da política não afeta o uso da política `AmazonTimestreamRe`	03 de junho de 2024

Alteração	Descrição	Data
	`adOnlyAccess` gerenciada.	
AmazonTimestreamReadOnlyAccess: atualizar para uma política existente	As `timestream:ListBatchLoadTasks` ações `timestream:DescribeBatchLoadTask` e foram adicionadas à política `AmazonTimestreamReadOnlyAccess` gerenciada existente. Essas ações são usadas ao listar e descrever tarefas de carregamento em lote. A atualização da política não afeta o uso da política `AmazonTimestreamReadOnlyAccess` gerenciada.	24 de fevereiro de 2023

Alteração	Descrição	Data
AmazonTimestreamReadOnlyAccess: atualizar para uma política existente	As `timestream:ListScheduledQueries` ações `timestream:DescribeScheduledQuery` e foram adicionadas à política `AmazonTimestreamReadOnlyAccess` gerenciada existente. Essas ações são usadas ao listar e descrever consultas agendadas existentes. A atualização da política não afeta o uso da política `AmazonTimestreamReadOnlyAccess` gerenciada.	29 de novembro de 2021
AmazonTimestreamConsoleFullAccess: atualizar para uma política existente	A `s3:ListAllMyBuckets` ação foi adicionada à política `AmazonTimestreamConsoleFullAccess` gerenciada existente. Essa ação é usada quando você especifica um bucket do Amazon S3 para o Timestream para registrar erros de gravação no armazenamento magnético. A atualização da política não afeta o uso da política `AmazonTimestreamConsoleFullAccess` gerenciada.	29 de novembro de 2021

Alteração	Descrição	Data
AmazonTimestreamFullAccess: atualizar para uma política existente	A `s3:ListAllMyBuckets` ação foi adicionada à política `AmazonTimestreamFullAccess` gerenciada existente. Essa ação é usada quando você especifica um bucket do Amazon S3 para o Timestream para registrar erros de gravação no armazenamento magnético. A atualização da política não afeta o uso da política `AmazonTimestreamFullAccess` gerenciada.	29 de novembro de 2021
AmazonTimestreamConsoleFullAccess: atualizar para uma política existente	Foram removidas ações redundantes da política `AmazonTimestreamConsoleFullAccess` gerenciada existente. Anteriormente, essa política incluía uma ação redundante. `dbqms:DescribeQueryHistory` A política atualizada remove a ação redundante. A atualização da política não afeta o uso da política `AmazonTimestreamConsoleFullAccess` gerenciada.	23 de abril de 2021

Alteração	Descrição	Data
O Timestream Live Analytics começou a monitorar as alterações	O Timestream Live Analytics começou a monitorar as alterações em suas políticas AWS gerenciadas.	21 de abril de 2021

Amazon Timestream LiveAnalytics para exemplos de políticas baseadas em identidade

Por padrão, IAM usuários e funções não têm permissão para criar ou modificar o Timestream para LiveAnalytics recursos. Eles também não podem realizar tarefas usando o AWS Management Console CQLSH, AWS CLI, ou AWS API. Um IAM administrador deve criar IAM políticas que concedam aos usuários e funções permissão para realizar API operações específicas nos recursos especificados de que precisam. O administrador deve então anexar essas políticas aos IAM usuários ou grupos que exigem essas permissões.

Para saber como criar uma política IAM baseada em identidade usando esses exemplos de documentos JSON de política, consulte Criação de políticas na JSON guia do IAMusuário.

Tópicos

- Melhores práticas de política
- Usando o Timestream para console LiveAnalytics
- Permitir que usuários visualizem suas próprias permissões
- Operações comuns no Timestream para LiveAnalytics
- Cronograma para acesso a LiveAnalytics recursos com base em tags
- Consultas programadas

Melhores práticas de política

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir o Timestream para LiveAnalytics recursos em sua conta. Essas ações podem incorrer em custos para seus Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com as políticas AWS gerenciadas e avance para as permissões de privilégios mínimos — Para começar a conceder permissões aos seus usuários e cargas de trabalho, use as políticas AWS gerenciadas que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis no seu Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo AWS cliente que sejam específicas para seus casos de uso. Para obter mais informações, consulte políticas AWS gerenciadas ou políticas AWS gerenciadas para funções de trabalho no Guia IAM do usuário.
- Aplique permissões com privilégios mínimos — Ao definir permissões com IAM políticas, conceda somente as permissões necessárias para realizar uma tarefa. Você faz isso definindo as ações que podem ser executadas em atributos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para obter mais informações sobre IAM como usar para aplicar permissões, consulte Políticas e permissões IAM no Guia IAM do usuário.
- Use condições nas IAM políticas para restringir ainda mais o acesso — Você pode adicionar uma condição às suas políticas para limitar o acesso a ações e recursos. Por exemplo, você pode escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usandoSSL. Você também pode usar condições para conceder acesso às ações de serviço se elas forem usadas por meio de uma ação específica AWS service (Serviço da AWS), como AWS CloudFormation. Para obter mais informações, consulte elementos IAM JSON da política: Condição no Guia IAM do usuário.
- Use o IAM Access Analyzer para validar suas IAM políticas e garantir permissões seguras e funcionais — o IAM Access Analyzer valida políticas novas e existentes para que as políticas sigam a linguagem da IAM política (JSON) e as melhores práticas. IAM IAMO Access Analyzer fornece mais de 100 verificações de políticas e recomendações práticas para ajudá-lo a criar políticas seguras e funcionais. Para obter mais informações, consulte Validar políticas com o IAM Access Analyzer no Guia do IAMUsuário.
- Exigir autenticação multifator (MFA) — Se você tiver um cenário que exija IAM usuários ou um usuário root Conta da AWS, ative MFA para obter segurança adicional. Para exigir MFA quando API as operações são chamadas, adicione MFA condições às suas políticas. Para obter mais informações, consulte APIAcesso seguro MFA no Guia do IAM usuário.

Para obter mais informações sobre as melhores práticas emIAM, consulte as melhores práticas de segurança IAM no Guia IAM do usuário.

Usando o Timestream para console LiveAnalytics

O Timestream for LiveAnalytics não exige permissões específicas para acessar o Amazon Timestream para console. LiveAnalytics Você precisa de pelo menos permissões de somente leitura para listar e visualizar detalhes sobre o Timestream dos LiveAnalytics recursos em sua conta. AWS Se você criar uma política baseada em identidade que seja mais restritiva do que as permissões mínimas exigidas, o console não funcionará conforme planejado para entidades (IAMusuários ou funções) com essa política.

Permitir que usuários visualizem suas próprias permissões

Este exemplo mostra como você pode criar uma política que permita IAM aos usuários visualizar as políticas embutidas e gerenciadas que estão anexadas à identidade do usuário. Essa política inclui permissões para concluir essa ação no console ou programaticamente usando o AWS CLI ou. 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": "*"
        }
    ]
}
```

Operações comuns no Timestream para LiveAnalytics

Abaixo estão exemplos de IAM políticas que permitem operações comuns no Timestream for LiveAnalytics service.

Tópicos

- Permitindo todas as operações
- Permitindo SELECT operações
- Permitindo SELECT operações em vários recursos
- Permitindo operações de metadados
- Permitindo INSERT operações
- Permitindo CRUD operações
- Cancelar consultas e selecionar dados sem especificar recursos
- Crie, descreva, exclua e descreva um banco de dados
- Limitar bancos de dados listados por tag {"Owner": "${username}"}
- Listar todas as tabelas em um banco de dados
- Crie, descreva, exclua, atualize e selecione em uma tabela
- Limitar uma consulta por tabela

Permitindo todas as operações

A seguir está um exemplo de política que permite todas as operações no Timestream para. LiveAnalytics

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "timestream:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Permitindo SELECT operações

O exemplo de política a seguir permite consultas no SELECT estilo -style em um recurso específico.

> **Note**
>
> <account_ID>Substitua pelo ID da sua conta Amazon.

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

Permitindo SELECT operações em vários recursos

O exemplo de política a seguir permite consultas no `SELECT` estilo -style em vários recursos.

> **Note**
>
> `<account_ID>`Substitua pelo ID da sua conta Amazon.

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

Permitindo operações de metadados

O exemplo de política a seguir permite que o usuário realize consultas de metadados, mas não permite que o usuário execute operações que leiam ou gravem dados reais no Timestream for. LiveAnalytics

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream:DescribeEndpoints",
                "timestream:DescribeTable",
                "timestream:ListMeasures",
                "timestream:SelectValues",
                "timestream:ListTables",
                "timestream:ListDatabases",
                "timestream:CancelQuery"
            ],
            "Resource": "*"
        }
    ]
}
```

Permitindo INSERT operações

O exemplo de política a seguir permite que um usuário execute uma `INSERT` operação `database/sampleDB/table/DevOps` na conta`<account_id>`.

> **Note**
>
> `<account_ID>`Substitua pelo ID da sua conta Amazon.

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

Permitindo CRUD operações

O exemplo de política a seguir permite que um usuário execute CRUD operações no Timestream para. LiveAnalytics

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream:DescribeEndpoints",
                "timestream:CreateTable",
                "timestream:DescribeTable",
                "timestream:CreateDatabase",
                "timestream:DescribeDatabase",
                "timestream:ListTables",
                "timestream:ListDatabases",
                "timestream:DeleteTable",
                "timestream:DeleteDatabase",
                "timestream:UpdateTable",
                "timestream:UpdateDatabase"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

Cancelar consultas e selecionar dados sem especificar recursos

O exemplo de política a seguir permite que um usuário cancele consultas e realize `Select` consultas em dados que não exigem especificação de recursos:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream:SelectValues",
                "timestream:CancelQuery"
            ],
            "Resource": "*"
        }
    ]
}
```

Crie, descreva, exclua e descreva um banco de dados

O exemplo de política a seguir permite que um usuário crie, descreva, exclua e descreva o banco de dados`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"
        }
    ]
}
```

Limitar bancos de dados listados por tag **`{"Owner": "${username}"}`**

O exemplo de política a seguir permite que um usuário liste todos os bancos de dados marcados com o par de valores-chave`{"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}"
                }
            }
        }
    ]
}
```

Listar todas as tabelas em um banco de dados

O exemplo de política a seguir para listar todas as tabelas no banco de dados`sampleDB`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "timestream:ListTables"
            ],
            "Resource": "arn:aws:timestream:us-east-1:<account_ID>:database/sampleDB/"
        }
    ]
}
```

Crie, descreva, exclua, atualize e selecione em uma tabela

O exemplo de política a seguir permite que um usuário crie tabelas, descreva tabelas, exclua tabelas, atualize tabelas e realize `Select` consultas em uma tabela `DevOps` no banco de dados`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"
        }
    ]
}
```

Limitar uma consulta por tabela

O exemplo de política a seguir permite que um usuário consulte todas as tabelas, exceto `DevOps` no banco de dados`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"
        }
    ]
}
```

Cronograma para acesso a LiveAnalytics recursos com base em tags

Você pode usar condições em sua política baseada em identidade para controlar o acesso ao Timestream para LiveAnalytics recursos com base em tags. Esta seção fornece alguns exemplos.

O exemplo a seguir mostra como você pode criar uma política que conceda permissões a um usuário para visualizar uma tabela se o `Owner` da tabela contiver o valor do nome do usuário em questão.

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

Você pode anexar essa política aos IAM usuários da sua conta. Se um usuário chamado `richard-roe` tentar visualizar um Timestream para uma LiveAnalytics tabela, a tabela deverá ser marcada com `Owner=richard-roe` ou. `owner=richard-roe` Caso contrário, ele terá o acesso negado. A chave da tag de condição `Owner` corresponde a `Owner` e a `owner` porque os nomes das chaves de condição não fazem distinção entre maiúsculas e minúsculas. Para obter mais informações, consulte Elementos da IAM JSON política: condição no Guia IAM do usuário.

A política a seguir concede permissões a um usuário para criar tabelas com tags se a tag passada na solicitação tiver uma chave `Owner` e um valor`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}"
                }
            }
        }
    ]
}
```

A política abaixo permite o uso do `DescribeDatabase` API em qualquer banco de dados que tenha a `env` tag definida como `dev` ou`test`:

```
{ "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowDescribeEndpoints",
      "Effect": "Allow",
      "Action": [
        "timestream:DescribeEndpoints"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowDevTestAccess",
      "Effect": "Allow",
      "Action": [
        "timestream:DescribeDatabase"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
```

```
          "timestream:tag/env": [
              "dev",
              "test"
          ]
        }
      }
    }
  ]
}
{ "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowTagAccessForDevResources",
      "Effect": "Allow",
      "Action": [
        "timestream:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/env": [
            "test",
            "dev"
          ]
        }
      }
    }
  ]
}
```

Essa política usa uma `Condition` chave para permitir que uma tag que tenha a chave `env` e o valor de `testqa`, ou `dev` seja adicionada a um recurso.

Consultas programadas

Listar, excluir, atualizar, executar ScheduledQuery

O exemplo de política a seguir permite que um usuário liste, exclua, atualize e execute consultas agendadas.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": [
                "timestream:DeleteScheduledQuery",
                "timestream:ExecuteScheduledQuery",
                "timestream:UpdateScheduledQuery",
                "timestream:ListScheduledQueries",
                "timestream:DescribeEndpoints"
            ],
            "Resource": "*"
        }
    ]
}
```

CreateScheduledQuery usando uma KMS chave gerenciada pelo cliente

O exemplo de política a seguir permite que um usuário crie uma consulta agendada criptografada usando uma KMS chave gerenciada pelo cliente; *<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 usando uma KMS chave gerenciada pelo cliente

O exemplo de política a seguir permite que um usuário descreva uma consulta agendada que foi criada usando uma KMS chave gerenciada pelo cliente; *<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"
        }
    ]
}
```

Permissões da função de execução (usando uma KMS chave gerenciada pelo cliente para consultas agendadas e SSE - KMS para relatórios de erros)

Anexe o exemplo de política a seguir à IAM função especificada no `ScheduledQueryExecutionRoleArn` parâmetro, da `CreateScheduledQuery` API que usa a KMS chave gerenciada pelo cliente para a criptografia da consulta agendada e a `SSE-KMS` criptografia para relatórios de erros.

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

Relação de confiança da função de execução

A seguir está a relação de confiança para a IAM função especificada no `ScheduledQueryExecutionRoleArn` parâmetro do `CreateScheduledQuery`API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "timestream.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Permitir acesso a todas as consultas agendadas criadas em uma conta

Anexe o exemplo de política a seguir à IAM função especificada no `ScheduledQueryExecutionRoleArn` parâmetro, do `CreateScheduledQuery`API, para permitir o acesso a todas as consultas agendadas criadas em uma conta *`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/*"
                }
            }
        }
    ]
}
```

Permitir acesso a todas as consultas agendadas com um nome específico

Anexe o exemplo de política a seguir à IAM função especificada no `ScheduledQueryExecutionRoleArn` parâmetro, do `CreateScheduledQuery`API, para permitir o acesso a todas as consultas agendadas com um nome que comece com *`Scheduled_Query_Name`*, dentro da conta *`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*"
                }
            }
```

```
        }
    ]
}
```

Solução de problemas do Amazon Timestream LiveAnalytics para identidade e acesso

Use as informações a seguir para ajudá-lo a diagnosticar e corrigir problemas comuns que você pode encontrar ao trabalhar com o Timestream para e. LiveAnalytics IAM

Tópicos

- Não estou autorizado a realizar uma ação no Timestream para LiveAnalytics
- Não estou autorizado a realizar iam: PassRole
- Quero permitir que pessoas fora da minha AWS conta acessem meu Timestream para obter recursos LiveAnalytics

Não estou autorizado a realizar uma ação no Timestream para LiveAnalytics

Se isso AWS Management Console indicar que você não está autorizado a realizar uma ação, entre em contato com o administrador para obter ajuda. Caso seu administrador seja a pessoa que forneceu suas credenciais de início de sessão.

O exemplo de erro a seguir ocorre quando o `mateojackson` IAM usuário tenta usar o console para ver detalhes sobre um *`table`* mas não tem `timestream:`*`Select`* permissões para a tabela.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 timestream:Select on resource: mytable
```

Neste caso, Mateo pede ao administrador para atualizar suas políticas para permitir a ele o acesso ao recurso *`mytable`* usando a ação `timestream:`*`Select`*.

Não estou autorizado a realizar iam: PassRole

Se você receber um erro informando que não está autorizado a realizar a `iam:PassRole` ação, suas políticas devem ser atualizadas para permitir que você passe uma função para a Timestream. LiveAnalytics

Alguns Serviços da AWS permitem que você passe uma função existente para esse serviço em vez de criar uma nova função de serviço ou uma função vinculada ao serviço. Para fazer isso, é preciso ter permissões para passar o perfil para o serviço.

O exemplo de erro a seguir ocorre quando um IAM usuário chamado `marymajor` tenta usar o console para realizar uma ação no Timestream for. LiveAnalytics No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Quero permitir que pessoas fora da minha AWS conta acessem meu Timestream para obter recursos LiveAnalytics

Você pode criar um perfil que os usuários de outras contas ou pessoas fora da sua organização podem usar para acessar seus recursos. Você pode especificar quem é confiável para assumir o perfil. Para serviços que oferecem suporte a políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se o Timestream for LiveAnalytics oferece suporte a esses recursos, consulte. Como o Amazon LiveAnalytics Timestream for funciona com IAM
- Para saber como fornecer acesso aos seus recursos em todos os Contas da AWS que você possui, consulte Fornecer acesso a um IAM usuário em outro Conta da AWS de sua propriedade no Guia do IAM usuário.
- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte Fornecer Contas da AWS acesso a terceiros no Guia do IAM usuário.
- Para saber como fornecer acesso por meio da federação de identidades, consulte Fornecendo acesso a usuários autenticados externamente (federação de identidades) no Guia do IAMusuário.
- Para saber a diferença entre usar funções e políticas baseadas em recursos para acesso entre contas, consulte Acesso a recursos entre contas IAM no Guia do IAM usuário.

Registro e monitoramento no Timestream para LiveAnalytics

O monitoramento é uma parte importante da manutenção da confiabilidade, disponibilidade e desempenho do Timestream LiveAnalytics e de suas AWS soluções. Você deve coletar dados de monitoramento de todas as partes da sua AWS solução para poder depurar com mais facilidade uma falha multiponto, caso ocorra. No entanto, antes de começar a monitorar o Timestream LiveAnalytics, você deve criar um plano de monitoramento que inclua respostas às seguintes perguntas:

- Quais são seus objetivos de monitoramento?
- Quais recursos você vai monitorar?
- Com que frequência você vai monitorar esses recursos?
- Quais ferramentas de monitoramento você usará?
- Quem realizará o monitoramento das tarefas?
- Quem deve ser notificado quando algo der errado?

A próxima etapa é estabelecer uma linha de base para o Timestream normal para LiveAnalytics desempenho em seu ambiente, medindo o desempenho em vários momentos e sob diferentes condições de carga. Ao monitorar o Timestream LiveAnalytics, armazene dados históricos de monitoramento para que você possa compará-los com os dados de desempenho atuais, identificar padrões normais de desempenho e anomalias de desempenho e criar métodos para resolver problemas.

Para estabelecer uma linha de base, você deve, no mínimo, monitorar os seguintes itens:

- Erros do sistema, para que você possa determinar se alguma solicitação resultou em erro.

Tópicos

- Ferramentas de monitoramento
- Registrando o Timestream para LiveAnalytics API chamadas com AWS CloudTrail

Ferramentas de monitoramento

AWS fornece várias ferramentas que você pode usar para monitorar o Timestream. LiveAnalytics É possível configurar algumas dessas ferramentas para fazer o monitoramento em seu lugar, e, ao mesmo tempo, algumas das ferramentas exigem intervenção manual. Recomendamos que as tarefas de monitoramento sejam automatizadas ao máximo possível.

Tópicos

- Ferramentas de monitoramento automatizadas
- Ferramentas de monitoramento manual

Ferramentas de monitoramento automatizadas

Você pode usar as seguintes ferramentas de monitoramento automatizado para assistir ao Timestream LiveAnalytics e relatar quando algo está errado:

- Amazon CloudWatch Alarms — Observe uma única métrica em um período especificado por você e execute uma ou mais ações com base no valor da métrica em relação a um determinado limite em vários períodos. A ação é uma notificação enviada para um tópico do Amazon Simple Notification Service (AmazonSNS) ou para uma política do Amazon EC2 Auto Scaling. CloudWatch os alarmes não invocam ações simplesmente porque estão em um determinado estado; o estado deve ter sido alterado e mantido por um determinado número de períodos. Para obter mais informações, consulte Monitoramento com a Amazon CloudWatch.

Ferramentas de monitoramento manual

Outra parte importante do monitoramento do Timestream LiveAnalytics envolve o monitoramento manual dos itens que os CloudWatch alarmes não cobrem. O Timestream para LiveAnalytics, CloudWatch, Trusted Advisor, e outros AWS Management Console painéis fornecem uma at-a-glance visão do estado do seu ambiente. AWS

- A página CloudWatch inicial mostra o seguinte:
  - Alertas e status atual
  - Gráficos de alertas e recursos
  - Estado de integridade do serviço

  Além disso, você pode usar CloudWatch para fazer o seguinte:
  - Crie painéis personalizados para monitorar os serviços com os quais você se preocupa.
  - Colocar em gráfico dados de métrica para solucionar problemas e descobrir tendências
  - Pesquise e navegue em todas as suas métricas AWS de recursos
  - Criar e editar alertas para ser notificado sobre problemas

Registrando o Timestream para LiveAnalytics API chamadas com AWS CloudTrail

O Timestream for LiveAnalytics está integrado com AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, função ou AWS serviço no Timestream for. LiveAnalytics CloudTrail captura API chamadas de Data Definition Language (DDL) para Timestream for LiveAnalytics as events. As chamadas capturadas incluem chamadas do Timestream para LiveAnalytics console e chamadas de código para o Timestream para operações. LiveAnalytics API Se você criar uma trilha, poderá habilitar a entrega contínua de CloudTrail eventos para um bucket do Amazon Simple Storage Service (Amazon S3), incluindo eventos para o Timestream for. LiveAnalytics Se você não configurar uma trilha, ainda poderá ver os eventos mais recentes no CloudTrail console no Histórico de eventos. Usando as informações coletadas por CloudTrail, você pode determinar a solicitação feita à Timestream LiveAnalytics, o endereço IP do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e detalhes adicionais.

Para saber mais sobre isso CloudTrail, consulte o Guia AWS CloudTrail do usuário.

Timestream para obter informações em LiveAnalytics CloudTrail

CloudTrail é ativado em sua AWS conta quando você cria a conta. Quando a atividade ocorre no Timestream for LiveAnalytics, essa atividade é registrada em um CloudTrail evento junto com outros eventos de AWS serviço no histórico de eventos. Você pode visualizar, pesquisar e baixar eventos recentes em sua AWS conta. Para obter mais informações, consulte Visualização de eventos com histórico de CloudTrail eventos.

> **Warning**
>
> Atualmente, o Timestream for LiveAnalytics gera CloudTrail eventos para todo o gerenciamento e `Query` API operações, mas não gera eventos para e. `WriteRecords` `DescribeEndpoints` APIs

Para um registro contínuo dos eventos em sua AWS conta, incluindo eventos do Timestream for LiveAnalytics, crie uma trilha. Uma trilha permite CloudTrail entregar arquivos de log para um bucket do Amazon S3. Por padrão, quando você cria uma trilha no console, a trilha se aplica a todas as AWS regiões. A trilha registra eventos de todas as regiões na AWS partição e entrega os arquivos de log ao bucket do Amazon S3 que você especificar. Além disso, você pode configurar outros AWS serviços para analisar e agir com base nos dados de eventos coletados nos CloudTrail registros.

Para obter mais informações, consulte os seguintes tópicos no Guia do usuário do AWS CloudTrail :

- Visão geral da criação de uma trilha
- CloudTrail Serviços e integrações compatíveis
- Configurando as SNS notificações da Amazon para CloudTrail
- Recebendo arquivos de CloudTrail log de várias regiões
- Recebendo arquivos de CloudTrail log de várias contas
- Registrando eventos de dados

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar:

- Se a solicitação foi feita com credenciais de usuário root ou AWS Identity and Access Management (IAM)
- Se a solicitação foi feita com credenciais de segurança temporárias de um perfil ou de um usuário federado
- Se a solicitação foi feita por outro AWS serviço

Para obter mais informações, consulte o CloudTrail userIdentityElemento.

Para `Query` API eventos:

- Crie uma trilha que receba todos os eventos ou selecione eventos com Timestream para o tipo `AWS::Timestream::Database` de LiveAnalytics recurso ou. `AWS::Timestream::Table`
- `Query`APIsolicitações que não acessam nenhum banco de dados ou tabela ou que resultam em uma exceção de validação devido a uma string de consulta malformada são registradas CloudTrail com um tipo de recurso `AWS::Timestream::Database` e um ARN valor de:

  ```
  arn:aws:timestream:(region):(accountId):database/NO_RESOURCE_ACCESSED
  ```

  Esses eventos são entregues somente para trilhas que recebem eventos com o tipo de recurso`AWS::Timestream::Database`.

Resiliência no Amazon Timestream Live Analytics

A infraestrutura AWS global é construída em torno de AWS regiões e zonas de disponibilidade. AWS As regiões fornecem várias zonas de disponibilidade fisicamente separadas e isoladas,

conectadas a redes de baixa latência, alta taxa de transferência e alta redundância. Com as zonas de disponibilidade, é possível projetar e operar aplicativos e bancos de dados que automaticamente executam o failover entre as zonas sem interrupção. As zonas de disponibilidade são altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data centers tradicionais.

Para obter mais informações sobre AWS regiões e zonas de disponibilidade, consulte Infraestrutura AWS global.

Para obter informações sobre a funcionalidade de proteção de dados do Timestream disponível por meio de AWS Backup, consulte. Trabalhando com AWS Backup

Segurança da infraestrutura no Amazon Timestream Live Analytics

Como um serviço gerenciado, o Amazon Timestream Live Analytics é protegido pelos AWS procedimentos globais de segurança de rede descritos no whitepaper Amazon Web Services: Visão geral dos processos de segurança.

Você usa API chamadas AWS publicadas para acessar o Timestream Live Analytics pela rede. Os clientes devem oferecer suporte ao Transport Layer Security (TLS) 1.0 ou posterior. Recomendamos TLS 1.2 ou posterior. Os clientes também devem oferecer suporte a pacotes de criptografia com sigilo direto perfeito (), como Ephemeral Diffie-Hellman (PFS) ou Elliptic Curve Ephemeral Diffie-Hellman (). DHE ECDHE A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Além disso, as solicitações devem ser assinadas usando uma ID de chave de acesso e uma chave de acesso secreta associada a um IAM principal. Ou você pode usar o AWS Security Token Service (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

O Timestream Live Analytics é arquitetado para que seu tráfego seja isolado AWS na região específica em que sua instância do Timestream Live Analytics reside.

Análise de configuração e vulnerabilidade no Timestream

A configuração e os controles de TI são uma responsabilidade compartilhada entre você AWS e você, nosso cliente. Para obter mais informações, consulte o modelo de responsabilidade AWS compartilhada. Além do modelo de responsabilidade compartilhada, o Timestream para LiveAnalytics usuários deve estar ciente do seguinte:

- É da responsabilidade do cliente corrigir as aplicações de clientes com as dependências relevantes do lado do cliente.
- Os clientes devem considerar o teste de penetração, se apropriado (consulte testes de https://aws.amazon.com/security/penetração/.)

Resposta a incidentes no Timestream para LiveAnalytics

O Amazon Timestream LiveAnalytics para incidentes de serviço é relatado no Personal Health Dashboard. Você pode aprender mais sobre o painel e AWS Health aqui.

Timestream para LiveAnalytics suportar o uso de relatórios. AWS CloudTrail Para obter mais informações, consulte Registrando o Timestream para LiveAnalytics API chamadas com AWS CloudTrail.

VPCpontos finais ()AWS PrivateLink

Você pode estabelecer uma conexão privada entre você VPC e o Amazon Timestream criando um endpoint LiveAnalytics de interface. VPC Os endpoints de interface são alimentados por AWS PrivateLinkuma tecnologia que permite acessar o Timestream de forma privada LiveAnalytics APIs sem um gateway de internet, NAT dispositivo, VPN conexão ou conexão Direct AWS Connect. Suas instâncias VPC não precisam de endereços IP públicos para se comunicar com o Timestream. LiveAnalytics APIs O tráfego entre você VPC e o Timestream for LiveAnalytics não sai da rede Amazon.

Cada endpoint de interface é representado por uma ou mais Interfaces de Rede Elástica nas sub-redes. Para obter mais informações sobre VPC endpoints de interface, consulte VPCEndpoints de interface (AWS PrivateLink) no Guia VPC do usuário da Amazon.

Para começar a usar o Timestream for LiveAnalytics and VPC endpoints, fornecemos informações sobre considerações específicas para o Timestream for LiveAnalytics with VPC endpoints, a criação de um endpoint de interface para o Timestream for, a criação de uma política de VPC endpoint para o Timestream for LiveAnalytics e o uso do VPC cliente Timestream (para Write ou LiveAnalytics Query) com endpoints. SDK VPC

Tópicos

- Como os VPC endpoints funcionam com o Timestream
- Criando um VPC endpoint de interface para o Timestream for LiveAnalytics

- Criação de uma política VPC de endpoint para o Timestream for LiveAnalytics

Como os VPC endpoints funcionam com o Timestream

Quando você cria um VPC endpoint para acessar o Timestream Write ou o Timestream QuerySDK, todas as solicitações são roteadas para endpoints dentro da rede Amazon e não acessam a Internet pública. Mais especificamente, suas solicitações são encaminhadas para os endpoints de gravação e consulta da célula para a qual sua conta foi mapeada para uma determinada região. Para saber mais sobre a arquitetura celular e os endpoints específicos da célula do Timestream, você pode consultar. Arquitetura celular Por exemplo, suponha que sua conta tenha sido mapeada para `cell1` in `us-west-2` e você tenha configurado endpoints de VPC interface para writes (`ingest-cell1.timestream.us-west-2.amazonaws.com`) e queries (). `query-cell1.timestream.us-west-2.amazonaws.com` Nesse caso, todas as solicitações de gravação enviadas usando esses endpoints permanecerão inteiramente na rede da Amazon e não acessarão a Internet pública.

Considerações sobre endpoints do VPC Timestream

Considere o seguinte ao criar um VPC endpoint para o Timestream:

- Antes de configurar um VPC endpoint de interface para o Timestream for LiveAnalytics, certifique-se de revisar as propriedades e limitações do endpoint da interface no Guia do usuário da Amazon. VPC
- Timestream for LiveAnalytics suporta a realização de chamadas para todas as suas API ações a partir de seu. VPC
- VPCas políticas de endpoint são suportadas pelo Timestream for. LiveAnalytics Por padrão, o acesso total ao Timestream for LiveAnalytics é permitido por meio do endpoint. Para obter mais informações, consulte Controle do acesso a serviços com VPC endpoints no Guia do VPC usuário da Amazon.
- Devido à arquitetura do Timestream, o acesso às ações de Gravação e Consulta requer a criação de dois endpoints de VPC interface, um para cada. SDK Além disso, você deve especificar um endpoint de célula (você só poderá criar um endpoint para a célula Timestream para a qual você está mapeado). Informações detalhadas podem ser encontradas na LiveAnalytics seção Criar um VPC endpoint de interface para Timestream for deste guia.

Agora que você entende como o Timestream for LiveAnalytics funciona com VPC endpoints, crie um endpoint de interface VPC para o Timestream for. LiveAnalytics

Criando um VPC endpoint de interface para o Timestream for LiveAnalytics

Você pode criar um VPCendpoint de interface para o LiveAnalytics serviço Timestream for usando o VPC console da Amazon ou o AWS Command Line Interface ().AWS CLI Para criar um VPC endpoint para o Timestream, conclua as etapas específicas do Timestream descritas abaixo.

> **Note**
>
> Antes de concluir as etapas abaixo, certifique-se de compreender as considerações específicas dos endpoints do Timestream VPC.

Construindo um nome de serviço de VPC endpoint usando sua célula Timestream

Devido à arquitetura exclusiva do Timestream, terminais de VPC interface separados devem ser criados para cada um SDK (gravação e consulta). Além disso, você deve especificar um ponto final da célula Timestream (você só poderá criar um endpoint para a célula Timestream para a qual você está mapeado). Para usar os VPC endpoints de interface para se conectar diretamente ao Timestream de dentro do seuVPC, conclua as etapas abaixo:

1. Primeiro, encontre um endpoint de célula Timestream disponível. Para encontrar um endpoint de célula disponível, use a `DescribeEndpointsação` (disponível por meio de Gravação e ConsultaAPIs) para listar os endpoints de célula disponíveis em sua conta Timestream. Veja o exemplo para obter mais detalhes.
2. Depois de selecionar um endpoint de célula para usar, crie uma string de endpoint de VPC interface para o Timestream Write ou Query: API
   - Para escreverAPI:

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

   - Para a consultaAPI:

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

   where *<region>* é um código de AWS região válido e *<cell>* é um dos endereços de endpoint da célula (como `cell1` ou `cell2`) retornados no objeto Endpoints pela DescribeEndpoints ação. Veja o exemplo para obter mais detalhes.

3. Agora que você criou um nome de serviço de VPC endpoint, crie um endpoint de interface. Quando solicitado a fornecer um nome de serviço de VPC endpoint, use o nome do serviço de VPC endpoint que você criou na Etapa 2.

Exemplo: Construindo seu nome de serviço VPC de endpoint

No exemplo a seguir, a `DescribeEndpoints` ação é executada AWS CLI usando a Gravação API na `us-west-2` região:

```
aws timestream-write describe-endpoints --region us-west-2
```

Esse comando retornará a seguinte saída:

```
{
    "Endpoints": [
        {
            "Address": "ingest-cell1.timestream.us-west-2.amazonaws.com",
            "CachePeriodInMinutes": 1440
        }
    ]
}
```

Nesse caso, *cell1* é o *<cell>* e *us-west-2* é o *<region>*. Portanto, o nome do serviço de VPC endpoint resultante terá a seguinte aparência:

```
com.amazonaws.us-west-2.timestream.ingest-cell1
```

Agora que você criou um VPC endpoint de interface para Timestream for LiveAnalytics, crie uma política de VPC endpoint para Timestream for. LiveAnalytics

Criação de uma política VPC de endpoint para o Timestream for LiveAnalytics

Você pode anexar uma política de endpoint ao seu VPC endpoint que controla o acesso ao Timestream para. LiveAnalytics Essa política especifica as seguintes informações:

- A entidade principal que pode executar ações.
- As ações que podem ser executadas.
- Os recursos sobre os quais as ações podem ser realizadas.

Para obter mais informações, consulte Controle do acesso a serviços com VPC endpoints no Guia do VPC usuário da Amazon.

Exemplo: política de VPC endpoint para Timestream for actions LiveAnalytics

Veja a seguir um exemplo de uma política de endpoint para Timestream for. LiveAnalytics Quando anexada a um endpoint, essa política concede acesso ao Timestream listado para LiveAnalytics ações (nesse caso, `ListDatabases`) para todos os diretores em todos os recursos.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "timestream:ListDatabases"
         ],
         "Resource":"*"
      }
   ]
}
```

Melhores práticas de segurança para o Amazon Timestream for LiveAnalytics

O Amazon Timestream LiveAnalytics for fornece vários recursos de segurança a serem considerados ao desenvolver e implementar suas próprias políticas de segurança. As melhores práticas a seguir são diretrizes gerais e não representam uma solução completa de segurança. Como essas melhores práticas podem não ser adequadas ou suficientes para o seu ambiente, trate-as como considerações úteis em vez de prescrições.

Tópicos

- Timestream para as melhores práticas de LiveAnalytics segurança preventiva

Timestream para as melhores práticas de LiveAnalytics segurança preventiva

As melhores práticas a seguir podem ajudá-lo a antecipar e evitar incidentes de segurança no Timestream for. LiveAnalytics

Criptografia em repouso

O Timestream for LiveAnalytics criptografa em repouso todos os dados do usuário armazenados em tabelas usando chaves de criptografia armazenadas em AWS Key Management Service ().AWS KMS Isso oferece uma camada de proteção de dados adicional ao proteger seus dados contra acesso não autorizado ao armazenamento subjacente.

O Timestream for LiveAnalytics usa uma única chave padrão de serviço (AWS própriaCMK) para criptografar todas as suas tabelas. Se essa chave não existir, ela será criada para você. As chaves padrão do serviço não podem ser desabilitadas. Para obter mais informações, consulte Timestream para LiveAnalytics criptografia em repouso.

Use IAM funções para autenticar o acesso ao Timestream para LiveAnalytics

Para que usuários, aplicativos e outros AWS serviços acessem o Timestream LiveAnalytics, eles devem incluir AWS credenciais válidas em suas solicitações. AWS API Você não deve armazenar AWS credenciais diretamente no aplicativo ou na EC2 instância. Essas são credenciais de longo prazo que não são automaticamente alternadas e, portanto, podem ter impacto comercial significativo se forem comprometidas. Uma IAM função permite que você obtenha chaves de acesso temporárias que podem ser usadas para acessar AWS serviços e recursos.

Para obter mais informações, consulte Funções do IAM.

Use IAM políticas do Timestream para autorização básica LiveAnalytics

Ao conceder permissões, você decide quem as está recebendo, para qual Timestream LiveAnalytics APIs elas estão recebendo permissões e as ações específicas que você deseja permitir nesses recursos. A implementação do privilégio mínimo é fundamental para reduzir o risco de segurança e o impacto que pode resultar de erros ou usuários mal-intencionados.

Anexe políticas de permissões às IAM identidades (ou seja, usuários, grupos e funções) e, assim, conceda permissões para realizar operações no Timestream para recursos. LiveAnalytics

Para isso, você pode usar o seguinte:

- AWS políticas gerenciadas (predefinidas)
- Políticas gerenciadas pelo cliente
- Autorização baseada em tags

Considere utilizar a criptografia do lado do cliente

Se você armazenar dados sensíveis ou confidenciais no Timestream for LiveAnalytics, talvez queira criptografar esses dados o mais próximo possível de sua origem para que sejam

protegidos durante todo o ciclo de vida. Criptografar seus dados confidenciais em trânsito e em repouso ajuda a garantir que seus dados em texto simples não estejam disponíveis para terceiros.

Como trabalhar com outros serviços do

O Amazon Timestream LiveAnalytics for se integra a uma variedade de serviços e ferramentas populares AWS de terceiros. Atualmente, o Timestream for LiveAnalytics suporta integrações com o seguinte:

Tópicos

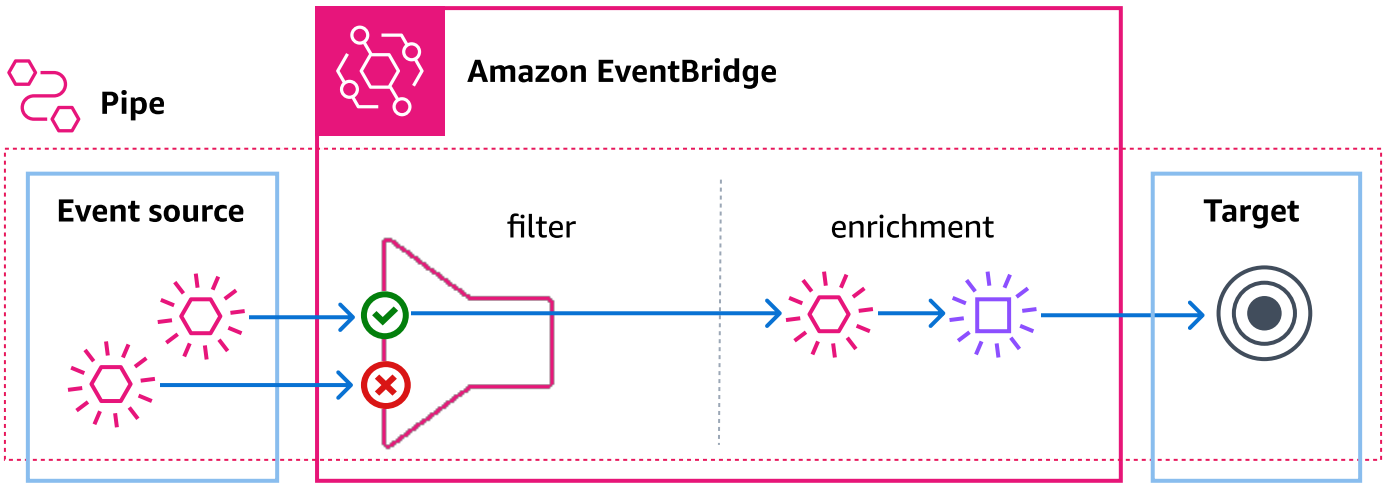
- Amazon DynamoDB
- AWS Lambda
- AWS IoT Core
- Amazon Managed Service for Apache Flink
- Amazon Kinesis
- Amazon MQ
- Amazon MSK
- Amazon QuickSight
- Amazon SageMaker
- Amazon SQS
- Usando DBeaver para trabalhar com o Amazon Timestream
- Grafana
- Usando SquaredUp para trabalhar com o Amazon Timestream
- Telegraf de código aberto
- JDBC
- ODBC
- VPCpontos finais ()AWS PrivateLink

Amazon DynamoDB

Usando EventBridge Pipes para enviar dados do DynamoDB para Timestream

Você pode usar o EventBridge Pipes para enviar dados de um stream do DynamoDB para um Amazon Timestream como tabela. LiveAnalytics

Os tubos são destinados a point-to-point integrações entre fontes e alvos suportados, com suporte para transformações e enriquecimento avançados. Os Pipes reduzem a necessidade de conhecimento especializado e código de integração ao desenvolver arquiteturas orientadas a eventos. Para configurar um pipe, a origem é escolhida, adiciona filtragem opcional, define o enriquecimento opcional e escolhe o destino para os dados do evento.



Para obter mais informações sobre EventBridge tubos, consulte EventBridge Tubos no Guia EventBridge do usuário. Para obter informações sobre como configurar um canal para entregar eventos a um Amazon LiveAnalytics Timestream para tabela EventBridge , consulte Especificações de destino do Pipes.

AWS Lambda

Você pode criar funções Lambda que interagem com o Timestream for. LiveAnalytics Por exemplo, você pode criar uma função Lambda que seja executada em intervalos regulares para executar uma consulta no Timestream e enviar uma SNS notificação com base nos resultados da consulta que satisfaçam um ou mais critérios. Para saber mais sobre o Lambda, consulte a documentação do Lambda AWS .

Tópicos

- Crie funções do AWS Lambda usando o Amazon Timestream for com Python LiveAnalytics
- Crie funções AWS Lambda usando o Amazon Timestream for com LiveAnalytics JavaScript
- Crie funções do AWS Lambda usando o Amazon Timestream for with Go LiveAnalytics
- Crie funções do AWS Lambda usando o Amazon Timestream for com C# LiveAnalytics

Crie funções do AWS Lambda usando o Amazon Timestream for com Python LiveAnalytics

Para criar funções do AWS Lambda usando o Amazon Timestream LiveAnalytics for com Python, siga as etapas abaixo.

1. Crie uma IAM função para o Lambda assumir que concederá as permissões necessárias para acessar o Timestream Service, conforme descrito em. Forneça um Timestream para acesso LiveAnalytics
2. Edite a relação de confiança da IAM função para adicionar o serviço Lambda. Você pode usar os comandos abaixo para atualizar uma função existente para que o AWS Lambda possa assumi-la:

    a. Crie o documento de política de confiança:

```
cat > Lambda-Role-Trust-Policy.json << EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "lambda.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

    b. Atualize a função da etapa anterior com o documento de confiança

```
aws iam update-assume-role-policy --role-name <name_of_the_role_from_step_1> --
policy-document file://Lambda-Role-Trust-Policy.json
```

As referências relacionadas estão em TimestreamWriteTimestreamQuerye.

Crie funções AWS Lambda usando o Amazon Timestream for com LiveAnalytics JavaScript

Para criar funções do AWS Lambda usando o Amazon Timestream LiveAnalytics for JavaScript with, siga as instruções descritas aqui.

As referências relacionadas estão em Timestream Write Client - AWS SDK para JavaScript v3 e Timestream Query Client - para v3. AWS SDK JavaScript

Crie funções do AWS Lambda usando o Amazon Timestream for with Go LiveAnalytics

Para criar funções do AWS Lambda usando o Amazon Timestream LiveAnalytics for with Go, siga as instruções descritas aqui.

As referências relacionadas estão em timestreamwrite e timestreamquery.

Crie funções do AWS Lambda usando o Amazon Timestream for com C# LiveAnalytics

Para criar funções do AWS Lambda usando o Amazon Timestream LiveAnalytics para C#, siga as instruções descritas aqui.

As referências relacionadas estão na Amazon. TimestreamWritee Amazon. TimestreamQuery.

AWS IoT Core

Você pode coletar dados de dispositivos de IoT usando o IoT Core e rotear os dados AWS para o Amazon Timestream por meio de ações de regras do IoT Core. AWS As ações das regras de IoT especificam o que fazer quando uma regra é acionada. Você pode definir ações para enviar dados para uma tabela do Amazon Timestream, um banco de dados do Amazon DynamoDB e invocar uma função Lambda. AWS

A ação Timestream nas Regras de IoT é usada para inserir dados de mensagens recebidas diretamente no Timestream. A ação analisa os resultados da declaração do IoT SQL Core e

armazena dados no Timestream. Os nomes dos campos do conjunto de SQL resultados retornado são usados como a medida: :nome e o valor do campo é a medida: :valor.

Por exemplo, considere a SQL declaração e o exemplo de carga útil da mensagem:

```
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
}
```

Se uma ação de regra do IoT Core para Timestream for criada com a SQL declaração acima, dois registros serão adicionados ao Timestream com nomes de medidas temperatura e umidade e valores de medida de 24,04 e 43,605, respectivamente.

Você pode modificar o nome da medida de um registro que está sendo adicionado ao Timestream usando o operador AS na SELECT instrução. A SQL declaração abaixo criará um registro com o nome da mensagem temp em vez de temperature.

O tipo de dados da medida é inferido do tipo de dados do valor da carga da mensagem. JSONtipos de dados como inteiro, duplo, booleano e string são mapeados para os tipos de dados Timestream deBIGINT,, e, DOUBLE respectivamente. BOOLEAN VARCHAR Os dados também podem ser forçados a tipos de dados específicos usando a função cast (). Você pode especificar a data e hora da medida. Se o carimbo de data/hora for deixado em branco, a hora em que a entrada foi processada será usada.

Você pode consultar a documentação de ação das regras do Timestream para obter detalhes adicionais.

Para criar uma ação de regra do IoT Core para armazenar dados no Timestream, siga as etapas abaixo:

Tópicos

- Pré-requisitos
- Usar o console
- Usando o CLI
- Aplicação de exemplo
- Tutorial em vídeo

Pré-requisitos

1. Crie um banco de dados no Amazon Timestream usando as instruções descritas em. Criar um banco de dados do
2. Crie uma tabela no Amazon Timestream usando as instruções descritas em. Criar uma tabela

Usar o console

1. Use o console AWS de gerenciamento do AWS IoT Core para criar uma regra clicando em Gerenciar > Roteamento de mensagens > Regras seguido por Criar regra.
2. Defina o nome da regra como um nome de sua escolha e SQL o texto mostrado abaixo

```
SELECT temperature as temp, humidity from 'iot/topic'
```

3. Selecione Timestream na lista de ações
4. Especifique os nomes do banco de dados, da tabela e da dimensão do Timestream junto com a função para gravar dados no Timestream. Se a função não existir, você pode criar uma clicando em Criar funções
5. Para testar a regra, siga as instruções mostradas aqui.

Usando o CLI

Se você não instalou a interface de linha de AWS comando (AWS CLI), faça isso aqui.

1. Salve a carga útil da regra a seguir em um JSON arquivo chamado timestream_rule.json. Substituir *arn:aws:iam::123456789012:role/TimestreamRole* com sua função arn, que concede acesso à AWS IoT para armazenar dados no Amazon Timestream

```
{
    "actions": [
            {
                "timestream": {
                    "roleArn": "arn:aws:iam::123456789012:role/TimestreamRole",
                    "tableName": "devices_metrics",
                    "dimensions": [
                        {
                            "name": "device_id",
                            "value": "${clientId()}"
                        },
                        {
                            "name": "device_firmware_sku",
                            "value": "My Static Metadata"
                        }
                    ],
                    "databaseName": "record_devices"
                }
            }
    ],
    "sql": "select * from 'iot/topic'",
    "awsIotSqlVersion": "2016-03-23",
    "ruleDisabled": false
}
```

2. Crie uma regra de tópico usando o seguinte comando

```
aws iot create-topic-rule --rule-name timestream_test --topic-rule-payload file://
<path/to/timestream_rule.json> --region us-east-1
```

3. Recupere detalhes da regra do tópico usando o seguinte comando

```
aws iot get-topic-rule --rule-name timestream_test
```

4. Salve a seguinte carga de mensagem em um arquivo chamado 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. Teste a regra usando o seguinte comando

```
aws iot-data publish --topic 'iot/topic' --payload file://<path/to/
timestream_msg.json>
```

Aplicação de exemplo

Para ajudar você a começar a usar o Timestream com o AWS IoT Core, criamos um aplicativo de amostra totalmente funcional que cria os artefatos necessários no AWS IoT Core e no Timestream para criar uma regra de tópico e um aplicativo de amostra para publicar dados no tópico.

1. Clone o GitHub repositório do aplicativo de amostra para integração com o AWS IoT Core seguindo as instruções de GitHub
2. Siga as instruções em READMEPara usar um AWS CloudFormation modelo para criar os artefatos necessários no Amazon AWS Timestream e no IoT Core e para publicar mensagens de exemplo sobre o tópico.

Tutorial em vídeo

Este vídeo explica como o IoT Core funciona com o Timestream.

Amazon Managed Service for Apache Flink

Você pode usar o Apache Flink para transferir seus dados de séries temporais do Amazon Managed Service para Apache Flink, MSK Amazon, Apache Kafka e outras tecnologias de streaming

diretamente para o Amazon Timestream for. LiveAnalytics Criamos um conector de dados de amostra do Apache Flink para o Timestream. Também criamos um aplicativo de amostra para enviar dados para o Amazon Kinesis para que os dados possam fluir do Kinesis para o Managed Service para Apache Flink e, finalmente, para o Amazon Timestream. Todos esses artefatos estão disponíveis para você em GitHub. Este tutorial em vídeo descreve a configuração.

> **Note**
>
> O Java 11 é a versão recomendada para usar o serviço gerenciado para o aplicativo Apache Flink. Se você tiver várias versões do Java, certifique-se de exportar o Java 11 para sua variável de HOME ambiente JAVA _.

Tópicos

- Aplicação de exemplo
- Tutorial em vídeo

Aplicação de exemplo

Para começar, siga o procedimento abaixo:

1. Crie um banco de dados no Timestream com o nome `kdaflink` seguindo as instruções descritas em Criar um banco de dados do
2. Crie uma tabela no Timestream com o nome `kinesisdata1` seguindo as instruções descritas em Criar uma tabela
3. Crie um Amazon Kinesis Data Stream com o nome, `TimestreamTestStream` seguindo as instruções descritas em Criação de um stream
4. Clone o GitHub repositório do conector de dados Apache Flink para Timestream seguindo as instruções de GitHub
5. Para compilar, executar e usar o aplicativo de amostra, siga as instruções no conector de dados de amostra do Apache Flink README
6. Compile o serviço gerenciado para o aplicativo Apache Flink seguindo as instruções para compilar o código do aplicativo
7. Faça o upload do serviço gerenciado para o binário do aplicativo Apache Flink seguindo as instruções para carregar o código de streaming do Apache Flink

a. Depois de clicar em Criar aplicativo, clique no link da IAM função do aplicativo

b. Anexe as IAM políticas para AmazonKinesisReadOnlyAccessAmazonTimestreamFullAccesse.

> **Note**
>
> As IAM políticas acima não se restringem a recursos específicos e não são adequadas para uso em produção. Para um sistema de produção, considere o uso de políticas que restrinjam o acesso a recursos específicos.

8. Clone o GitHub repositório do aplicativo de amostra, gravando dados no Kinesis, seguindo as instruções do GitHub
9. Siga as instruções em READMEpara executar o aplicativo de amostra para gravar dados no Kinesis
10. Execute uma ou mais consultas no Timestream para garantir que os dados estejam sendo enviados do Kinesis para o Managed Service for Apache Flink to Timestream seguindo as instruções para Criar uma tabela

Tutorial em vídeo

Este vídeo explica como usar o Timestream com o Managed Service para Apache Flink.

Amazon Kinesis

Usando Amazon Managed Service for Apache Flink

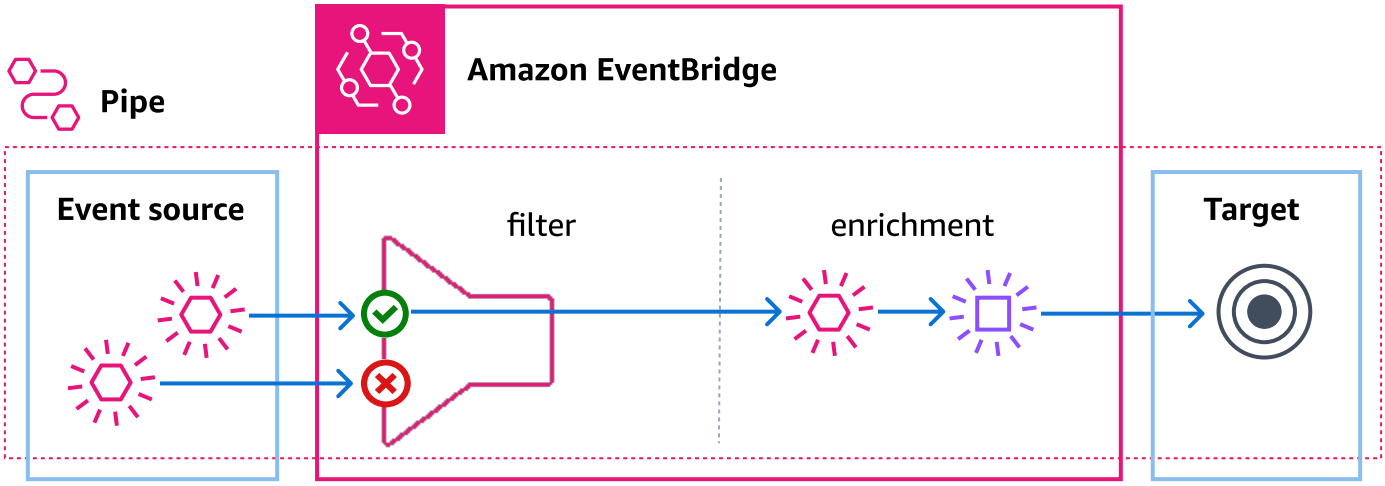
Você pode enviar dados do Kinesis Data Streams para o LiveAnalytics Timestream para usar o conector de dados de amostra Timestream do Managed Service for Apache Flink. Consulte Apache Flink Amazon Managed Service for Apache Flink para obter mais informações.

Usando EventBridge Pipes para enviar dados do Kinesis para Timestream

Você pode usar o EventBridge Pipes para enviar dados de um stream do Kinesis para um Amazon Timestream como tabela. LiveAnalytics

Os tubos são destinados a point-to-point integrações entre fontes e alvos suportados, com suporte para transformações e enriquecimento avançados. Os Pipes reduzem a necessidade

de conhecimento especializado e código de integração ao desenvolver arquiteturas orientadas a eventos. Para configurar um pipe, a origem é escolhida, adiciona filtragem opcional, define o enriquecimento opcional e escolhe o destino para os dados do evento.



Essa integração permite que você aproveite o poder dos recursos de análise de dados Timestream de séries temporais e, ao mesmo tempo, simplifique seu pipeline de ingestão de dados.

O uso EventBridge de tubos com Timestream oferece os seguintes benefícios:

- Ingestão de dados em tempo real: transmita dados do Kinesis diretamente para o Timestream LiveAnalytics for, permitindo análises e monitoramento em tempo real.
- Integração perfeita: utilize EventBridge Pipes para gerenciar o fluxo de dados sem a necessidade de integrações personalizadas complexas.
- Filtragem e transformação aprimoradas: filtre ou transforme os registros do Kinesis antes que eles sejam armazenados Timestream para atender aos seus requisitos específicos de processamento de dados.
- Escalabilidade: gerencie fluxos de dados de alto rendimento e garanta um processamento eficiente de dados com recursos integrados de paralelismo e agrupamento em lotes.

Configuração

Para configurar um EventBridge Pipe para o qual transmitir dados do Kinesis Timestream, siga estas etapas:

1. Criar uma transmissão do Kinesis

Certifique-se de ter um stream de dados ativo do Kinesis a partir do qual você deseja ingerir dados.

2. Crie um Timestream banco de dados e uma tabela

   Configure seu Timestream banco de dados e tabela onde os dados serão armazenados.

3. Configure o EventBridge tubo:
   - Fonte: selecione seu stream do Kinesis como fonte.
   - Alvo: Escolha Timestream como alvo.
   - Configurações de lote: defina a janela e o tamanho do lote para otimizar o processamento de dados e reduzir a latência.

> **Important**
>
> Ao configurar um tubo, recomendamos testar a exatidão de todas as configurações ingerindo alguns registros. Observe que a criação bem-sucedida de um canal não garante que o pipeline esteja correto e que os dados fluam sem erros. Pode haver erros de tempo de execução, como tabela incorreta, parâmetro de caminho dinâmico incorreto ou Timestream registro inválido após a aplicação do mapeamento, que serão descobertos quando os dados reais fluírem pelo canal.

As configurações a seguir determinam a taxa na qual os dados são ingeridos:

- BatchSize: o tamanho máximo do lote que será enviado ao Timestream. LiveAnalytics Intervalo: 0 - 100. A recomendação é manter esse valor como 100 para obter a taxa de transferência máxima.
- MaximumBatchingWindowInSeconds: O tempo máximo de espera para preencher o lote batchSize antes que o lote seja enviado ao Timestream para LiveAnalytics o destino. Dependendo da taxa de entrada de eventos, essa configuração decidirá o atraso da ingestão. A recomendação é manter esse valor < 10s para continuar enviando os dados quase Timestream em tempo real.
- ParallelizationFactor: o número de lotes a serem processados simultaneamente a partir de cada fragmento. A recomendação é usar o valor máximo de 10 para obter a taxa de transferência máxima e a ingestão quase em tempo real.

  Se seu stream for lido por vários alvos, use o fan-out aprimorado para fornecer um consumidor dedicado ao seu canal para obter um alto rendimento. Para obter mais informações, consulte

Desenvolvendo consumidores avançados com o Guia Kinesis Data Streams API do Kinesis Data Streams Usuário.

> **Note**
>
> A taxa de transferência máxima que pode ser alcançada é limitada pelas execuções simultâneas de canais por conta.

A configuração a seguir garante a prevenção da perda de dados:

- DeadLetterConfig: A recomendação é sempre configurar DeadLetterConfig para evitar qualquer perda de dados nos casos em que os eventos não puderam ser ingeridos no Timestream LiveAnalytics devido a erros do usuário.

Otimize o desempenho do seu canal com as seguintes configurações, que ajudam a evitar que os registros causem lentidão ou bloqueios.

- MaximumRecordAgeInSeconds: registros anteriores a esse não serão processados e serão movidos diretamente para DLQ o. Recomendamos definir esse valor para não ser maior do que o período de retenção do armazenamento de memória configurado da Timestream tabela de destino.
- MaximumRetryAttempts: o número de tentativas de repetição de um registro antes que o registro seja enviado para DeadLetterQueue. A recomendação é configurar isso em 10. Isso deve ser capaz de ajudar a resolver quaisquer problemas transitórios e, para problemas persistentes, o registro será movido DeadLetterQueue e desbloqueado o resto do stream.
- OnPartialBatchItemFailure: para fontes que oferecem suporte ao processamento parcial em lote, recomendamos que você habilite isso e configure como AUTOMATIC _ BISECT para tentar novamente os registros com falha antes de descartar/enviar para. DLQ

Exemplo de configuração

Aqui está um exemplo de como configurar um EventBridge Pipe para transmitir dados de um stream do Kinesis para uma Timestream tabela:

Example IAM atualizações de políticas para 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 Configuração do stream do Kinesis

```
{
  "Source": "arn:aws:kinesis:us-east-1:123456789012:stream/my-kinesis-stream",
  "SourceParameters": {
    "KinesisStreamParameters": {
        "BatchSize": 100,
        "DeadLetterConfig": {
            "Arn": "arn:aws:sqs:us-east-1:123456789012:my-sqs-queue"
        },
       "MaximumBatchingWindowInSeconds": 5,
        "MaximumRecordAgeInSeconds": 1800,
        "MaximumRetryAttempts": 10,
        "StartingPosition": "LATEST",
       "OnPartialBatchItemFailure": "AUTOMATIC_BISECT"
    }
  }
}
```

Example Timestream configuração de destino

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

Transformação de eventos

EventBridge Os tubos permitem que você transforme os dados antes que eles cheguem Timestream. Você pode definir regras de transformação para modificar os Kinesis registros recebidos, como alterar os nomes dos campos.

Suponha que seu Kinesis stream contenha dados de temperatura e umidade. Você pode usar uma EventBridge transformação para renomear esses campos antes de inseri-los em. Timestream

Práticas recomendadas

Armazenamento em lotes e armazenamento em buffer

- Configure a janela e o tamanho do lote para equilibrar a latência de gravação e a eficiência do processamento.
- Use uma janela de agrupamento em lotes para acumular dados suficientes antes do processamento, reduzindo a sobrecarga de pequenos lotes frequentes.

Processamento paralelo

Utilize a ParallelizationFactorconfiguração para aumentar a simultaneidade, especialmente para fluxos de alto rendimento. Isso garante que vários lotes de cada fragmento possam ser processados simultaneamente.

Transformação de dados

Aproveite os recursos de transformação do EventBridge Pipes para filtrar e aprimorar os registros antes de armazená-los Timestream. Isso pode ajudar a alinhar os dados com seus requisitos analíticos.

Segurança

- Certifique-se de que as IAM funções usadas para EventBridge Pipes tenham as permissões necessárias para leitura Kinesis e gravação Timestream.
- Use medidas de criptografia e controle de acesso para proteger dados em trânsito e em repouso.

Falhas de depuração

- Desativação automática de tubulações

  Os tubos serão desativados automaticamente em cerca de 2 horas se o alvo não existir ou tiver problemas de permissão
- Controles de utilização

  Os tubos têm a capacidade de recuar automaticamente e tentar novamente até que os aceleradores sejam reduzidos.
- Habilitando registros

  Recomendamos que você ative os registros em um ERROR nível e inclua dados de execução para obter mais informações sobre falhas. Em caso de falha, esses registros conterão request/response sent/received de Timestream. Isso ajuda você a entender o erro associado e, se necessário, a reprocessar os registros após corrigi-lo.

Monitorar

Recomendamos que você configure os alarmes a seguir para detectar quaisquer problemas com o fluxo de dados:

- Idade máxima do registro na fonte
  - `GetRecords.IteratorAgeMilliseconds`
- Métricas de falha em tubulações
  - `ExecutionFailed`
  - `TargetStageFailed`
- Timestream APIErros de gravação
  - `UserErrors`

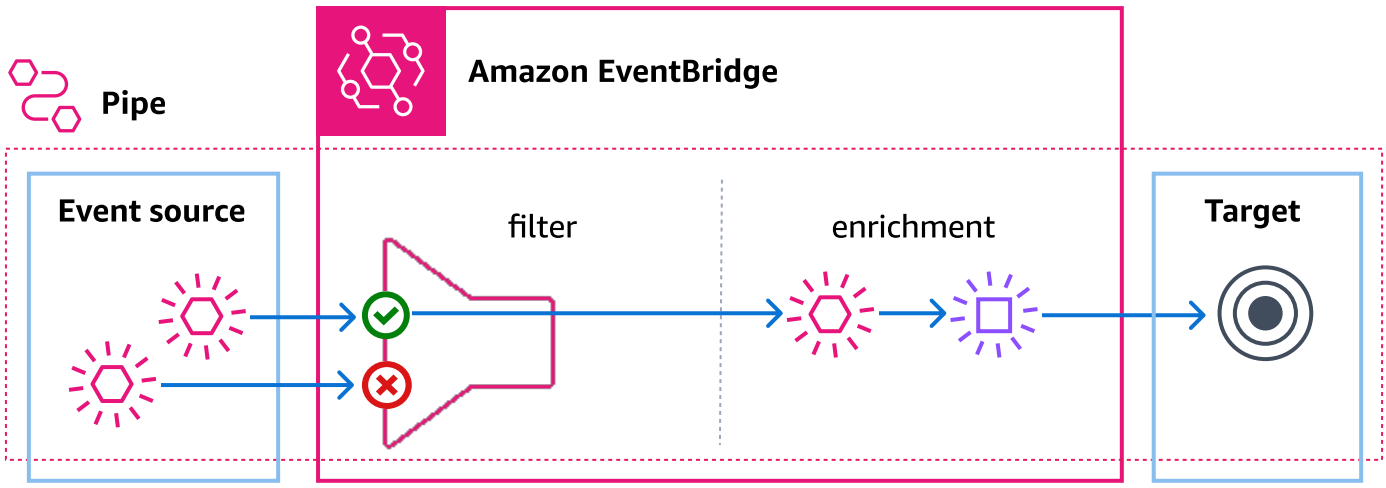
Para obter métricas adicionais de monitoramento, consulte Monitoramento EventBridge no Guia EventBridge do usuário.

Amazon MQ

Usando EventBridge Pipes para enviar dados do Amazon MQ para Timestream

Você pode usar o EventBridge Pipes para enviar dados de um agente do Amazon MQ para um Amazon Timestream como tabela. LiveAnalytics

Os tubos são destinados a point-to-point integrações entre fontes e alvos suportados, com suporte para transformações e enriquecimento avançados. Os Pipes reduzem a necessidade de conhecimento especializado e código de integração ao desenvolver arquiteturas orientadas a eventos. Para configurar um pipe, a origem é escolhida, adiciona filtragem opcional, define o enriquecimento opcional e escolhe o destino para os dados do evento.



Para obter mais informações sobre EventBridge tubos, consulte EventBridge Tubos no Guia EventBridge do usuário. Para obter informações sobre como configurar um canal para entregar eventos a um Amazon LiveAnalytics Timestream para tabela EventBridge , consulte Especificações de destino do Pipes.

Amazon MSK

Usando o Managed Service para Apache Flink para enviar Amazon MSK dados ao Timestream para LiveAnalytics

Você pode enviar dados de Amazon MSK para Timestream criando um conector de dados semelhante ao conector de Timestream dados de amostra do Managed Service for Apache Flink. Consulte Amazon Managed Service for Apache Flink para obter mais informações.

Usando o Kafka Connect para enviar MSK dados da Amazon ao Timestream para LiveAnalytics

Você pode usar o Kafka Connect para ingerir seus dados de séries temporais Amazon MSK diretamente no Timestream for. LiveAnalytics

Criamos um exemplo de conector de pia Kafka para. Timestream Também criamos um exemplo de plano de jMeter teste do Apache para publicar dados em um tópico do Kafka, para que os dados possam fluir do tópico por meio do conector Timestream Kafka Sink até um Timestream para a tabela. LiveAnalytics Todos esses artefatos estão disponíveis em. GitHub

> **Note**
>
> O Java 11 é a versão recomendada para usar o Timestream Kafka Sink Connector. Se você tiver várias versões do Java, certifique-se de exportar o Java 11 para sua variável de HOME ambiente JAVA _.

Criação de um aplicativo de amostra

Para começar, siga o procedimento abaixo.

1. No Timestream for LiveAnalytics, crie um banco de dados com o nome. `kafkastream`

   Consulte o procedimento ??? para obter instruções detalhadas.
2. Em Timestream for LiveAnalytics, crie uma tabela com o nome. `purchase_history`

   Consulte o procedimento ??? para obter instruções detalhadas.
3. Siga as instruções compartilhadas no para criar o seguinte:, e.

- Um Amazon MSK cluster
- Uma Amazon EC2 instância configurada como uma máquina cliente produtora do Kafka
- Um tópico sobre Kafka

Consulte os pré-requisitos do projeto kafka_ingestor para obter instruções detalhadas.

4. Clone o repositório do Timestream Kafka Sink Connector.

   Consulte Clonar um repositório em GitHub para obter instruções detalhadas.

5. Compile o código do plugin.

   Consulte Conector - Construir do código-fonte GitHub para obter instruções detalhadas.

6. Faça upload dos seguintes arquivos em um bucket do S3: seguindo as instruções descritas em.

   - O arquivo jar (kafka-connector-timestream-> VERSION <- jar-with-dependencies .jar) do diretório `/target`
   - O arquivo de amostra do esquema json,. `purchase_history.json`

   Consulte Carregar objetos no Guia do Amazon S3 usuário para obter instruções detalhadas.

7. Crie dois VPC endpoints. Esses endpoints seriam usados pelo MSK conector para acessar os recursos usados AWS PrivateLink.

   - Um para acessar o Amazon S3 bucket
   - Um para acessar o Timestream da mesa. LiveAnalytics

   Consulte VPCEndpoints para obter instruções detalhadas.

8. Crie um plug-in personalizado com o arquivo jar enviado.

   Consulte Plugins no Guia do Amazon MSK desenvolvedor para obter instruções detalhadas.

9. Crie uma configuração de trabalhador personalizada com o JSON conteúdo descrito nos parâmetros de configuração do trabalhador, seguindo as instruções descritas em

   Consulte Criação de uma configuração de trabalhador personalizada no Guia do Amazon MSK desenvolvedor para obter instruções detalhadas.

10. Crie uma IAM função de execução de serviço.

Consulte IAM Service Role para obter instruções detalhadas.

11. Crie um Amazon MSK conector com o plug-in personalizado, a configuração personalizada do trabalhador e a IAM função de execução do serviço criada nas etapas anteriores e com o Sample Connector Configuration.

    Consulte Criação de um conector no Guia do Amazon MSK desenvolvedor para obter instruções detalhadas.

    Certifique-se de atualizar os valores dos parâmetros de configuração abaixo com os respectivos valores. Consulte os parâmetros de configuração do conector para obter detalhes.

    - `aws.region`
    - `timestream.schema.s3.bucket.name`
    - `timestream.ingestion.endpoint`

    A criação do conector leva de 5 a 10 minutos para ser concluída. O pipeline está pronto quando seu status muda para`Running`.

12. Publique um fluxo contínuo de mensagens para gravar dados no tópico criado pelo Kafka.

    Consulte Como usá-lo para obter instruções detalhadas.

13. Execute uma ou mais consultas para garantir que os dados estejam sendo enviados da tabela MSK Connect Amazon MSK to the Timestream for. LiveAnalytics

    Consulte o procedimento ??? para obter instruções detalhadas.

Recursos adicionais

O blog Ingestão de dados sem servidor em tempo real de seus clusters do Kafka no Timestream para LiveAnalytics usar o Kafka Connect explica a configuração de end-to-end um pipeline usando o Timestream for Kafka Sink Connector, começando com uma máquina cliente produtora do Kafka que usa o plano de teste do jMeter Apache LiveAnalytics para publicar milhares de mensagens de amostra em um tópico do Kafka até a verificação dos registros ingeridos em um Timestream for table. LiveAnalytics

Amazon QuickSight

Você pode usar QuickSight a Amazon para analisar e publicar painéis de dados que contêm seus dados do Amazon Timestream. Esta seção descreve como você pode criar uma nova conexão de fonte de QuickSight dados, modificar permissões, criar novos conjuntos de dados e realizar uma análise. Este tutorial em vídeo descreve como trabalhar com o Timestream e a Amazon. QuickSight

> **Note**
>
> Todos os conjuntos de dados na Amazon QuickSight são somente para leitura. Você não pode fazer nenhuma alteração em seus dados reais no Timestream usando a Amazon QuickSight para remover a fonte de dados, o conjunto de dados ou os campos.

Tópicos

- Acessando o Amazon Timestream a partir de QuickSight
- Crie uma nova conexão de fonte QuickSight de dados para o Timestream
- Permissões de edição para a conexão da fonte de QuickSight dados para Timestream
- Crie um novo QuickSight conjunto de dados para o Timestream
- Crie uma nova análise para o Timestream
- Tutorial em vídeo

Acessando o Amazon Timestream a partir de QuickSight

Antes de continuar, a Amazon QuickSight precisa estar autorizada a se conectar ao Amazon Timestream. Se as conexões não estiverem habilitadas, você receberá uma mensagem de erro ao tentar se conectar. Um QuickSight administrador pode autorizar conexões com AWS recursos. Para autorizar uma conexão com QuickSight o Timestream, siga o procedimento em Usando outros AWS serviços: reduzindo o acesso, escolhendo Amazon Timestream na etapa 5.

Crie uma nova conexão de fonte QuickSight de dados para o Timestream

> **Note**
>
> A conexão entre a Amazon QuickSight e o Amazon Timestream é criptografada em trânsito SSL usando TLS (1.2). Você não pode criar uma conexão não criptografada.

1. Certifique-se de ter configurado as permissões apropriadas para QuickSight que a Amazon acesse o Amazon Timestream, conforme descrito em. Acessando o Amazon Timestream a partir de QuickSight
2. Comece criando um conjunto de dados. Escolha Conjuntos de dados no painel de navegação e, em seguida, escolha Novo conjunto de dados.
3. Selecione o cartão de origem de dados Timestream.
4. Em Nome da fonte de dados, insira um nome para sua conexão de fonte de dados Timestream, por exemplo. `US Timestream Data`

   > **Note**
   >
   > Como você pode criar vários conjuntos de dados usando uma conexão com o Timestream, é melhor manter o nome simples.

5. Escolha Validar conexão para verificar se você consegue se conectar com êxito ao Timestream.

   > **Note**
   >
   > Validar conexão só valida que você pode se conectar. No entanto, ele não valida uma tabela ou consulta específica.

6. Escolha Criar fonte de dados para continuar.
7. Em Banco de dados, escolha Selecionar... para ver a lista de opções disponíveis. Escolha o que você deseja usar.
8. Escolha Selecionar para continuar.
9. Escolha uma das seguintes opções:

   - Para importar seus dados para QuickSight o mecanismo na memória (chamadoSPICE), escolha Importar SPICE para para uma análise mais rápida.
   - QuickSight Para permitir a execução de uma consulta em seus dados sempre que você atualizar o conjunto de dados ou usar a análise ou o painel, escolha Consultar diretamente seus dados.

10. Escolha Editar ou visualizar e selecione Salvar para salvar seu conjunto de dados e fechá-lo.

Permissões de edição para a conexão da fonte de QuickSight dados para Timestream

O procedimento a seguir descreve como visualizar, adicionar e revogar permissões para outros QuickSight usuários para que eles possam acessar a mesma fonte de dados do Timestream. As pessoas precisam ser usuários ativos QuickSight antes que você possa adicioná-las.

> **Note**
>
> Em QuickSight, as fontes de dados têm dois níveis de permissão: usuário e proprietário.
>
> - Escolha o usuário para permitir o acesso de leitura.
> - Escolha o proprietário para permitir que o usuário edite, compartilhe ou exclua essa fonte QuickSight de dados.

1. Certifique-se de ter configurado as permissões apropriadas para QuickSight que a Amazon acesse o Amazon Timestream, conforme descrito em. Acessando o Amazon Timestream a partir de QuickSight
2. Escolha Conjuntos de dados à esquerda e role para baixo até encontrar o cartão da fonte de dados para a conexão do Timestream. Por exemplo, `US Timestream Data`.
3. Escolha a placa `Timestream` de origem de dados.
4. Selecione `Share data source`. Uma lista das permissões atuais é exibida.
5. (Opcional) Para editar as permissões, você pode escolher `user` ou`owner`.
6. (Opcional) Para revogar as permissões, escolha`Revoke access`. As pessoas que você revoga não podem criar novos conjuntos de dados a partir dessa fonte de dados. No entanto, seus conjuntos de dados existentes ainda terão acesso a essa fonte de dados.
7. Para adicionar permissões`Invite users`, escolha e siga estas etapas para adicionar um usuário:
   a. Adicione pessoas para permitir que elas usem a mesma fonte de dados.
   b. Para cada um, escolha o `Permission` que você deseja aplicar.
8. Quando terminar, selecione `Close`.

Crie um novo QuickSight conjunto de dados para o Timestream

1. Certifique-se de ter configurado as permissões apropriadas para QuickSight que a Amazon acesse o Amazon Timestream, conforme descrito em. Acessando o Amazon Timestream a partir de QuickSight
2. Escolha Conjuntos de dados à esquerda e role para baixo até encontrar o cartão da fonte de dados para a conexão do Timestream. Se você tiver muitas fontes de dados, poderá usar a barra de pesquisa na parte superior da página para encontrá-la com uma correspondência parcial no nome.
3. Escolha a placa de origem de dados Timestream. Em seguida, escolha Criar conjunto de dados.
4. Em Banco de dados, escolha Selecionar para visualizar a lista de opções disponíveis. Escolha o banco de dados que você deseja usar.
5. Em Tabelas, escolha a tabela que deseja usar.
6. Escolha Editar ou visualizar.
7. (Opcional) Para adicionar mais dados, escolha Adicionar dados no canto superior direito.
    a. Escolha Alternar fonte de dados e escolha uma fonte de dados diferente.
    b. Siga as instruções da interface do usuário para concluir a adição de dados.
    c. Após adicionar novos dados ao mesmo conjunto de dados, escolha Configurar esta junção (os dois pontos vermelhos). Configure uma junção para cada tabela adicional.
    d. Se quiser adicionar campos calculados, escolha Adicionar campos calculados.
    e. Para usar o Sagemaker, escolha Aumentar com. SageMaker Essa opção só está disponível na edição QuickSight Enterprise.
    f. Desmarque todos os campos que você deseja omitir.
    g. Atualize todos os tipos de dados que você deseja alterar.
8. Quando concluir, escolha Salvar para salvar e fechar o conjunto de dados.

Crie uma nova análise para o Timestream

1. Certifique-se de ter configurado as permissões apropriadas para QuickSight que a Amazon acesse o Amazon Timestream, conforme descrito em. Acessando o Amazon Timestream a partir de QuickSight
2. Escolha Análises à esquerda.
3. Escolha uma das seguintes opções:

- Para criar uma análise, escolha Nova análise à direita.
- Para adicionar o conjunto de dados Timestream a uma análise existente, abra a análise que você deseja editar. Escolha o ícone de lápis próximo ao canto superior esquerdo e, em seguida, Adicionar conjunto de dados.

4. Comece a primeira visualização de dados escolhendo os campos à esquerda.
5. Para obter mais informações, consulte Trabalhando com análises - Amazon QuickSight

Tutorial em vídeo

Este vídeo explica como a Amazon QuickSight trabalha com o Timestream.

Amazon SageMaker

Você pode usar o Amazon SageMaker Notebooks para integrar seus modelos de aprendizado de máquina com o Amazon Timestream. Para ajudar você a começar, criamos um SageMaker Notebook de amostra que processa dados do Timestream. Os dados são inseridos no Timestream a partir de um aplicativo Python multiencadeado que envia dados continuamente. O código-fonte do SageMaker Notebook de amostra e do aplicativo Python de amostra está disponível em. GitHub

1. Crie um banco de dados e uma tabela seguindo as instruções descritas em Criar um banco de dados do e Criar uma tabela
2. Clone o GitHub repositório do aplicativo de amostra Python com vários segmentos seguindo as instruções de GitHub
3. Clone o GitHub repositório para o exemplo do Timestream SageMaker Notebook seguindo as instruções de. GitHub
4. Execute o aplicativo para ingerir dados continuamente no Timestream seguindo as instruções no README
5. Siga as instruções para criar um bucket do Amazon S3 para a Amazon, SageMaker conforme descrito aqui.
6. Crie uma SageMaker instância da Amazon com o boto3 mais recente instalado: Além das instruções descritas aqui, siga as etapas abaixo:

    a. Na página Criar instância do notebook, clique em Configuração adicional
    b. Clique em Configuração do ciclo de vida - opcional e selecione Criar uma nova configuração do ciclo de vida

c. Na caixa do assistente Criar configuração do ciclo de vida, faça o seguinte:

i. Preencha o nome desejado para a configuração, por exemplo `on-start`

ii. Em Iniciar script do Notebook, copie e cole o conteúdo do script do Github

iii. `PACKAGE=scipy`Substitua por `PACKAGE=boto3` no script colado.

7. Clique em Criar configuração
8. Acesse o IAM serviço no AWS Management Console e encontre a função de SageMaker execução recém-criada para a instância do notebook.
9. Anexe a IAM política para `AmazonTimestreamFullAccess` à função de execução.

> **Note**
>
> A `AmazonTimestreamFullAccess` IAM política não se restringe a recursos específicos e não é adequada para uso em produção. Para um sistema de produção, considere o uso de políticas que restrinjam o acesso a recursos específicos.

10. Quando o status da instância do notebook for InService, escolha Abrir Jupyter para iniciar um SageMaker Notebook para a instância
11. Faça upload dos arquivos `timestreamquery.py` e `Timestream_SageMaker_Demo.ipynb` para o Notebook selecionando o botão Carregar
12. Escolha `Timestream_SageMaker_Demo.ipynb`

> **Note**
>
> Se você ver um pop-up com Kernel não encontrado, escolha conda_python3 e clique em Definir kernel.

13. Modifique `DB_NAME`,`TABLE_NAME`,`bucket`, e `ENDPOINT` para corresponder ao nome do banco de dados, nome da tabela, nome do bucket do S3 e região dos modelos de treinamento.
14. Escolha o ícone de reprodução para executar as células individuais
15. Ao chegar à célula`Leverage Timestream to find hosts with average CPU utilization across the fleet`, certifique-se de que a saída retorne pelo menos 2 nomes de host.

> **Note**
>
> Se houver menos de 2 nomes de host na saída, talvez seja necessário executar novamente o aplicativo Python de amostra que ingere dados no Timestream com um número maior de threads e escala de host.

16. Ao chegar à célula`Train a Random Cut Forest (RCF) model using the CPU utilization history`, altere o `train_instance_type` com base nos requisitos de recursos para seu trabalho de treinamento
17. Ao chegar à célula`Deploy the model for inference`, altere a `instance_type` com base nos requisitos de recursos para seu trabalho de inferência

> **Note**
>
> Pode levar alguns minutos para treinar o modelo. Quando o treinamento for concluído, você verá a mensagem Concluído - Trabalho de treinamento concluído na saída da célula.

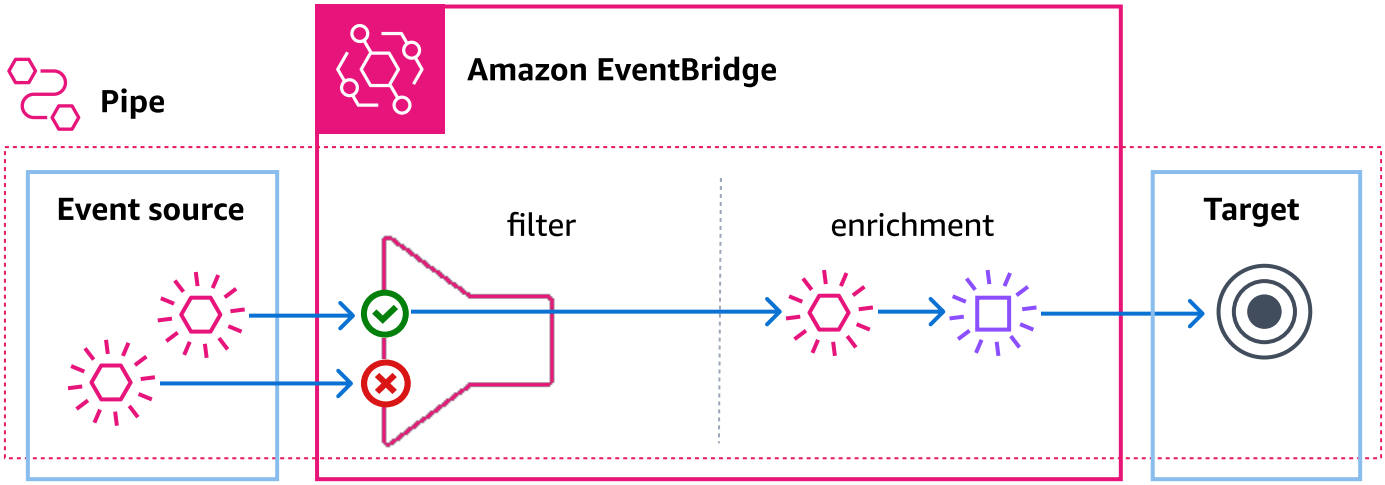
18. Execute a célula `Stop and delete the endpoint` para limpar os recursos. Você também pode parar e excluir a instância do SageMaker console.

Amazon SQS

Usando EventBridge Pipes para enviar SQS dados da Amazon para Timestream

Você pode usar o EventBridge Pipes para enviar dados de uma SQS fila da Amazon para um Amazon LiveAnalytics Timestream como tabela.

Os tubos são destinados a point-to-point integrações entre fontes e alvos suportados, com suporte para transformações e enriquecimento avançados. Os Pipes reduzem a necessidade de conhecimento especializado e código de integração ao desenvolver arquiteturas orientadas a eventos. Para configurar um pipe, a origem é escolhida, adiciona filtragem opcional, define o enriquecimento opcional e escolhe o destino para os dados do evento.



Para obter mais informações sobre EventBridge tubos, consulte EventBridge Tubos no Guia EventBridge do usuário. Para obter informações sobre como configurar um canal para entregar eventos a um Amazon LiveAnalytics Timestream para tabela EventBridge , consulte Especificações de destino do Pipes.

Usando DBeaver para trabalhar com o Amazon Timestream

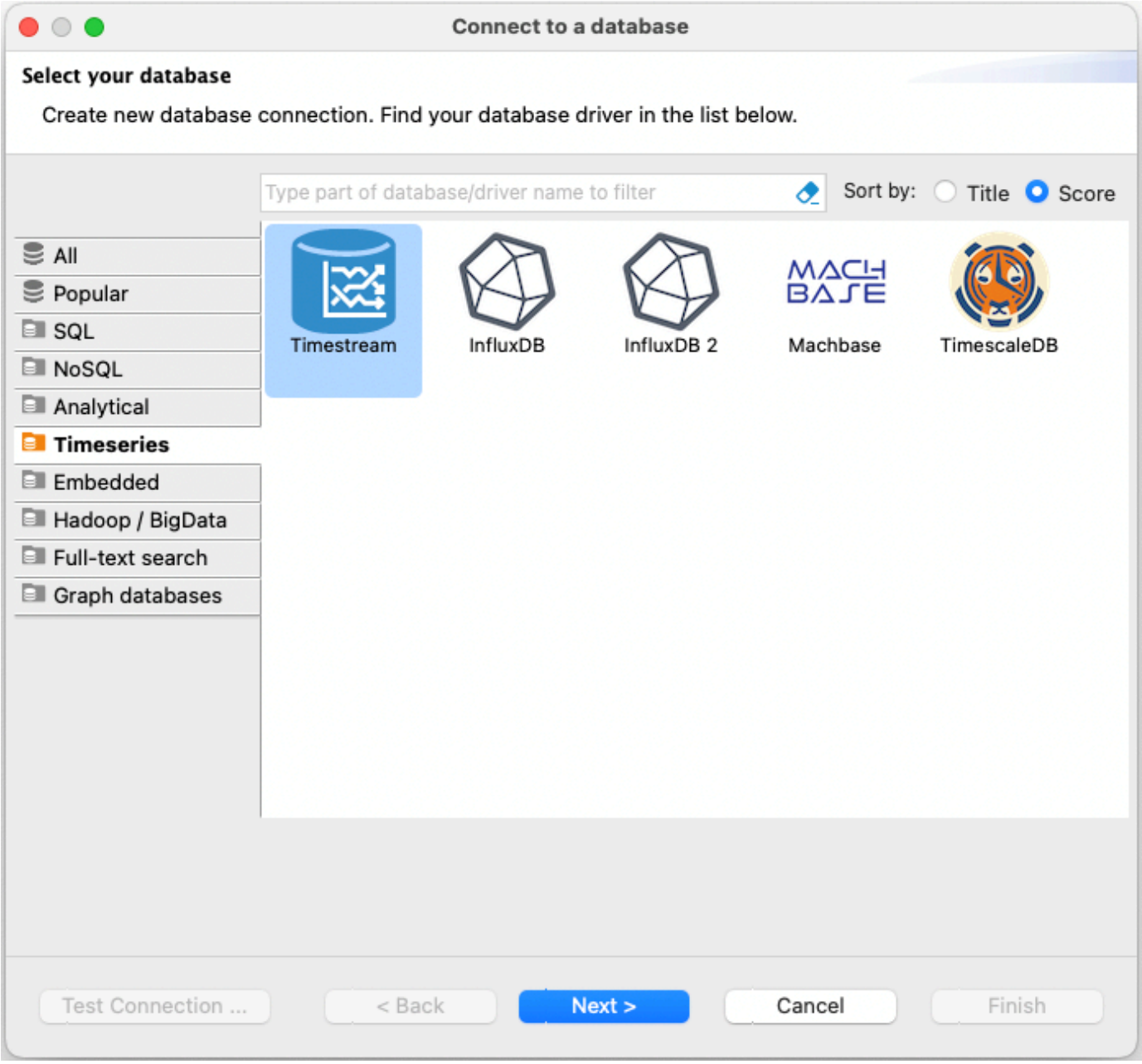
DBeaveré um SQL cliente universal gratuito que pode ser usado para gerenciar qualquer banco de dados que tenha um JDBC driver. É amplamente usado entre desenvolvedores e administradores de banco de dados por causa de seus recursos robustos de visualização, edição e gerenciamento de dados.

Usando as opções DBeaver de conectividade na nuvem, você pode se conectar DBeaver ao Amazon Timestream de forma nativa. DBeaverfornece uma interface abrangente e intuitiva para trabalhar com dados de séries temporais diretamente de dentro de um DBeaver aplicativo. Usando suas credenciais, ele também oferece acesso total a qualquer consulta que você possa executar em outra interface de consulta. Ele ainda permite criar gráficos para melhor compreensão e visualização dos resultados da consulta.

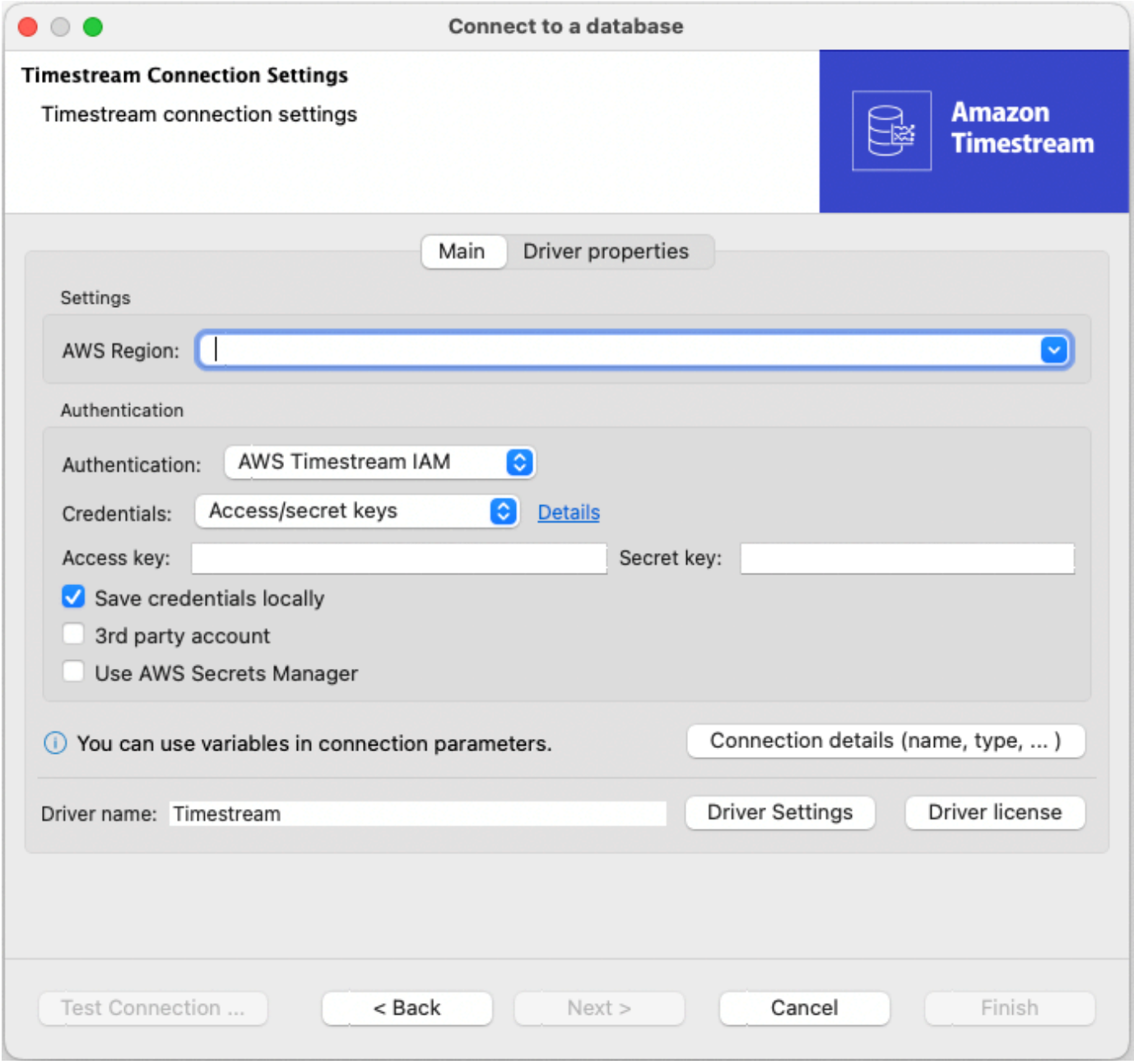
Configurando DBeaver para trabalhar com o Timestream

Siga as etapas a seguir para configurar DBeaver para trabalhar com o Timestream:

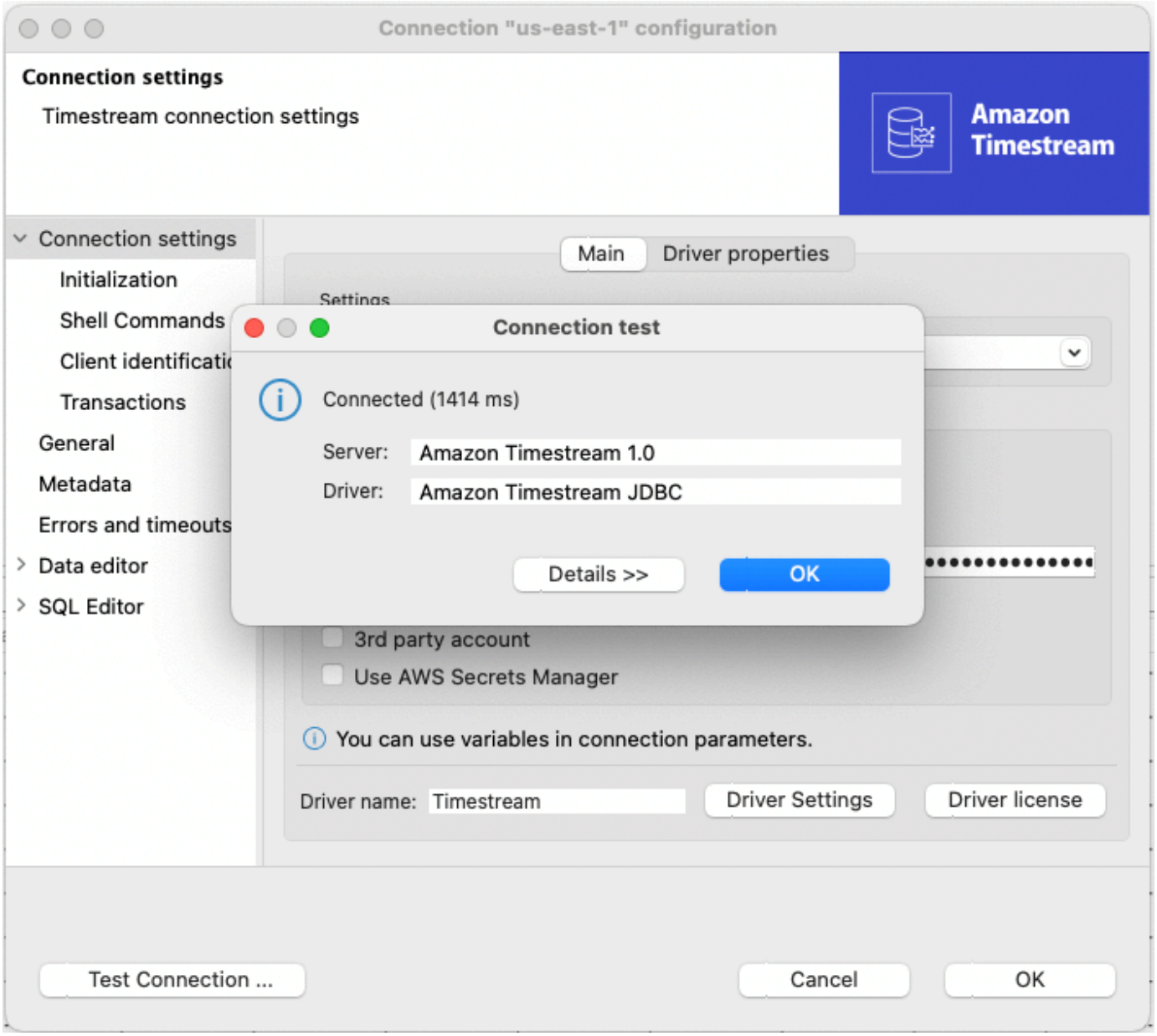
1. Baixe e instale DBeaver em sua máquina local.
2. DBeaverInicie, navegue até a área de seleção do banco de dados, escolha Timeseries no painel esquerdo e selecione o ícone Timestream no painel direito:



3. Na janela Configurações de conexão do Timestream, insira todas as informações necessárias para se conectar ao seu banco de dados do Amazon Timestream. Certifique-se de que as chaves de usuário inseridas tenham as permissões necessárias para acessar seu banco de dados do Timestream. Além disso, certifique-se de manter as informações e as chaves inseridas em DBeaver segurança e privacidade, como acontece com qualquer informação confidencial.



4. Teste a conexão para garantir que tudo esteja configurado corretamente:



5. Se o teste de conexão for bem-sucedido, agora você poderá interagir com seu banco de dados Amazon Timestream da mesma forma que faria com qualquer outro banco de dados em. DBeaver Por exemplo, você pode navegar até o SQL editor ou até a visualização do Diagrama ER para executar consultas:

6. DBeavertambém fornece ferramentas poderosas de visualização de dados. Para usá-los, execute sua consulta e selecione o ícone do gráfico para visualizar o conjunto de resultados. A ferramenta gráfica pode ajudar você a entender melhor as tendências dos dados ao longo do tempo.

Combinar o Amazon Timestream com o Amazon DBeaver Timestream cria um ambiente eficaz para gerenciar dados de séries temporais. Você pode integrá-lo perfeitamente ao seu fluxo de trabalho existente para aumentar a produtividade e a eficiência.

Grafana

Você pode visualizar seus dados de séries temporais e criar alertas usando o Grafana. Para ajudar você a começar com a visualização de dados, criamos um painel de amostra no Grafana que visualiza os dados enviados para o Timestream a partir de um aplicativo Python e um tutorial em vídeo que descreve a configuração.

Tópicos

- Aplicação de exemplo
- Tutorial em vídeo

Aplicação de exemplo

1. Crie um banco de dados e uma tabela no Timestream seguindo as instruções descritas em Criar um banco de dados do para obter mais informações.

   > **Note**
   >
   > O nome padrão do banco de dados e o nome da tabela para o painel do Grafana são definidos como GrafanaDB e respectivamente. grafanaTable Use esses nomes para minimizar a configuração.

2. Instale o Python 3.7 ou superior
3. Instale e configure o Timestream Python SDK
4. Clone o GitHub repositório do aplicativo Python de vários threads, ingerindo dados continuamente no Timestream seguindo as instruções de GitHub
5. Execute o aplicativo para ingerir dados continuamente no Timestream seguindo as instruções no README
6. Conclua a introdução ao Amazon Managed Grafana ou conclua a instalação do Grafana.
7. Se estiver instalando o Grafana em vez de usar o Amazon Managed Grafana, conclua a instalação do plug-in Timestream para o Grafana.

8. Abra o painel do Grafana usando um navegador de sua escolha. Se você instalou localmente o Grafana, você pode seguir as instruções descritas na documentação do Grafana para fazer login
9. Depois de iniciar o Grafana, vá para Datasources, clique em Adicionar Datasource, pesquise Timestream e selecione a fonte de dados Timestream
10. Configure o provedor de autenticação e a região e clique em Salvar e testar
11. Definir as macros padrão
    a. Defina $__database como o nome do seu banco de dados Timestream (por exemplo, GrafanaDB)
    b. Defina $__table como o nome da sua tabela Timestream (por exemplo) grafanaTable
    c. Defina $__measure como a medida mais usada na tabela
12. Clique em Salvar e testar
13. Clique na guia Painéis
14. Clique em Importar para importar o painel
15. Clique duas vezes no painel de aplicativos de amostra
16. Clique nas configurações do painel
17. Selecionar variáveis
18. Alterar dbName e tableName combinar os nomes do banco de dados e da tabela Timestream
19. Clique em Salvar
20. Atualize o painel
21. Para criar alertas, siga as instruções descritas na documentação do Grafana para Criar uma regra de alerta gerenciada do Grafana
22. Para solucionar problemas de alertas, siga as instruções descritas na documentação da Grafana para Solução de Problemas
23. Para obter informações adicionais, consulte a documentação da Grafana

Tutorial em vídeo

Este vídeo explica como o Grafana funciona com o Timestream.

Usando SquaredUp para trabalhar com o Amazon Timestream

SquaredUpé uma plataforma de observabilidade que se integra ao Amazon Timestream. Você pode usar o designer SquaredUp de painel intuitivo para visualizar, analisar e monitorar seus dados de

séries temporais. Os painéis podem ser compartilhados de forma pública ou privada, e canais de notificação podem ser criados para alertá-lo quando o estado de saúde de um monitor mudar.

Usando SquaredUp com o Amazon Timestream

1. Inscreva-se SquaredUpe comece gratuitamente.
2. Adicione uma fonte AWS de dados.
3. Crie um quadro de painel que use o fluxo de dados do Timestream Query.
4. Opcionalmente, ative o monitoramento do bloco, crie um canal de notificação ou compartilhe o painel de forma pública ou privada.
5. Opcionalmente, crie outros blocos para ver seus dados do Timestream junto com os dados de suas outras ferramentas de monitoramento e observabilidade.

Telegraf de código aberto

Você pode usar o plug-in Timestream for LiveAnalytics output do Telegraf para gravar métricas no Timestream diretamente do Telegraf de LiveAnalytics código aberto.

Esta seção fornece uma explicação de como instalar o Telegraf com o Timestream para plug-in de LiveAnalytics saída, como executar o Telegraf com o Timestream para plug-in de LiveAnalytics saída e como o Telegraf de código aberto funciona com o Timestream for. LiveAnalytics

Tópicos

- Instalando o Telegraf com o Timestream para plug-in de saída LiveAnalytics
- Executando o Telegraf com o Timestream para plug-in de saída LiveAnalytics
- Mapeando métricas do Telegraf/InfluxDB para o Timestream for model LiveAnalytics

Instalando o Telegraf com o Timestream para plug-in de saída LiveAnalytics

A partir da versão 1.16, o plug-in Timestream for LiveAnalytics output está disponível na versão oficial do Telegraf. Para instalar o plug-in de saída na maioria dos principais sistemas operacionais, siga as etapas descritas na documentação do InfluxData Telegraf. Para instalar no sistema operacional Amazon Linux 2, siga as instruções abaixo.

Instalando o Telegraf com o Timestream para LiveAnalytics plug-in de saída no Amazon Linux 2

Para instalar o Telegraf com o Timestream Output Plugin no Amazon Linux 2, execute as etapas a seguir.

1. Instale o Telegraf usando o gerenciador de `yum` pacotes.

```
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. Execute o seguinte comando .

```
sudo sed -i "s/\$releasever/$(rpm -E %{rhel})/g" /etc/yum.repos.d/influxdb.repo
```

3. Instale e inicie o Telegraf.

```
sudo yum install telegraf
sudo service telegraf start
```

Executando o Telegraf com o Timestream para plug-in de saída LiveAnalytics

Você pode seguir as instruções abaixo para executar o Telegraf com o plug-in Timestream for. LiveAnalytics

1. Gere um exemplo de configuração usando o Telegraf.

```
telegraf --section-filter agent:inputs:outputs --input-filter cpu:mem --output-
filter timestream config > example.config
```

2. Crie um banco de dados no Timestream usando o console de gerenciamento CLI, ou. SDKs
3. No `example.config` arquivo, adicione o nome do seu banco de dados editando a chave a seguir na `[[outputs.timestream]]` seção.

```
database_name = "yourDatabaseNameHere"
```

4. Por padrão, o Telegraf criará uma tabela. Se você quiser criar uma tabela manualmente, `create_table_if_not_exists` defina `false` e siga as instruções para criar uma tabela usando o console de gerenciamento CLI, ou SDKs.
5. No arquivo example.config, configure as credenciais na seção. `[[outputs.timestream]]` As credenciais devem permitir as seguintes operações.

```
timestream:DescribeEndpoints
timestream:WriteRecords
```

> **Note**
>
> Se você deixar `create_table_if_not_exists` definido como`true`, inclua:
>
> ```
> timestream:CreateTable
> ```

> **Note**
>
> Se você `describe_database_on_start` definir como`true`, inclua o seguinte.
>
> ```
> timestream:DescribeDatabase
> ```

6. Você pode editar o resto da configuração de acordo com suas preferências.
7. Quando terminar de editar o arquivo de configuração, execute o Telegraf com o seguinte.

```
./telegraf --config example.config
```

8. As métricas devem aparecer em alguns segundos, dependendo da configuração do seu agente. Você também deve ver as novas tabelas, cpu e mem, no console Timestream.

Mapeando métricas do Telegraf/InfluxDB para o Timestream for model LiveAnalytics

Ao gravar dados do Telegraf no Timestream for LiveAnalytics, os dados são mapeados da seguinte forma.

- O carimbo de data/hora é escrito como o campo de hora.

- As etiquetas são escritas como dimensões.
- Os campos são escritos como medidas.
- As medidas são escritas principalmente como nomes de tabelas (mais sobre isso abaixo).

O plug-in Timestream for LiveAnalytics output para Telegraf oferece várias opções para organizar e armazenar dados no Timestream for. LiveAnalytics Isso pode ser descrito com um exemplo que começa com os dados no formato de protocolo de linha.

```
weather,location=us-midwest,season=summer temperature=82,humidity=71
1465839830100400200 airquality,location=us-west no2=5,pm25=16
1465839830100400200
```

A seguir, descrevemos os dados.

- Os nomes das medidas são `weather` `airquality` e.
- As etiquetas são `location` `season` e.
- Os campos são `temperature` `humidity`no2,, `pm25` e.

Tópicos

- Armazenando os dados em várias tabelas
- Armazenando os dados em uma única tabela

Armazenando os dados em várias tabelas

Você pode escolher criar uma tabela separada por medida e armazenar cada campo em uma linha separada por tabela.

A configuração é`mapping_mode = "multi-table"`.

- O Timestream para o LiveAnalytics adaptador criará duas tabelas, a saber, e. `weather` `airquality`
- Cada linha da tabela conterá somente um único campo.

O Timestream resultante para LiveAnalytics tabelas `weather` e`airquality`, terá a seguinte aparência.

`weather`

horário	local	temporada	nome_medida	valor_medida::bigint
2016-06-13 17:43:50	centro-oeste dos EUA	verão	temperatura	82
2016-06-13 17:43:50	centro-oeste dos EUA	verão	umidade	71

`airquality`

horário	local	nome_medida	valor_medida::bigint
2016-06-13 17:43:50	centro-oeste dos EUA	não 2	5
2016-06-13 17:43:50	centro-oeste dos EUA	pm25	16

Armazenando os dados em uma única tabela

Você pode optar por armazenar todas as medidas em uma única tabela e armazenar cada campo em uma linha separada da tabela.

A configuração é`mapping_mode = "single-table"`. Há duas configurações adicionais ao usar `single-table` `single_table_name` e. `single_table_dimension_name_for_telegraf_measurement_name`

- O plug-in Timestream for LiveAnalytics output criará uma única tabela com nome *`<single_table_name>`* que inclui um *`<single_table_dimension_name_for_telegraf_measurement_name>`* coluna.
- A tabela pode conter vários campos em uma única linha da tabela.

O Timestream resultante para a LiveAnalytics tabela terá a seguinte aparência.

`weather`

horário	local	temporada	*<single_table_dimension_name_for_telegraf_measurement_name>*	nome_medida	valor_medida::bigint
2016-06-13 17:43:50	centro-oeste dos EUA	verão	tempo	temperatura	82
2016-06-13 17:43:50	centro-oeste dos EUA	verão	tempo	umidade	71
2016-06-13 17:43:50	centro-oeste dos EUA	verão	qualidade do ar	não 2	5
2016-06-13 17:43:50	centro-oeste dos EUA	verão	tempo	pm25	16

JDBC

Você pode usar uma conexão Java Database Connectivity (JDBC) para conectar o Timestream LiveAnalytics às suas ferramentas de business intelligence e outros aplicativos, como SQL o Workbench. Atualmente, o Timestream for LiveAnalytics JDBC driver é compatível SSO com Okta e Microsoft Azure AD.

Tópicos

- Configurando o JDBC driver do Timestream para LiveAnalytics
- Propriedades de conexão
- JDBCURLexemplos
- Configurando o Timestream para autenticação de login LiveAnalytics JDBC único com o Okta
- Configurando o Timestream para autenticação de logon LiveAnalytics JDBC único com o Microsoft Azure AD

Configurando o JDBC driver do Timestream para LiveAnalytics

Siga as etapas abaixo para configurar o JDBC driver.

Tópicos

- Timestream para o motorista LiveAnalytics JDBC JARs
- Cronograma para classe e LiveAnalytics JDBC formato do motorista URL
- Aplicação de exemplo

Timestream para o motorista LiveAnalytics JDBC JARs

Você pode obter o Timestream para o LiveAnalytics JDBC driver por meio de download direto ou adicionando o driver como uma dependência do Maven.

- Como download direto:. Para baixar diretamente o Timestream para o LiveAnalytics JDBC driver, conclua as seguintes etapas:

  1. Navegue até https://github.com/awslabs/amazon-timestream-driver-jdbc/releases
  2. Você pode usar `amazon-timestream-jdbc-1.0.1-shaded.jar` diretamente com suas ferramentas e aplicativos de business intelligence
  3. Faça o download `amazon-timestream-jdbc-1.0.1-javadoc.jar` para um diretório de sua escolha.
  4. No diretório em que você baixou`amazon-timestream-jdbc-1.0.1-javadoc.jar`, execute o seguinte comando para extrair os arquivos JavadocHTML:

     ```
     jar -xvf amazon-timestream-jdbc-1.0.1-javadoc.jar
     ```

- Como uma dependência do Maven: Para adicionar o Timestream for LiveAnalytics JDBC driver como uma dependência do Maven, conclua as seguintes etapas:

  1. Navegue até o `pom.xml` arquivo do seu aplicativo e abra-o em um editor de sua escolha.
  2. Adicione o JDBC driver como uma dependência ao `pom.xml` arquivo do seu aplicativo:

     ```
     <!-- 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>
```

Cronograma para classe e LiveAnalytics JDBC formato do motorista URL

A classe de driver para Timestream for LiveAnalytics JDBC driver é:

```
software.amazon.timestream.jdbc.TimestreamDriver
```

O JDBC driver Timestream requer o seguinte formato: JDBC URL

```
jdbc:timestream:
```

Para especificar as propriedades do banco de dados por meio do JDBCURL, use o seguinte URL formato:

```
jdbc:timestream://
```

Aplicação de exemplo

Para ajudar você a começar a usar o Timestream for LiveAnalytics withJDBC, criamos um aplicativo de amostra totalmente funcional em. GitHub

1. Crie um banco de dados com dados de amostra seguindo as instruções descritas aqui.
2. Clone o GitHub repositório do aplicativo de amostra para JDBC seguir as instruções de. GitHub
3. Siga as instruções em READMEpara começar a usar o aplicativo de amostra.

Propriedades de conexão

O Timestream for LiveAnalytics JDBC driver suporta as seguintes opções:

Tópicos

- Opções básicas de autenticação
- Opção padrão de informações do cliente
- Opção de configuração do driver
- SDKopção

- Opção de configuração de endpoint
- Opções de provedores de credenciais
- SAMLopções de autenticação baseadas para Okta
- SAMLopções de autenticação baseadas para Azure AD

> **Note**
>
> Se nenhuma das propriedades for fornecida, o Timestream for LiveAnalytics JDBC driver usará a cadeia de credenciais padrão para carregar as credenciais.

> **Note**
>
> Todas as chaves de propriedade diferenciam maiúsculas de minúsculas.

Opções básicas de autenticação

A tabela a seguir descreve as opções de autenticação básica disponíveis.

Opção	Descrição	Padrão
AccessKeyId	O ID da chave de acesso do AWS usuário.	NONE
SecretAccessKey	A chave de acesso secreta do AWS usuário.	NONE
SessionToken	O token de sessão temporári a necessário para acessar um banco de dados com a autenticação multifator (MFA) ativada.	NONE

Opção padrão de informações do cliente

A tabela a seguir descreve a opção Standard Client Info.

Opção	Descrição	Padrão
ApplicationName	O nome do aplicativo que está utilizando a conexão no momento. `ApplicationName` é usado para fins de depuração e não será comunicado ao Timestream para fins de serviço. LiveAnalytics	O nome do aplicativo detectado pelo driver.

Opção de configuração do driver

A tabela a seguir descreve a opção de configuração do driver.

Opção	Descrição	Padrão
EnableMetaDataPreparedStatement	Permite que o LiveAnalytics JDBC motorista retorne metadados para o Timestream `PreparedStatements`, mas isso acarretará um custo adicional com o Timestream para recuperar os metadados. LiveAnalytics	FALSE
Região	A região do banco de dados.	us-east-1

SDKopção

A tabela a seguir descreve a SDK Opção.

Opção	Descrição	Padrão
RequestTimeout	O tempo em milissegundos em que AWS SDK aguardará	0

Opção	Descrição	Padrão
	uma solicitação de consulta antes de atingir o tempo limite. O valor não positivo desativa o tempo limite da solicitação.	
SocketTimeout	O tempo em milissegundos em que AWS SDK esperará que os dados sejam transferi dos por uma conexão aberta antes de atingir o tempo limite. O valor não deve ser negativo. Um valor de `0` desativa o tempo limite do soquete.	50000
MaxRetryCountClient	O número máximo de tentativa s de repetição para erros que podem ser repetidos com códigos de erro 5XX no. SDK O valor não deve ser negativo.	NONE
MaxConnections	O número máximo permitido de HTTP conexões abertas simultaneamente com o Timestream para serviço. LiveAnalytics O valor deve ser positivo.	50

Opção de configuração de endpoint

A tabela a seguir descreve a opção de configuração do endpoint.

Opção	Descrição	Padrão
Endpoint	O endpoint do Timestream for service. LiveAnalytics	NONE

Opções de provedores de credenciais

A tabela a seguir descreve as opções disponíveis do provedor de credenciais.

Opção	Descrição	Padrão
AwsCredentialsProviderClass	Um dos `PropertiesFileCredentialsProvider` ou `InstanceProfileCredentialsProvider` para usar para autenticação.	NONE
CustomCredentialsFilePath	O caminho para um arquivo de propriedades contendo credenciais de AWS segurança `accessKey` e. `secretKey` Isso só é necessário se `AwsCredentialsProviderClass` for especificado como`PropertiesFileCredentialsProvider`.	NONE

SAMLopções de autenticação baseadas para Okta

A tabela a seguir descreve as opções de autenticação SAML baseada disponíveis para o Okta.

Opção	Descrição	Padrão
IdpName	O nome do provedor de identidade (Idp) a ser usado para autenticação SAML baseada. Um dos `Okta` ou`AzureAD`.	NONE

Opção	Descrição	Padrão
IdpHost	O nome do host do Idp especificado.	NONE
IdpUserName	O nome de usuário da conta Idp especificada.	NONE
IdpPassword	A senha da conta Idp especificada.	NONE
OktaApplicationID	O ID exclusivo fornecido pela Okta associado ao Timestream do aplicativo. LiveAnalytics `AppId`podem ser encontrados no `entityID` campo fornecido nos metadados do aplicativo. Considere o exemplo a seguir: `entityID = http://www.okta.com//IdpAppID`	NONE
Função ARN	O Amazon Resource Name (ARN) da função que o chamador está assumindo.	NONE
Idp ARN	O Amazon Resource Name (ARN) do SAML provedor IAM que descreve o Idp.	NONE

SAMLopções de autenticação baseadas para Azure AD

A tabela a seguir descreve as opções de autenticação SAML baseada disponíveis para o Azure AD.

Opção	Descrição	Padrão
IdpName	O nome do provedor de identidade (Idp) a ser usado	NONE

Opção	Descrição	Padrão
	para autenticação SAML baseada. Um dos `Okta` ou`AzureAD`.	
IdpHost	O nome do host do Idp especificado.	NONE
IdpUserName	O nome de usuário da conta Idp especificada.	NONE
IdpPassword	A senha da conta Idp especific ada.	NONE
AADApplicationID	O ID exclusivo do aplicativo registrado no Azure AD.	NONE
AADClientSecret	O segredo do cliente associado ao aplicativo registrado no Azure AD usado para autorizar a busca de tokens.	NONE
AADTenant	O ID do locatário do Azure AD.	NONE
Idp ARN	O Amazon Resource Name (ARN) do SAML provedor IAM que descreve o Idp.	NONE

JDBCURLexemplos

Esta seção descreve como criar uma JDBC conexão URL e fornece exemplos. Para especificar as propriedades de conexão opcionais, use o seguinte URL formato:

```
jdbc:timestream://PropertyName1=value1;PropertyName2=value2...
```

> **Note**
>
> Todas as propriedades da conexão são opcionais. Todas as chaves de propriedade diferenciam maiúsculas de minúsculas.

Abaixo estão alguns exemplos de JDBC conexãoURLs.

Exemplo com opções básicas de autenticação e região:

```
jdbc:timestream://
AccessKeyId=<myAccessKeyId>;SecretAccessKey=<mySecretAccessKey>;SessionToken=<mySessionToken
east-1
```

Exemplo com informações do cliente, região e SDK opções:

```
jdbc:timestream://ApplicationName=MyApp;Region=us-
east-1;MaxRetryCountClient=10;MaxConnections=5000;RequestTimeout=20000
```

Conecte-se usando a cadeia de provedores de credenciais padrão com AWS credenciais definidas nas variáveis de ambiente:

```
jdbc:timestream
```

Conecte-se usando a cadeia de provedores de credenciais padrão com a AWS credencial definida na conexão: URL

```
jdbc:timestream://
AccessKeyId=<myAccessKeyId>;SecretAccessKey=<mySecretAccessKey>;SessionToken=<mySessionToken
```

Conecte-se usando o PropertiesFileCredentialsProvider como método de autenticação:

```
jdbc:timestream://
AwsCredentialsProviderClass=PropertiesFileCredentialsProvider;CustomCredentialsFilePath=<pat
 to properties file>
```

Conecte-se usando o InstanceProfileCredentialsProvider como método de autenticação:

```
jdbc:timestream://AwsCredentialsProviderClass=InstanceProfileCredentialsProvider
```

Conecte-se usando as credenciais do Okta como método de autenticação:

```
jdbc:timestream:// 
IdpName=Okta;IdpHost=<host>;IdpUserName=<name>;IdpPassword=<password>;OktaApplicationID=<id>
```

Conecte-se usando as credenciais do Azure AD como método de autenticação:

```
jdbc:timestream:// 
IdpName=AzureAD;IdpUserName=<name>;IdpPassword=<password>;AADApplicationID=<id>;AADClientSec
```

Conecte-se com um endpoint específico:

```
jdbc:timestream://Endpoint=abc.us-east-1.amazonaws.com;Region=us-east-1
```

Configurando o Timestream para autenticação de login LiveAnalytics JDBC único com o Okta

Timestream for LiveAnalytics suporta Timestream para autenticação de login LiveAnalytics JDBC único com Okta. Para usar o Timestream para autenticação de login LiveAnalytics JDBC único com o Okta, preencha cada uma das seções listadas abaixo.

Tópicos

- Pré-requisitos
- AWS federação de contas em Okta
- Configurando o Okta para SAML

Pré-requisitos

Verifique se você atendeu aos seguintes pré-requisitos antes de usar o Timestream para autenticação de login LiveAnalytics JDBC único com o Okta:

- Permissões de administrador AWS para criar o provedor de identidade e as funções.
- Uma conta Okta (acesse https://www.okta.com/login/para criar uma conta).
- Acesso ao Amazon LiveAnalytics Timestream para.

Agora que você concluiu os pré-requisitos, prossiga para. AWS federação de contas em Okta

AWS federação de contas em Okta

O Timestream for LiveAnalytics JDBC driver suporta a federação de AWS contas em Okta. Para configurar a Federação de AWS Contas no Okta, conclua as seguintes etapas:

1. Faça login no painel do Okta Admin usando o seguinte: URL

```
https://<company-domain-name>-admin.okta.com/admin/apps/active
```

> **Note**
>
> Substitua < company-domain-name > pelo nome do seu domínio.

2. Após o login bem-sucedido, escolha Adicionar aplicativo e pesquise por Federação de AWS contas.
3. Escolha Adicionar
4. Altere o Login URL para o apropriadoURL.
5. Escolha Avançar.
6. Escolha SAML2.0 como método de login
7. Escolha os metadados do provedor de identidade para abrir o arquivo de metadadosXML. Salve o arquivo localmente.
8. Deixe todas as outras opções de configuração em branco.
9. Escolha Concluído

Agora que você concluiu a Federação de AWS Contas em Okta, você pode prosseguir paraConfigurando o Okta para SAML.

Configurando o Okta para SAML

1. Escolha a guia Sign On (Fazer logon). Escolha a visualização.
2. Escolha o botão Instruções de configuração na seção Configurações.

Encontrando o documento de metadados do Okta

1. Para encontrar o documento, acesse:

```
https://<domain>-admin.okta.com/admin/apps/active
```

> **Note**
>
> <domain>é seu nome de domínio exclusivo para sua conta Okta.

2. Escolha o aplicativo de federação de AWS contas
3. Escolha a guia Sign On

Configurando o Timestream para autenticação de logon LiveAnalytics JDBC único com o Microsoft Azure AD

Timestream for LiveAnalytics suporta Timestream para autenticação de logon LiveAnalytics JDBC único com o Microsoft Azure AD. Para usar o Timestream para autenticação de logon LiveAnalytics JDBC único com o Microsoft Azure AD, conclua cada uma das seções listadas abaixo.

Tópicos

- Pré-requisitos
- Configurando o Azure AD
- Configurando o provedor de IAM identidade e as funções no AWS

Pré-requisitos

Verifique se você atendeu aos seguintes pré-requisitos antes de usar o Timestream para autenticação de LiveAnalytics JDBC logon único com o Microsoft Azure AD:

- Permissões de administrador AWS para criar o provedor de identidade e as funções.
- Uma conta do Azure Active Directory (acesse https://azure.microsoft.com/en-ca/services/active-directory/ para criar uma conta)
- Acesso ao Amazon LiveAnalytics Timestream para.

Configurando o Azure AD

1. Entre no Portal do Azure

2. Escolha Azure Active Directory na lista de serviços do Azure. Isso redirecionará para a página Diretório padrão.
3. Escolha Aplicativos corporativos na seção Gerenciar na barra lateral
4. Escolha + Novo aplicativo.
5. Encontre e selecione Amazon Web Services.
6. Escolha Login único na seção Gerenciar na barra lateral
7. Escolha SAMLcomo método de login único
8. Na seção SAML Configuração básica, insira o seguinte URL para o Identificador e a RespostaURL:

```
https://signin.aws.amazon.com/saml
```

9. Escolha Salvar
10. Baixe os metadados da federação XML na seção Certificado de SAML assinatura. Isso será usado ao criar o provedor de IAM identidade posteriormente.
11. Volte para a página Diretório padrão e escolha Registros de aplicativos em Gerenciar.
12. Escolha Timestream para na LiveAnalytics seção Todos os aplicativos. A página será redirecionada para a página de visão geral do aplicativo

> **Note**
>
> Anote o ID do aplicativo (cliente) e o ID do diretório (inquilino). Esses valores são necessários para criar uma conexão.

13. Escolha certificados e segredos
14. Em Segredos do cliente, crie um novo segredo do cliente com + Novo segredo do cliente.

> **Note**
>
> Observe o segredo do cliente gerado, pois isso é necessário ao criar uma conexão com o Timestream for. LiveAnalytics

15. Na barra lateral, em Gerenciar, selecione permissões API
16. Nas permissões configuradas, use Adicionar uma permissão para conceder permissão ao Azure AD para entrar no Timestream para. LiveAnalytics Escolha Microsoft Graph na página Solicitar API permissões.

17 Escolha Permissões delegadas e selecione a permissão User.Read

18 Escolha Adicionar permissões

19 Escolha Conceder consentimento do administrador para o Diretório padrão

Configurando o provedor de IAM identidade e as funções no AWS

Conclua cada seção abaixo para configurar o Timestream IAM para autenticação de logon LiveAnalytics JDBC único com o Microsoft Azure AD:

Tópicos

- Crie um provedor SAML de identidade
- Crie uma IAM função
- Crie uma IAM política
- Provisionamento

Crie um provedor SAML de identidade

Para criar um provedor de SAML identidade para o Timestream para autenticação de logon LiveAnalytics JDBC único com o Microsoft Azure AD, conclua as seguintes etapas:

1. Faça login no console AWS de gerenciamento
2. Escolha Serviços e selecione IAMem Segurança, Identidade e Conformidade
3. Escolha provedores de identidade em Gerenciamento de acesso
4. Escolha Criar provedor e escolha SAMLcomo tipo de provedor. Insira o nome do provedor. Este exemplo usará zureADProvider A.
5. Carregue o arquivo de XML metadados da federação baixado anteriormente
6. Escolha Avançar e, em seguida, escolha Criar.
7. Após a conclusão, a página será redirecionada de volta para a página Provedores de identidade

Crie uma IAM função

Para criar uma IAM função para o Timestream para autenticação de logon LiveAnalytics JDBC único com o Microsoft Azure AD, conclua as seguintes etapas:

1. Na barra lateral, selecione Funções em Gerenciamento de acesso.

2. Selecione Create role (Criar função)
3. Escolha a federação SAML 2.0 como entidade confiável
4. Escolha o provedor do Azure AD
5. Escolha Permitir acesso programático e ao console AWS de gerenciamento
6. Escolha Próximo: Permissões
7. Anexe políticas de permissões ou continue em Next:Tags
8. Adicione tags opcionais ou continue em Next:Review
9. Insira um Role name. Este exemplo usará A. zureSAMLRole
10. Forneça uma descrição da função
11. Escolha Criar função para concluir

Crie uma IAM política

Para criar uma IAM política para o Timestream para autenticação de logon LiveAnalytics JDBC único com o Microsoft Azure AD, conclua as seguintes etapas:

1. Na barra lateral, escolha Políticas em Gerenciamento de acesso
2. Escolha Criar política e selecione a JSONguia
3. Adicione a seguinte política

```
{
"Version": "2012-10-17",
"Statement": [
    {
          "Effect": "Allow",
          "Action": [
                "iam:ListRoles",
                "iam:ListAccountAliases"
           ],
           "Resource": "*"
      }
]
}
```

4. Escolha Create policy (Criar política).
5. Insira um nome de política. Este exemplo usará TimestreamAccessPolicy.
6. Escolha Criar política

7. Na barra lateral, escolha Funções em Gerenciamento de acesso.
8. Escolha a função do Azure AD criada anteriormente e escolha Anexar políticas em Permissões.
9. Selecione a política de acesso criada anteriormente.

Provisionamento

Para provisionar o provedor de identidade do Timestream para autenticação de logon LiveAnalytics JDBC único com o Microsoft Azure AD, conclua as seguintes etapas:

1. Volte para o Portal do Azure
2. Escolha Azure Active Directory na lista de serviços do Azure. Isso redirecionará para a página Diretório padrão
3. Escolha Aplicativos corporativos na seção Gerenciar na barra lateral
4. Escolha o provisionamento
5. Escolha o modo Automático para o Método de Provisionamento
6. Em Credenciais de administrador, insira seu AwsAccessKeyID para clientsecret e SecretAccessKeySecret Token
7. Defina o status do aprovisionamento como Ativado
8. Escolha salvar. Isso permite que o Azure AD carregue as IAM funções necessárias
9. Quando o status do ciclo atual estiver concluído, escolha Usuários e grupos na barra lateral
10. Escolha + Adicionar usuário
11. Escolha o usuário do Azure AD para fornecer acesso ao Timestream para LiveAnalytics
12. Escolha a função IAM do Azure AD e o Provedor de Identidade do Azure correspondente criado em AWS
13. Escolha Atribuir

ODBC

O ODBCdriver de código aberto do Amazon LiveAnalytics Timestream for SQL fornece uma interface -relacional para o LiveAnalytics Timestream para desenvolvedores e permite a conectividade a partir de ferramentas de business intelligence (BI), como Power BI Desktop e Microsoft Excel. O Timestream for LiveAnalytics ODBC driver está atualmente disponível no Windows, macOS e Linux e também é compatível com SSO Okta e Microsoft Azure Active Directory (AD).

Para obter mais informações, consulte Amazon Timestream LiveAnalytics ODBC para obter a documentação do motorista sobre. GitHub

Tópicos

- Configurando o Timestream para o motorista LiveAnalytics ODBC
- Sintaxe da cadeia de conexão e opções para o driver ODBC
- Exemplos de cadeia de conexão para o Timestream for driver LiveAnalytics ODBC
- Solução de problemas de conexão com o ODBC driver

Configurando o Timestream para o motorista LiveAnalytics ODBC

Configure o acesso ao Timestream LiveAnalytics em sua conta AWS

Se você ainda não configurou sua AWS conta para trabalhar com o Timestream LiveAnalytics, siga as instruções em. Acessando o Timestream para LiveAnalytics

Instale o ODBC driver no seu sistema

Baixe o instalador do ODBC driver Timestream apropriado para o seu sistema a partir do ODBC GitHubrepositório, e siga as instruções de instalação que se aplicam ao seu sistema:.

- Guia de instalação do Windows
- Guia de instalação do macOS
- Guia de instalação do Linux

Configurar um nome de fonte de dados (DSN) para o ODBC driver

Siga as instruções no guia DSN de configuração do seu sistema:

- DSNConfiguração do Windows
- Configuração do macOS DSN
- DSNConfiguração Linux

Configure seu aplicativo de business intelligence (BI) para trabalhar com o ODBC motorista

Aqui estão as instruções para configurar vários aplicativos comuns de BI para trabalhar com o ODBC driver:

- Configurando o Microsoft Power BI.
- Configurando o Microsoft Excel
- Configurando o Tableau

Para outras aplicações

Sintaxe da cadeia de conexão e opções para o driver ODBC

A sintaxe para especificar as opções de cadeia de conexão para o driver é a ODBC seguinte:

```
DRIVER={Amazon Timestream ODBC Driver};(option)=(value);
```

As opções disponíveis são as seguintes:

Opções de conexão do driver

- **`Driver`**(obrigatório) — O driver com o qual está sendo usadoODBC.

  O padrão é Amazon Timestream.
- **`DSN`**— O nome da fonte de dados (DSN) a ser usado para configurar a conexão.

  O padrão é `NONE`.
- **`Auth`**— O modo de autenticação. Este valor deve ser um dos seguintes:
  - `AWS_PROFILE`— Use a cadeia de credenciais padrão.
  - `IAM`— Use AWS IAM credenciais.
  - `AAD`— Use o provedor de identidade do Azure Active Directory (AD).
  - `OKTA`— Use o provedor de identidade Okta.

  O padrão é `AWS_PROFILE`.

Opções de configuração do endpoint

- **`EndpointOverride`**— A substituição do endpoint para o Timestream for service. LiveAnalytics Essa é uma opção avançada que substitui a região. Por exemplo:

  ```
  query-cell2.timestream.us-east-1.amazonaws.com
  ```
- **`Region`**— A região de assinatura do Timestream for LiveAnalytics Service Endpoint.

O padrão é `us-east-1`.

Opção de provedor de credenciais

- **`ProfileName`**— O nome do perfil no arquivo de AWS configuração.

  O padrão é `NONE`.

AWS IAMopções de autenticação

- **`UID`**ou **`AccessKeyId`**— O ID da chave de acesso do AWS usuário. Se ambos `UID` `AccessKeyId` forem fornecidos na cadeia de conexão, o `UID` valor será usado, a menos que esteja vazio.

  O padrão é `NONE`.
- **`PWD`**ou **`SecretKey`**— A chave de acesso secreta AWS do usuário. Se ambos `PWD` `SecretKey` forem fornecidos na string de conexão, o `PWD` valor with será usado, a menos que esteja vazio.

  O padrão é `NONE`.
- **`SessionToken`**— O token de sessão temporário necessário para acessar um banco de dados com a autenticação multifator (MFA) ativada. Não inclua um final = na entrada.

  O padrão é `NONE`.

SAMLopções de autenticação baseadas para Okta

- **`IdPHost`**— O nome do host do IdP especificado.

  O padrão é `NONE`.
- **`UID`**ou **`IdPUserName`**— O nome de usuário da conta IdP especificada. Se ambos `UID` `IdPUserName` forem fornecidos na cadeia de conexão, o `UID` valor será usado, a menos que esteja vazio.

  O padrão é `NONE`.
- **`PWD`**ou **`IdPPassword`**— A senha da conta IdP especificada. Se ambos `PWD` `IdPPassword` forem fornecidos na cadeia de conexão, o `PWD` valor será usado, a menos que esteja vazio.

  O padrão é `NONE`.

- **OktaApplicationID**— O ID exclusivo fornecido pela Okta associado ao Timestream do aplicativo. LiveAnalytics Um local para encontrar o ID do aplicativo (AppId) está no `entityID` campo fornecido nos metadados do aplicativo. Um exemplo é:

  ```
  entityID="http://www.okta.com//(IdPAppID)
  ```

  O padrão é `NONE`.
- **RoleARN**— O Amazon Resource Name (ARN) da função que o chamador está assumindo.

  O padrão é `NONE`.
- **IdPARN**— O Amazon Resource Name (ARN) do SAML provedor IAM que descreve o IdP.

  O padrão é `NONE`.

SAMLopções de autenticação baseadas para o Azure Active Directory

- **UID**ou **IdPUserName**— O nome de usuário da conta IdP especificada.

  O padrão é `NONE`.
- **PWD**ou **IdPPassword**— A senha da conta IdP especificada.

  O padrão é `NONE`.
- **AADApplicationID**— O ID exclusivo do aplicativo registrado no Azure AD.

  O padrão é `NONE`.
- **AADClientSecret**— O segredo do cliente associado ao aplicativo registrado no Azure AD usado para autorizar a busca de tokens.

  O padrão é `NONE`.
- **AADTenant**— O ID do locatário do Azure AD.

  O padrão é `NONE`.
- **RoleARN**— O Amazon Resource Name (ARN) da função que o chamador está assumindo.

  O padrão é `NONE`.
- **IdPARN**— O Amazon Resource Name (ARN) do SAML provedor IAM que descreve o IdP.

  O padrão é `NONE`.

AWS SDKOpções (avançadas)

- **`RequestTimeout`**— O tempo em milissegundos em que o usuário AWS SDK espera por uma solicitação de consulta antes de atingir o tempo limite. Qualquer valor não positivo desativa o tempo limite da solicitação.

  O padrão é `3000`.
- **`ConnectionTimeout`**— O tempo em milissegundos em que os AWS SDK dados são transferidos por uma conexão aberta antes de atingir o tempo limite. Um valor de 0 desativa o tempo limite da conexão. Esse valor não deve ser negativo.

  O padrão é `1000`.
- **`MaxRetryCountClient`**— O número máximo de tentativas para erros que podem ser repetidos com códigos de erro 5xx no. SDK O valor não deve ser negativo.

  O padrão é `0`.
- **`MaxConnections`**— O número máximo de HTTP conexões abertas simultaneamente permitidas com o serviço Timestream. O valor deve ser positivo.

  O padrão é `25`.

ODBCOpções de registro de drivers

- **`LogLevel`**— O nível de registro para registro do driver. Deve ser um dos seguintes:
  - (0OFF).
  - (1ERROR).
  - (2WARNING).
  - (3INFO).
  - (4DEBUG).

  O padrão é 1 (ERROR).

  Aviso: as informações pessoais podem ser registradas pelo motorista ao usar o modo de DEBUG registro.
- **`LogOutput`**— Pasta na qual armazenar o arquivo de log.

  O padrão é:

- Windows:`%USERPROFILE%`, ou se não estiver disponível,`%HOMEDRIVE%%HOMEPATH%`.
- macOS e Linux: ou`$HOME`, se não estiver disponível, o campo do valor `pw_dir` de `getpwuid(getuid())` retorno da função.

SDKopções de registro

O nível de AWS SDK registro é separado do Timestream para o nível de registro LiveAnalytics ODBC do driver. Definir um não afeta o outro.

O nível de SDK log é definido usando a variável de ambiente`TS_AWS_LOG_LEVEL`. Os valores válidos são:

- `OFF`
- `ERROR`
- `WARN`
- `INFO`
- `DEBUG`
- `TRACE`
- `FATAL`

Se não `TS_AWS_LOG_LEVEL` estiver definido, o nível do SDK log será definido como o padrão, que é`WARN`.

Conectando-se por meio de um proxy

O ODBC driver suporta a conexão com o Amazon Timestream LiveAnalytics por meio de um proxy. Para usar esse recurso, configure as seguintes variáveis de ambiente com base na sua configuração de proxy:

- **`TS_PROXY_HOST`**— o host proxy.
- **`TS_PROXY_PORT`**— O número da porta proxy.
- **`TS_PROXY_SCHEME`**— O esquema de proxy, `http` ou`https`.
- **`TS_PROXY_USER`**— O nome de usuário para autenticação por proxy.
- **`TS_PROXY_PASSWORD`**— A senha do usuário para autenticação por proxy.

- **TS_PROXY_SSL_CERT_PATH**— O arquivo SSL de certificado a ser usado para se conectar a um HTTPS proxy.
- **TS_PROXY_SSL_CERT_TYPE**— O tipo do SSL certificado do cliente proxy.
- **TS_PROXY_SSL_KEY_PATH**— O arquivo de chave privada a ser usado para se conectar a um HTTPS proxy.
- **TS_PROXY_SSL_KEY_TYPE**— O tipo do arquivo de chave privada usado para se conectar a um HTTPS proxy.
- **TS_PROXY_SSL_KEY_PASSWORD**— A frase secreta do arquivo de chave privada usado para se conectar a um proxy. HTTPS

Exemplos de cadeia de conexão para o Timestream for driver LiveAnalytics ODBC

Exemplo de conexão com o ODBC motorista com IAM credenciais

```
Driver={Amazon Timestream ODBC Driver};Auth=IAM;AccessKeyId=(your access key
 ID);secretKey=(your secret key);SessionToken=(your session token);Region=us-east-2;
```

Exemplo de conexão com o ODBC motorista com um perfil

```
Driver={Amazon Timestream ODBC Driver};ProfileName=(the profile name);region=us-west-2;
```

O driver tentará se conectar usando as credenciais fornecidas em ou`~/.aws/credentials`, se um arquivo for especificado na variável de ambiente`AWS_SHARED_CREDENTIALS_FILE`, usando as credenciais desse arquivo.

Exemplo de conexão com o ODBC driver com o 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);
```

Exemplo de conexão com o ODBC driver com o 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);
```

Exemplo de conexão com o ODBC driver com um endpoint especificado e um nível de log de 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;
```

Solução de problemas de conexão com o ODBC driver

> **Note**
>
> Quando o nome de usuário e a senha já estão especificados noDSN, não há necessidade de especificá-los novamente quando o gerente do ODBC motorista os solicitar.

Um código de erro `01S02` com uma mensagem `Re-writing (connection string option) (have you specified it several times?` ocorre quando uma opção de cadeia de conexão é passada mais de uma vez na cadeia de conexão. Especificar uma opção mais de uma vez gera um erro. Ao fazer uma conexão com a DSN e uma string de conexão, se uma opção de conexão já estiver especificada noDSN, não a especifique novamente na string de conexão.

VPCpontos finais ()AWS PrivateLink

Você pode estabelecer uma conexão privada entre você VPC e o Amazon Timestream criando um endpoint LiveAnalytics de interface. VPC Para obter mais informações, consulte VPCpontos finais ()AWS PrivateLink.

Práticas recomendadas

Para aproveitar plenamente os benefícios do Amazon Timestream LiveAnalytics for, siga as melhores práticas descritas abaixo.

> **Note**
>
> Ao executar proof-of-concept aplicativos, considere a quantidade de dados que seu aplicativo acumulará em alguns meses ou anos enquanto avalia o desempenho e a escala do Timestream. LiveAnalytics À medida que seus dados crescem com o tempo, você notará que o desempenho do Timestream for LiveAnalytics permanece praticamente inalterado porque sua arquitetura sem servidor pode aproveitar grandes quantidades de paralelismo

para processar grandes volumes de dados e escalar automaticamente para atender às necessidades do seu aplicativo.

Tópicos

- Modelagem de dados
- Segurança
- Configurando o Amazon Timestream para LiveAnalytics
- Escreve
- Consultas
- Consultas programadas
- Aplicativos cliente e integrações suportadas
- Geral

Modelagem de dados

O Amazon Timestream LiveAnalytics for foi projetado para coletar, armazenar e analisar dados de séries temporais de aplicativos e dispositivos que emitem uma sequência de dados com um timestamp. Para um desempenho ideal, os dados enviados ao Timestream LiveAnalytics devem ter características temporais e o tempo deve ser um componente essencial dos dados.

O Timestream for LiveAnalytics fornece a flexibilidade de modelar seus dados de maneiras diferentes para atender aos requisitos do seu aplicativo. Nesta seção, abordamos vários desses padrões e fornecemos diretrizes para você otimizar seus custos e desempenho. Familiarize-se com os principais LiveAnalytics conceitos do Timestream, como dimensões e medidas. Nesta seção, você aprenderá mais sobre:

Ao decidir se deseja criar uma única tabela ou várias tabelas para armazenar dados, considere o seguinte:

- Quais dados colocar na mesma tabela versus quando você deseja separar os dados em várias tabelas e bancos de dados.
- Como escolher entre o Timestream para registros de LiveAnalytics várias medidas versus registros de medida única e os benefícios da modelagem usando registros de várias medidas, especialmente quando seu aplicativo está rastreando várias medições ao mesmo tempo e instante.
- Quais atributos modelar como dimensões ou medidas.

- Como usar com eficiência os atributos do nome da medida para otimizar a latência da consulta.

Tópicos

- Tabela única versus várias tabelas
- Registros de várias medidas versus registros de medida única
- Dimensões e medidas
- Usando o nome da medida com registros de várias medidas
- Recomendações para particionar registros de várias medidas

Tabela única versus várias tabelas

Ao modelar seus dados no aplicativo, outro aspecto importante é como modelar os dados em tabelas e bancos de dados. Bancos de dados e tabelas no Timestream for LiveAnalytics são abstrações para controle de acesso, especificando KMS chaves, períodos de retenção etc. O Timestream para particionar LiveAnalytics automaticamente seus dados e foi projetado para escalar os recursos de acordo com a ingestão, o armazenamento e a carga de consultas e os requisitos de seus aplicativos.

Uma tabela no Timestream for LiveAnalytics pode ser dimensionada para petabytes de dados armazenados, dezenas de gigabytes/seg de gravações de dados e as consultas podem processar centenas de por hora. TBs As consultas no Timestream for LiveAnalytics podem abranger várias tabelas e bancos de dados, fornecendo uniões e uniões para fornecer acesso contínuo aos seus dados em várias tabelas e bancos de dados. Portanto, a escala dos dados ou os volumes de solicitações geralmente não são a principal preocupação ao decidir como organizar seus dados no Timestream for. LiveAnalytics Abaixo estão algumas considerações importantes ao decidir quais dados colocar na mesma tabela versus em tabelas diferentes ou tabelas em bancos de dados diferentes.

- As políticas de retenção de dados (retenção de armazenamento de memória, retenção de armazenamento magnético etc.) são suportadas na granularidade de uma tabela. Portanto, os dados que exigem políticas de retenção diferentes precisam estar em tabelas diferentes.
- AWS KMS as chaves usadas para criptografar seus dados são configuradas no nível do banco de dados. Portanto, diferentes requisitos de chave de criptografia implicam que os dados precisarão estar em bancos de dados diferentes.
- O Timestream for LiveAnalytics oferece suporte ao controle de acesso baseado em recursos na granularidade de tabelas e bancos de dados. Considere seus requisitos de controle de acesso ao decidir quais dados você grava na mesma tabela versus em tabelas diferentes.

- Esteja ciente dos limites do número de dimensões, nomes de medidas e nomes de atributos de várias medidas ao decidir quais dados são armazenados em qual tabela.
- Considere a carga de trabalho e os padrões de acesso da consulta ao decidir como você organiza seus dados, pois a latência da consulta e a facilidade de escrever suas consultas dependerão disso.
  - Se você armazenar dados que consulta com frequência na mesma tabela, isso geralmente facilitará a maneira como você escreve suas consultas, para que você possa evitar a necessidade de escrever junções, uniões ou expressões de tabela comuns. Isso geralmente também resulta em menor latência de consulta. Você pode usar predicados em dimensões e nomes de medidas para filtrar os dados relevantes para as consultas.

    Por exemplo, considere um caso em que você armazena dados de dispositivos localizados em seis continentes. Se suas consultas acessam frequentemente dados de vários continentes para obter uma visão global agregada, armazenar dados desses continentes na mesma tabela resultará em consultas mais fáceis de escrever. Por outro lado, se você armazenar dados em tabelas diferentes, ainda poderá combinar os dados na mesma consulta. No entanto, precisará escrever uma consulta para unir os dados de várias tabelas.
  - O Timestream for LiveAnalytics usa particionamento e indexação adaptáveis em seus dados. Portanto, as consultas são cobradas apenas pelos dados relevantes para suas consultas. Por exemplo, se você tiver uma tabela armazenando dados de um milhão de dispositivos em seis continentes, se sua consulta tiver predicados do formulário `WHERE device_id = 'abcdef'` ou`WHERE continent = 'North America'`, as consultas serão cobradas apenas pelos dados do dispositivo ou do continente.
  - Sempre que possível, se você usar o nome da medida para separar dados na mesma tabela que não são emitidos ao mesmo tempo ou não são consultados com frequência, usando predicados como `WHERE measure_name = 'cpu'` em sua consulta, você não só obtém os benefícios da medição, mas o Timestream for também LiveAnalytics pode eliminar efetivamente as partições que não têm o nome da medida usado no predicado da consulta. Isso permite que você armazene dados relacionados com nomes de medidas diferentes na mesma tabela sem afetar a latência ou os custos da consulta e evita a distribuição dos dados em várias tabelas. O nome da medida é usado essencialmente para particionar os dados e remover partições irrelevantes para a consulta.

Registros de várias medidas versus registros de medida única

O Timestream for LiveAnalytics permite que você grave dados com várias medidas por registro (várias medidas) ou uma única medida por registro (medida única).

Registros de várias medidas

Em muitos casos de uso, um dispositivo ou aplicativo que você está rastreando pode emitir várias métricas ou eventos ao mesmo tempo. Nesses casos, você pode armazenar todas as métricas emitidas no mesmo timestamp no mesmo registro de várias medidas. Ou seja, todas as medidas armazenadas no mesmo registro de várias medidas aparecem como colunas diferentes na mesma linha de dados.

Considere, por exemplo, que seu aplicativo está emitindo métricas como cpu, memória, disk_iops de um dispositivo medido no mesmo instante. Veja a seguir um exemplo dessa tabela em que várias métricas emitidas no mesmo instante são armazenadas na mesma linha. Você verá que dois hosts estão emitindo as métricas uma vez a cada segundo.

Hostname	nome_medida	Tempo	cpu	Memória	iops de disco
Anfitrião-24GJU	métricas	2021-12-01 19:00:00	35	54,9	38,2
Anfitrião-24GJU	métricas	2021-12-01 19:00:01	36	58	39
Host-28GJU	métricas	2021-12-01 19:00:00	15	55	92
Host-28GJU	métricas	2021-12-01 19:00:01	16	50	40

Registros de medida única

Os registros de medida única são adequados quando seus dispositivos emitem métricas diferentes em períodos de tempo diferentes ou quando você está usando uma lógica de processamento personalizada que emite metrics/events at different time periods (for instance, when a device's reading/state alterações). Como cada medida tem um registro de data e hora exclusivo, as medidas

podem ser armazenadas em seus próprios registros no Timestream for. LiveAnalytics Por exemplo, considere um sensor de IoT, que monitora a temperatura e a umidade do solo, que emite um registro somente quando detecta uma alteração em relação à entrada relatada anteriormente. O exemplo a seguir fornece um exemplo desses dados sendo emitidos usando registros de medida única.

id_dispositivo	nome_medida	Tempo	valor_medida::duplo	valor_medida::bigint
sensor-sea478	temperatura	2021-12-01 19:22:32	35	NULL
sensor-sea478	temperatura	2021-12-01 18:07:51	36	NULL
sensor-sea478	umidade	2021-12-01 19:05:30	NULL	21
sensor-sea478	umidade	2021-12-01 19:00:01	NULL	23

Comparando registros de medida única e de várias medidas

O Timestream for LiveAnalytics fornece a flexibilidade de modelar seus dados como registros de medida única ou de várias medidas, dependendo dos requisitos e características do seu aplicativo. Uma única tabela pode armazenar registros de medida única e de várias medidas, se a sua aplicação assim o desejar. Em geral, quando seu aplicativo está emitindo várias medidas/eventos ao mesmo tempo instantâneo, a modelagem dos dados como registros de várias medidas geralmente é recomendada para acesso eficiente aos dados e armazenamento de dados econômico.

Por exemplo, se você considerar um caso de DevOps uso rastreando métricas e eventos de centenas de milhares de servidores, cada servidor emite periodicamente 20 métricas e 5 eventos, onde os eventos e métricas são emitidos no mesmo instante. Esses dados podem ser modelados usando registros de medida única ou usando registros de várias medidas (consulte o gerador de dados de código aberto para ver o esquema resultante). Para esse caso de uso, modelar os dados usando registros de várias medidas em comparação com registros de medida única resulta em:

- Medição de ingestão - registros de várias medidas resultam em cerca de 40% menos bytes de ingestão gravados.

- Ingestão em lotes - registros de várias medidas resultam no envio de lotes maiores de dados, o que significa que os clientes precisam de menos threads e menos CPU para processar a ingestão.
- Medição de armazenamento - registros de várias medidas resultam em cerca de 8 vezes menos armazenamento, resultando em uma economia significativa de armazenamento tanto na memória quanto no armazenamento magnético.
- Latência da consulta - registros de várias medidas resultam em menor latência de consulta para a maioria dos tipos de consulta quando comparados aos registros de medida única.
- Bytes medidos por consulta - Para consultas que digitalizam menos de 10 MB de dados, os registros de medida única e de várias medidas são comparáveis. Para consultas que acessam uma única medida e digitalizam dados de mais de 10 MB, registros de medida única geralmente resultam em bytes menores medidos. Para consultas que fazem referência a 3 ou mais medidas, registros de várias medidas resultam em bytes menores medidos.
- Facilidade de expressar consultas com várias medidas — Quando suas consultas fazem referência a várias medidas, modelar seus dados com registros de várias medidas resulta em consultas mais compactas e fáceis de escrever.

Os fatores anteriores variarão dependendo de quantas métricas você está monitorando, quantas dimensões seus dados têm, etc. Embora o exemplo anterior forneça alguns dados concretos para um exemplo, vemos muitos cenários de aplicativos e casos de uso em que, se seu aplicativo emitir várias medidas no mesmo instante, armazenar dados como registros de várias medidas é mais eficaz. Além disso, os registros de várias medidas oferecem a flexibilidade dos tipos de dados e do armazenamento de vários outros valores como contexto (por exemplo, armazenamento de solicitações IDs e registros de data e hora adicionais, que serão discutidos posteriormente).

Observe que um registro de várias medidas também pode modelar medidas esparsas, como o exemplo anterior para registros de medida única: você pode usar o measure_name para armazenar o nome da medida e usar um nome genérico de atributo de várias medidas, como value_double para armazenar `DOUBLE` medidas, value_bigint para armazenar medidas, value_timestamp para armazenar valores adicionais etc. `BIGINT` `TIMESTAMP`

Dimensões e medidas

Uma tabela no Timestream LiveAnalytics permite que você armazene dimensões (identificando atributos do dispositivo/dados que você está armazenando) e medidas (as métricas/valores que você está rastreando). Consulte Timestream para obter conceitos e obter mais detalhes. LiveAnalytics Ao modelar seu aplicativo no Timestream for LiveAnalytics, a forma como você mapeia seus dados em

dimensões e medidas afeta sua ingestão e latência de consulta. A seguir estão as diretrizes sobre como modelar seus dados como dimensões e medidas que você pode aplicar ao seu caso de uso.

Escolhendo dimensões

Os dados que identificam a fonte que está enviando os dados da série temporal são adequados naturalmente às dimensões, que são atributos que não mudam com o tempo. Por exemplo, se você tiver um servidor emitindo métricas, os atributos que identificam o servidor, como nome do host, região, rack e zona de disponibilidade, são candidatos às dimensões. Da mesma forma, para um dispositivo de IoT com vários sensores relatando dados de séries temporais, a identificação do dispositivo, a identificação do sensor etc. são candidatas às dimensões.

Se você estiver gravando dados como registros de várias medidas, as dimensões e os atributos de várias medidas aparecerão como colunas na tabela quando você fizer DESCRIBE ou executar uma SELECT declaração na tabela. Portanto, ao escrever suas consultas, você pode usar livremente as dimensões e medidas na mesma consulta. No entanto, ao criar seu registro de gravação para ingerir dados, lembre-se do seguinte ao escolher quais atributos são especificados como dimensões e quais são valores de medida:

- Os nomes das dimensões, os valores, o nome da medida e o carimbo de data/hora identificam de forma exclusiva os dados da série temporal. O Timestream for LiveAnalytics usa esse identificador exclusivo para desduplicar dados automaticamente. Ou seja, se o Timestream for LiveAnalytics receber dois pontos de dados com os mesmos valores de nomes de dimensão, valores de dimensão, nome de medida e timestamp, se os valores tiverem o mesmo número de versão, então Timestream for deduplicates. LiveAnalytics Se a nova solicitação de gravação tiver uma versão inferior aos dados já existentes no Timestream for LiveAnalytics, a solicitação de gravação será rejeitada. Se a nova solicitação de gravação tiver uma versão superior, o novo valor substituirá o valor antigo. Portanto, a forma como você escolhe seus valores de dimensão afetará esse comportamento de desduplicação.
- Os nomes e valores das dimensões não podem ser atualizados, o valor da medida pode ser. Portanto, qualquer dado que precise de atualizações é melhor modelado como valores de medida. Por exemplo, se você tiver uma máquina no chão de fábrica cuja cor pode mudar, você pode modelar a cor como um valor de medida, a menos que queira usar a cor também como atributo de identificação necessário para a desduplicação. Ou seja, os valores de medida podem ser usados para armazenar atributos que mudam apenas lentamente com o tempo.

Observe que uma tabela no Timestream para LiveAnalytics não limita o número de combinações exclusivas de nomes e valores de dimensões. Por exemplo, você pode ter bilhões dessas

combinações de valores exclusivos armazenadas em uma tabela. No entanto, como você verá nos exemplos a seguir, uma escolha cuidadosa de dimensões e medidas pode otimizar significativamente a latência da solicitação, especialmente para consultas.

Único IDs em dimensões

Se o cenário do seu aplicativo exigir que você armazene um identificador exclusivo para cada ponto de dados (por exemplo, um ID de solicitação, um ID de transação ou um ID de correlação), modelar o atributo ID como um valor de medida resultará em uma latência de consulta significativamente melhor. Ao modelar seus dados com registros de várias medidas, o ID aparece na mesma linha no contexto de suas outras dimensões e dados de séries temporais, para que suas consultas possam continuar a usá-los com eficiência. Por exemplo, considerando um caso de DevOps uso em que cada ponto de dados emitido por um servidor tem um atributo de ID de solicitação exclusivo, modelar o ID da solicitação como um valor de medida resulta em uma latência de consulta até 4 vezes menor em diferentes tipos de consulta, em vez de modelar o ID de solicitação exclusivo como uma dimensão.

Você pode usar a analogia semelhante para atributos que não são totalmente exclusivos para cada ponto de dados, mas têm centenas de milhares ou milhões de valores exclusivos. Você pode modelar esses atributos como dimensões ou valores de medida. Você gostaria de modelá-la como uma dimensão se os valores forem necessários para a desduplicação no caminho de gravação, conforme discutido anteriormente, ou se você costuma usá-la como um predicado (por exemplo, na `WHERE` cláusula com um predicado de igualdade em um valor desse atributo, como `device_id = 'abcde'` onde seu aplicativo está rastreando milhões de dispositivos) em suas consultas.

Riqueza de tipos de dados com registros de várias medidas

Os registros de várias medidas oferecem a flexibilidade de modelar seus dados com eficiência. Os dados que você armazena em um registro de várias medidas aparecem como colunas na tabela semelhantes às dimensões, oferecendo a mesma facilidade de consulta de dimensões e valores de medida. Você viu alguns desses padrões nos exemplos discutidos anteriormente. Abaixo, você encontrará padrões adicionais para usar com eficiência registros de várias medidas para atender aos casos de uso do seu aplicativo.

Os registros de várias medidas oferecem suporte a atributos dos tipos de dados `DOUBLE` `BIGINTVARCHAR`,,`BOOLEAN`, e. `TIMESTAMP` Portanto, eles se encaixam naturalmente em diferentes tipos de atributos:

- Informações de localização: por exemplo, se você quiser rastrear a localização (expressa como latitude e longitude), modelá-la como um atributo de várias medidas resultará em menor latência

de consulta em comparação com armazená-las como VARCHAR dimensões, especialmente quando você tem predicados sobre latitude e longitudes.

- Vários carimbos de data/hora em um registro: se o cenário do seu aplicativo exigir que você rastreie vários carimbos de data e hora para um registro de série temporal, você poderá modelá-los como atributos adicionais no registro de várias medidas. Esse padrão pode ser usado para armazenar dados com timestamps futuros ou antigos. Observe que cada registro ainda usará o timestamp na coluna de tempo para particionar, indexar e identificar um registro de forma exclusiva.

Em particular, se você tiver dados numéricos ou registros de data e hora nos quais tenha predicados na consulta, modelar esses atributos como atributos de várias medidas, em vez de dimensões, resultará em menor latência da consulta. Isso ocorre porque, ao modelar esses dados usando os tipos de dados avançados suportados em registros de várias medidas, você pode expressar os predicados usando tipos de dados nativos em vez de converter valores VARCHAR para outro tipo de dados se você modelou esses dados como dimensões.

Usando o nome da medida com registros de várias medidas

As tabelas no Timestream LiveAnalytics oferecem suporte a um atributo especial (ou coluna) chamado nome da medida. Você especifica um valor para esse atributo para cada registro para o qual você grava no Timestream. LiveAnalytics Para registros de medida única, é natural usar o nome de sua métrica (como CPU, memória para métricas de servidor ou temperatura, pressão para métricas de sensores). Ao usar registros de várias medidas, como os atributos em um registro de várias medidas são nomeados (e esses nomes se tornam nomes de colunas na tabela), cpu, memória ou temperatura, a pressão pode se tornar nomes de atributos de várias medidas. Portanto, uma questão natural é como usar efetivamente o nome da medida.

O Timestream for LiveAnalytics usa os valores no atributo nome da medida para particionar e indexar os dados. Portanto, se uma tabela tiver vários nomes de medidas diferentes e se as consultas usarem esses valores como predicados de consulta, o Timestream for LiveAnalytics poderá usar seu particionamento e indexação personalizados para eliminar dados que não sejam relevantes para as consultas. Por exemplo, se sua tabela tiver nomes de medidas de CPU e memória e sua consulta tiver um predicado`WHERE measure_name = 'cpu'`, o Timestream for LiveAnalytics poderá efetivamente remover dados de nomes de medidas não relevantes para a consulta, por exemplo, linhas com memória de nome de medida neste exemplo. Essa redução se aplica mesmo ao usar nomes de medidas com registros de várias medidas. Você pode usar o atributo de nome da medida de forma eficaz como um atributo de particionamento para uma tabela. O nome da medida

junto com os nomes e valores das dimensões e o tempo são usados para particionar os dados em um Timestream for LiveAnalytics table. Esteja ciente dos limites do número de nomes de medidas exclusivos permitidos em uma tabela Timestream for LiveAnalytics . Observe também que o nome de uma medida também está associado a um tipo de dados de valor de medida, por exemplo, um único nome de medida só pode ser associado a um tipo de valor de medida. Esse tipo pode ser um dos `DOUBLE` `BIGINT``BOOLEAN`,`VARCHAR`,, `MULTI` e. Registros de várias medidas armazenados com um nome de medida terão o tipo de dados como`MULTI`. Como um único registro de várias medidas pode armazenar várias métricas com diferentes tipos de dados (`DOUBLE`,, `BIGINT``VARCHAR`, e`TIMESTAMP`)`BOOLEAN`, você pode associar dados de diferentes tipos em um registro de várias medidas.

As seções a seguir descrevem alguns exemplos diferentes de como o atributo nome da medida pode ser usado com eficiência para agrupar diferentes tipos de dados na mesma tabela.

Sensores de IoT que relatam qualidade e valor

Suponha que você tenha um aplicativo monitorando dados de sensores de IoT. Cada sensor rastreia medidas diferentes, como temperatura e pressão. Além dos valores reais, os sensores também relatam a qualidade das medições, que é uma medida da precisão da leitura e uma unidade para a medição. Como qualidade, unidade e valor são emitidos juntos, eles podem ser modelados como registros de várias medidas, conforme mostrado nos dados de exemplo abaixo, onde device_id é uma dimensão, qualidade, valor e unidade são atributos de várias medidas:

id_dispositivo	nome_medida	Tempo	Qualidade	Valor	Unidade
sensor-sea478	temperatura	2021-12-01 19:22:32	92	35	c
sensor-sea478	temperatura	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

Essa abordagem permite combinar os benefícios dos registros de várias medidas com o particionamento e a remoção de dados usando os valores do nome da medida. Se as consultas fizerem referência a uma única medida, por exemplo, temperatura, você poderá incluir um predicado de nome de medida na consulta. Veja a seguir um exemplo dessa consulta, que também projeta a unidade para medições cuja qualidade esteja acima de 90.

```
SELECT device_id, time, value AS temperature, unit
FROM db.table
WHERE time > ago(1h)
    AND measure_name = 'temperature'
    AND quality > 90
```

Usar o predicado measure_name na consulta permite que o Timestream elimine efetivamente partições e dados que não são relevantes LiveAnalytics para a consulta, melhorando assim a latência da consulta.

Também é possível armazenar todas as métricas no mesmo registro de várias medidas se todas as métricas forem emitidas no mesmo timestamp e/ou várias métricas forem consultadas juntas na mesma consulta. Por exemplo, você pode construir um registro de várias medidas com os atributos temperatura_qualidade, valor_temperatura, unidade_temperatura, qualidade_pressão, valor_pressão, unidade_pressão etc. Muitos dos pontos discutidos anteriormente sobre a modelagem de dados usando registros de medida única versus registros de várias medidas se aplicam à sua decisão de como modelar os dados. Considere seus padrões de acesso à consulta e como seus dados são gerados para escolher um modelo que otimize seu custo, ingestão e latência de consultas e a facilidade de escrever suas consultas.

Diferentes tipos de métricas na mesma tabela

Outro caso de uso em que você pode combinar registros de várias medidas com valores de nomes de medidas é modelar diferentes tipos de dados que são emitidos de forma independente pelo mesmo dispositivo. Considere o caso de uso de DevOps monitoramento que os servidores estão emitindo dois tipos de dados: métricas emitidas regularmente e eventos irregulares. Um exemplo dessa abordagem é o esquema discutido no gerador de dados que modela um caso de DevOps uso. Nesse caso, você pode armazenar os diferentes tipos de dados emitidos pelo mesmo servidor na mesma tabela usando nomes de medidas diferentes. Por exemplo, todas as métricas emitidas no mesmo instante são armazenadas com métricas de nome de medida. Todos os eventos emitidos em um instante de tempo diferente das métricas são armazenados com eventos de nome de medida. O esquema de medidas para a tabela (por exemplo, saída da `SHOW MEASURES` consulta) é:

nome_medida	data_type	Dimensões
eventos	multi	[{"data_type” :"varchar ”, "nome_da-dimensão” :"zona_disponibilidade "}, {" data_type ” :"varchar”, "nome_da-dimensão” :"nome_do _microserviço "}, {" data_type” :"varchar”, "nome_dimensão” :"nome_instância "}, {" data_type ” :"varchar”, "nome_da-dimensão” :"nome_do _processo "}, {" data_type ” :"varchar”, "nome_da-dimensão” :"jdk_version "}, {" data_type” :"varchar”, "nome_dimensão” :"célula "}, {" data_type” :"varchar”, "nome_dimensão” :"região "}, {" data_type” :"data_região "}, {" data_type” :"data_região "} type” :"varchar”, "nome_da_dimensão” :"silo "}]
métricas	multi	[{"data_type” :"varchar ”, "nome_da-dimensão” :"zona_disponibilidade "}, {" data_type ” :"varchar”, "nome_da-dimensão” :"nome_do _microserviço "}, {" data_type” :"varchar”, "nome_dimensão” :"nome_instância "}, {" data_type

nome_medida	data_type	Dimensões
		” :"varchar”, "nome_da-dimensão” :"os_versão_"}, {"data_type” :"varchar”, "nome_dimensão” :"célula "}, {" data_type” :"varchar”, "nome_dimensão” :"região "}, {" data_type” :"varchar ”, "nome_dimensão” :"silo "}, {" data_type” :"tipo de dados” "varchar”, "nome_dimensão” :"tipo_instância "}]

Nesse caso, você pode ver que os eventos e as métricas também têm conjuntos diferentes de dimensões, em que os eventos têm dimensões diferentes jdk_version e process_name, enquanto as métricas têm dimensões instance_type e os_version.

O uso de nomes de medidas diferentes permite que você escreva consultas com predicados, de modo `WHERE measure_name = 'metrics'` a obter somente as métricas. Além disso, ter todos os dados emitidos da mesma instância na mesma tabela significa que você também pode escrever uma consulta mais simples com o predicado instance_name para obter todos os dados dessa instância. Por exemplo, um predicado do formulário `WHERE instance_name = ‘instance-1234’` sem um predicado measure_name retornará todos os dados de uma instância específica do servidor.

Recomendações para particionar registros de várias medidas

> **Important**
>
> Esta seção está obsoleta!
> Essas recomendações estão desatualizadas. Agora, o particionamento é melhor controlado usando chaves de partição definidas pelo cliente.

Vimos que há um número crescente de cargas de trabalho no ecossistema de séries temporais que precisam ingerir e armazenar grandes quantidades de dados e, ao mesmo tempo, precisam de

respostas de consulta de baixa latência ao acessar dados por meio de um conjunto de valores de dimensão de alta cardinalidade.

Devido a essas características, as recomendações desta seção serão úteis para cargas de trabalho de clientes que tenham o seguinte.

- Adotou ou deseja adotar registros de várias medidas.
- Espere ter um grande volume de dados entrando no sistema que serão armazenados por longos períodos.
- Exigem tempos de resposta de baixa latência para seus padrões principais de acesso (consulta).
- Saiba que os padrões de consultas mais importantes envolvem algum tipo de condição de filtragem no predicado. Essa condição de filtragem é baseada em uma dimensão de alta cardinalidade. Por exemplo, considere eventos ou agregações por UserId,, ServerID DeviceId, nome do host e assim por diante.

Nesses casos, um único nome para todas as medidas de várias medidas não ajudará, pois nosso mecanismo usa um nome de várias medidas para particionar os dados e ter um único valor limita a vantagem de partição que você obtém. O particionamento desses registros é baseado principalmente em duas dimensões. Digamos que o tempo esteja no eixo x e em um hash dos nomes das dimensões e `measure_name` no eixo y. Nesses `measure_name` casos, funciona quase como uma chave de particionamento.

Nossa recomendação é a seguinte.

- Ao modelar seus dados para casos de uso como o que mencionamos, use um `measure_name` que seja um derivado direto do seu padrão de acesso à consulta principal. Por exemplo:
  - Seu caso de uso exige monitorar o desempenho do aplicativo e a QoE do ponto de vista do usuário final. Isso também pode ser medições de rastreamento para um único servidor ou dispositivo de IoT.
  - Se você estiver consultando e filtrando por UserId, precisará, no momento da ingestão, encontrar a melhor maneira de se associar a. `measure_name` UserId
  - Como uma tabela de várias medidas só pode conter 8192 nomes de medidas diferentes, qualquer fórmula adotada não deve gerar mais do que 8192 valores diferentes.
- Uma abordagem que aplicamos com sucesso para valores de string é aplicar um algoritmo de hash ao valor da string. Em seguida, execute a operação do módulo com o valor absoluto do resultado do hash e 8192.

```
measure_name = getMeasureName(UserId)
int getMeasureName(value) {
    hash_value =  abs(hash(value))
    return hash_value % 8192
}
```

- Também adicionamos `abs()` para remover o sinal, eliminando a possibilidade de os valores variarem de -8192 a 8192. Isso deve ser executado antes da operação do módulo.
- Ao usar esse método, suas consultas podem ser executadas em uma fração do tempo necessário para serem executadas em um modelo de dados não particionado.
- Ao consultar os dados, certifique-se de incluir uma condição de filtragem no predicado que usa o valor recém-derivado do nome_medida. Por exemplo:
  - ```
    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))
    ```
  - Isso minimizará o número total de partições digitalizadas para obter dados que se traduzirão em consultas mais rápidas ao longo do tempo.

Lembre-se de que, se você quiser obter os benefícios desse esquema de partição, o hash precisa ser calculado no lado do cliente e passado para o Timestream LiveAnalytics como um valor estático para o mecanismo de consulta. O exemplo anterior fornece uma forma de validar se o hash gerado pode ser resolvido pelo mecanismo quando necessário.

horário	host_name	local	tipo_servidor	cpu_usage	memória_disponível	cpu_temp
2022-09-07 21:48:44 .000 0	host-1235	leste dos EUA1	5,8xl	55	16.2	78
R2022-09-07	host-3587	oeste dos EUA 1	5,8xl	62	18.1	81
    ```

horário	host_name	local	tipo_servidor	cpu_usage	memória_disponível	cpu_temp
21:48:44 .000 0						
2022-09-07 21:48:45.000 000000	host-258743	eu-central	5,8xl	88	9.4	91
2022-09-07 21:48:45 .000 0	host-35654	leste dos EUA 2	5,8xl	29	24	54
R2022-09-07 21:48:45 .000 0	host-254	oeste dos EUA 1	5,8xl	44	32	48

Para gerar o associado `measure_name` seguindo nossa recomendação, há dois caminhos que dependem do seu padrão de ingestão.

1. Para ingestão em lote de dados históricos — você pode adicionar a transformação ao seu código de gravação se usar seu próprio código para o processo em lote.

   Construindo com base no exemplo anterior.

```
        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);
```

```
        }
```

Saída

```
host-1235 - 6445
host-3587 - 6399
host-258743 - 640
host-35654 - 2093
host-254 - 7051
```

Conjunto de dados resultante

horário	host_name	local	nome_medida	tipo_servidor	cpu_usage	memória_disponível	cpu_temp
2022-09-07 21:48:44 .00	host-1235	leste dos EUA1	6445	5,8xl	55	16.2	78
R2022-09-07 21:48:44 .00	host-3587	oeste dos EUA 1	6399	5,8xl	62	18.1	81
2022-09-07 21:48:45.000 000000	host-258743	eu-central	640	5,8xl	88	9.4	91
2022-09-07 21:48:45 .00	host-35654	leste dos EUA 2	2093	5,8xl	29	24	54

horário	host_name	local	nome_med da	tipo_serv idor	cpu_usage	memória_d isponível	cpu_temp
R2022-09-07 21:48:45 .0 0	host-254	oeste dos EUA 1	7051	5,8xl	44	32	48

2. Para ingestão em tempo real — você precisa gerar a transmissão à `measure_name` medida que os dados chegam.

Em ambos os casos, recomendamos que você teste seu algoritmo de geração de hash nas duas extremidades (ingestão e consulta) para garantir que você esteja obtendo os mesmos resultados.

Aqui estão alguns exemplos de código para gerar o valor em hash com base em`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 dependência no Maven

```
        <dependency>
            <groupId>net.jpountz.lz4</groupId>
            <artifactId>lz4</artifactId>
            <version>1.3.0</version>
        </dependency>
```

Segurança

- Para acesso contínuo ao Timestream for LiveAnalytics, certifique-se de que as chaves de criptografia estejam protegidas e não sejam revogadas ou tornadas inacessíveis.

- Monitore os registros de API acesso de AWS CloudTrail. Audite e revogue qualquer padrão de acesso anômalo de usuários não autorizados.
- Siga as diretrizes adicionais descritas emMelhores práticas de segurança para o Amazon Timestream for LiveAnalytics.

Configurando o Amazon Timestream para LiveAnalytics

Configure o período de retenção de dados para o armazenamento de memória e o armazenamento magnético de acordo com os requisitos de processamento, armazenamento, desempenho de consultas e custo de dados.

- Defina a retenção de dados do armazenamento de memória de acordo com os requisitos do seu aplicativo para processar dados que chegam tarde. Os dados de chegada tardia são dados recebidos com um carimbo de data/hora anterior à hora atual. É emitido de recursos que agrupam eventos por um período de tempo antes de enviar os dados para o Timestream e também de recursos com conectividade intermitente LiveAnalytics, por exemplo, um sensor de IoT que está on-line de forma intermitente.
- Se você espera que os dados de chegada tardia cheguem ocasionalmente com registros de data e hora antes da retenção do armazenamento de memória, você deve habilitar as gravações do armazenamento magnético para sua tabela. Depois de configurar uma tabela, a EnableMagneticStoreWrites tabela aceitará dados com registro de data e hora antes da retenção do armazenamento na memória, mas dentro do período de retenção do armazenamento magnético. MagneticStoreWritesProperties
- Considere as características das consultas que você planeja executar no Timestream, LiveAnalytics como os tipos de consultas, a frequência, o intervalo de tempo e os requisitos de desempenho. Isso ocorre porque o armazenamento de memória e o armazenamento magnético são otimizados para diferentes cenários. O armazenamento de memória é otimizado para point-in-time consultas rápidas que processam pequenas quantidades de dados recentes enviados ao Timestream for. LiveAnalytics O armazenamento magnético é otimizado para consultas analíticas rápidas que processam volumes médios a grandes de dados enviados ao Timestream for. LiveAnalytics
- Seu período de retenção de dados também deve ser influenciado pelos requisitos de custo do seu sistema.

  Por exemplo, considere um cenário em que o limite de dados de chegada tardia para seu aplicativo é de 2 horas e seus aplicativos enviam muitas consultas que processam dados de um

dia, uma semana ou um mês. Nesse caso, talvez você queira configurar um período de retenção menor para o armazenamento de memória (2 a 3 horas) e permitir que mais dados fluam para o armazenamento magnético, já que o armazenamento magnético é otimizado para consultas analíticas rápidas.

Entenda o impacto de aumentar ou diminuir o período de retenção de dados do armazenamento de memória e do armazenamento magnético de uma tabela existente.

- Quando você diminui o período de retenção do armazenamento de memória, os dados são movidos do armazenamento de memória para o armazenamento magnético, e essa transferência de dados é permanente. O Timestream for LiveAnalytics não recupera dados do armazenamento magnético para preencher o armazenamento de memória. Quando você diminui o período de retenção do armazenamento magnético, os dados são excluídos do sistema e a exclusão dos dados é permanente.
- Quando você aumenta o período de retenção do armazenamento de memória ou do armazenamento magnético, a alteração entra em vigor para os dados enviados ao Timestream a LiveAnalytics partir desse ponto. O Timestream for LiveAnalytics não recupera dados do armazenamento magnético para preencher o armazenamento de memória. Por exemplo, se o período de retenção do armazenamento de memória foi inicialmente definido para 2 horas e depois aumentado para 24 horas, serão necessárias 22 horas para que o armazenamento de memória contenha 24 horas de dados.

Escreve

- Certifique-se de que o registro de data e hora dos dados recebidos não seja anterior à retenção de dados configurada para o armazenamento de memória e não posterior ao período de ingestão futuro definido em. Cotas Enviar dados com um timestamp fora desses limites resultará na rejeição dos dados pelo Timestream, a LiveAnalytics menos que você habilite gravações de armazenamento magnético para sua tabela. Se você habilitar gravações no armazenamento magnético, certifique-se de que o registro de data e hora dos dados recebidos não seja anterior à retenção de dados configurada para o armazenamento magnético.
- Se você espera que os dados cheguem tarde, ative as gravações do armazenamento magnético em sua tabela. Isso permitirá a ingestão de dados com registros de data e hora que estejam fora do período de retenção do armazenamento na memória, mas ainda dentro do período de retenção do armazenamento magnético. Você pode definir isso atualizando a `EnableMagneticStoreWrites` bandeira em `MagneticStoreWritesProperties` para sua

tabela. Essa propriedade é falsa por padrão. Observe que as gravações no armazenamento magnético não estarão imediatamente disponíveis para consulta. Eles estarão disponíveis em 6 horas.

- Direcione cargas de trabalho de alta taxa de transferência para o armazenamento de memória, garantindo que os registros de data e hora dos dados ingeridos estejam dentro dos limites de retenção do armazenamento de memória. As gravações no armazenamento magnético são limitadas a um número máximo de partições ativas do armazenamento magnético que podem receber ingestão simultânea para um banco de dados. Você pode ver essa `ActiveMagneticStorePartitions` métrica em CloudWatch. Para reduzir as partições ativas do armazenamento magnético, tente reduzir o número de séries e a duração do tempo que você ingere simultaneamente para a ingestão do armazenamento magnético.
- Ao enviar dados para o Timestream for LiveAnalytics, agrupe vários registros em uma única solicitação para otimizar o desempenho da ingestão de dados.
  - É vantajoso agrupar registros da mesma série temporal e registros com o mesmo nome de medida.
  - Agrupe o maior número possível de registros em uma única solicitação, desde que as solicitações estejam dentro dos limites de serviço definidos emCotas.
  - Use atributos comuns sempre que possível para reduzir os custos de transferência e ingestão de dados. Para obter mais informações, consulte WriteRecords API.
- Se você encontrar falhas parciais do lado do cliente ao gravar dados no Timestream for LiveAnalytics, poderá reenviar o lote de registros que falharam na ingestão depois de resolver a causa da rejeição.
- Os dados ordenados por timestamps têm melhor desempenho de gravação.
- O Amazon Timestream LiveAnalytics for foi projetado para ser escalado automaticamente de acordo com as necessidades do seu aplicativo. Quando o Timestream for LiveAnalytics notices aumenta nas solicitações de gravação do seu aplicativo, seu aplicativo pode experimentar algum nível de limitação inicial do armazenamento de memória. Se seu aplicativo sofrer limitação do armazenamento de memória, continue enviando dados para o Timestream LiveAnalytics na mesma taxa (ou maior) para permitir que o Timestream seja escalado automaticamente para atender LiveAnalytics às necessidades do seu aplicativo. Se você observar uma limitação do armazenamento magnético, diminua a taxa de ingestão do armazenamento magnético até o número de quedas. `ActiveMagneticStorePartitions`

Carregamento em lote

As melhores práticas para carregamento em lote estão descritas emPráticas recomendadas de carregamento em lote.

Consultas

A seguir estão as melhores práticas sugeridas para consultas com o Amazon Timestream for. LiveAnalytics

- Inclua somente os nomes de medidas e dimensões essenciais para a consulta. Adicionar colunas estranhas aumentará as varreduras de dados, o que afeta o desempenho das consultas.
- Antes de implantar sua consulta na produção, recomendamos que você analise os insights da consulta para garantir que a redução espacial e temporal seja ideal. Para obter mais informações, consulte Usando insights de consulta para otimizar consultas no Amazon Timestream.
- Sempre que possível, envie a computação de dados para o Timestream para LiveAnalytics usar os agregados integrados e as funções escalares na cláusula e na SELECT cláusula, conforme aplicável, para melhorar o desempenho da WHERE consulta e reduzir os custos. Consulte SELECT e Funções agregadas.
- Sempre que possível, use funções aproximadas. Por exemplo, use APPROX _ DISTINCT em vez de COUNT (DISTINCTcolumn_name) para otimizar o desempenho da consulta e reduzir o custo da consulta. Consulte Funções agregadas.
- Use uma CASE expressão para realizar agregações complexas em vez de selecionar na mesma tabela várias vezes. Consulte A CASE declaração.
- Sempre que possível, inclua um intervalo de tempo na WHERE cláusula da sua consulta. Isso otimiza o desempenho e os custos das consultas. Por exemplo, se você precisar apenas da última hora de dados em seu conjunto de dados, inclua um predicado de tempo, como hora > atrás (1h). Consulte SELECT e Intervalo e duração.
- Quando uma consulta acessa um subconjunto de medidas em uma tabela, sempre inclua os nomes das medidas na WHERE cláusula da consulta.
- Sempre que possível, use o operador de igualdade ao comparar dimensões e medidas na WHERE cláusula de uma consulta. Um predicado de igualdade nas dimensões e nos nomes das medidas permite melhorar o desempenho da consulta e reduzir os custos da consulta.
- Sempre que possível, evite usar funções na WHERE cláusula para otimizar o custo.
- Evite usar a LIKE cláusula várias vezes. Em vez disso, use expressões regulares ao filtrar vários valores em uma coluna de string. Consulte Funções de expressões regulares.

- Use somente as colunas necessárias na cláusula GROUP BY de uma consulta.
- Se o resultado da consulta precisar estar em uma ordem específica, especifique explicitamente essa ordem na cláusula ORDER BY da consulta mais externa. Se o resultado da consulta não exigir ordenação, evite usar uma cláusula ORDER BY para melhorar o desempenho da consulta.
- Use uma LIMIT cláusula se você precisar apenas das primeiras N linhas em sua consulta.
- Se você estiver usando uma ORDER cláusula BY para examinar os valores N superiores ou inferiores, use uma LIMIT cláusula para reduzir os custos da consulta.
- Use o token de paginação da resposta retornada para recuperar os resultados da consulta. Para obter mais informações, consulte Consulta.
- Se você começou a executar uma consulta e percebeu que a consulta não retornará os resultados que você está procurando, cancele a consulta para economizar custos. Para obter mais informações, consulte CancelQuery.
- Se seu aplicativo sofrer limitação, continue enviando dados para o Amazon Timestream na mesma taxa LiveAnalytics para permitir que o Amazon Timestream for escale automaticamente para satisfazer as necessidades de taxa de transferência de LiveAnalytics consultas do seu aplicativo.
- Se os requisitos de simultaneidade de consultas de seus aplicativos excederem os limites padrão do Timestream for LiveAnalytics, entre em contato AWS Support para saber se o limite aumenta.

Consultas programadas

As consultas agendadas ajudam você a otimizar seus painéis pré-computando algumas estatísticas agregadas de toda a frota. Portanto, uma pergunta natural a ser feita é como você usa seu caso de uso e identifica quais resultados devem ser pré-computados e como usar esses resultados armazenados em uma tabela derivada para criar seu painel. A primeira etapa desse processo é identificar quais painéis devem ser pré-computados. Abaixo estão algumas diretrizes de alto nível:

- Considere os bytes verificados pelas consultas usadas para preencher os painéis, a frequência de recarga do painel e o número de usuários simultâneos que carregariam esses painéis. Você deve começar com os painéis carregados com mais frequência e digitalizar quantidades significativas de dados. Os dois primeiros painéis no exemplo do painel agregado, bem como o painel agregado no exemplo detalhado, são bons exemplos desses painéis.
- Considere quais cálculos estão sendo usados repetidamente. Embora seja possível criar uma consulta agendada para cada painel e cada valor de variável usado no painel, você pode otimizar significativamente seus custos e o número de consultas agendadas procurando maneiras de usar um único cálculo para pré-computar os dados necessários para vários painéis.

- Considere a frequência de suas consultas agendadas para atualizar os resultados materializados na tabela derivada. Você gostaria de analisar a frequência com que um painel é atualizado versus a janela de tempo que é consultada em um painel versus o intervalo de tempo usado na pré-computação, bem como nos painéis nos painéis. Por exemplo, se um painel que está plotando agregados de hora em hora nos últimos dias for atualizado apenas uma vez em algumas horas, convém configurar suas consultas agendadas para serem atualizadas apenas uma vez a cada 30 minutos ou uma hora. Por outro lado, se você tiver um painel que traça agregados por minuto e é atualizado a cada minuto, convém que suas consultas agendadas atualizem os resultados a cada minuto ou alguns minutos.
- Considere quais padrões de consulta podem ser ainda mais otimizados (tanto do ponto de vista do custo quanto da latência da consulta) usando consultas agendadas. Por exemplo, ao calcular os valores de dimensão exclusivos frequentemente usados como variáveis em painéis ou ao retornar o último ponto de dados emitido por um sensor ou o primeiro ponto de dados emitido por um sensor após uma determinada data, etc. Alguns desses exemplos de padrões são discutidos neste guia.

As considerações anteriores terão um impacto significativo na economia quando você mover o painel para consultar as tabelas derivadas, na atualização dos dados nos painéis e no custo incorrido pelas consultas agendadas.

Aplicativos cliente e integrações suportadas

Execute seu aplicativo cliente na mesma região do Timestream LiveAnalytics para reduzir as latências de rede e os custos de transferência de dados. Para obter mais informações sobre como trabalhar com outros serviços, consulteComo trabalhar com outros serviços do . A seguir estão alguns outros links úteis.

- Melhores práticas para AWS desenvolvimento com o AWS SDK for Java
- Práticas recomendadas para trabalhar com AWS Lambda funções
- Melhores práticas do Amazon Managed Service para Apache Flink
- Melhores práticas para criar painéis no Grafana

Geral

- Certifique-se de seguir o The AWS Well-Architected Framework ao usar o Timestream para. LiveAnalytics Este whitepaper fornece orientação sobre as melhores práticas em excelência operacional, segurança, confiabilidade, eficiência de desempenho e otimização de custos.

Medição e otimização de custos

Com o Amazon Timestream LiveAnalytics for, você paga somente pelo que usa. Timestream separadamente para LiveAnalytics medidores para gravações, dados armazenados e dados digitalizados por consultas. O preço de cada dimensão de medição é especificado na página de preços. Você pode estimar sua fatura mensal usando a calculadora de preços do Amazon Timestream LiveAnalytics for Pricing.

Esta seção descreve como a medição funciona para gravações, armazenamento e consultas no Timestream for. LiveAnalytics Também são fornecidos exemplos de cenários e cálculos. Além disso, uma lista das melhores práticas para otimização de custos está incluída. Você pode selecionar um tópico abaixo:

Tópicos

- Escreve
- Armazenamento
- Consultas
- Otimização de custo
- Monitoramento com a Amazon CloudWatch

Escreve

O tamanho de gravação de cada evento de série temporal é calculado como a soma do tamanho do timestamp e um ou mais nomes de dimensão, valores de dimensão, nomes de medidas e valores de medida. O tamanho do timestamp é de 8 bytes. O tamanho dos nomes das dimensões, dos valores das dimensões e dos nomes das medidas são o comprimento dos UTF -8 bytes codificados da string que representa cada nome de dimensão, valor de dimensão e nome de medida. O tamanho do valor da medida depende do tipo de dados. É 1 byte para o tipo de dados booleano, 8 bytes para bigint e double e o comprimento dos UTF -8 bytes codificados para strings. Cada gravação é contada em unidades de 1 KiB.

Dois exemplos de cálculos são fornecidos abaixo:

Tópicos

- Cálculo do tamanho de gravação de um evento de série temporal
- Calculando o número de gravações

Cálculo do tamanho de gravação de um evento de série temporal

Considere um evento de série temporal representando a CPU utilização de uma EC2 instância, conforme mostrado abaixo:

Tempo	região	az	vpc	Hostname	nome_medida	valor_medida::duplo
1602983435238563000	us-east-1	1d	vpc-1a2b3c4d	Anfitrião-24GJU	utilização_da CPU	35,0

O tamanho de gravação do evento da série temporal pode ser calculado como:

- tempo = 8 bytes
- primeira dimensão = 15 bytes (`region+us-east-1`)
- segunda dimensão = 4 bytes (`az+1d`)
- terceira dimensão = 15 bytes (`vpc+vpc-1a2b3c4d`)
- quarta dimensão = 18 bytes (`hostname+host-24Gju`)
- nome da medida = 15 bytes (`cpu_utilization`)
- valor da medida = 8 bytes

Tamanho de gravação do evento da série temporal = 83 bytes

Calculando o número de gravações

Agora, considere 100 EC2 instâncias, semelhantes à instância descrita emCálculo do tamanho de gravação de um evento de série temporal, emitindo métricas a cada 5 segundos. O total de gravações mensais das EC2 instâncias variará com base em quantos eventos de séries temporais

existem por gravação e se atributos comuns estão sendo usados durante o agrupamento de eventos de séries temporais. Um exemplo de cálculo do total de gravações mensais é fornecido para cada um dos seguintes cenários:

Tópicos

- Um evento de série temporal por gravação
- Agrupando eventos de séries temporais em uma gravação
- Agrupando eventos de séries temporais e usando atributos comuns em uma gravação

Um evento de série temporal por gravação

Se cada gravação contiver somente um evento de série temporal, o total de gravações mensais será calculado como:

- 100 eventos de séries temporais = 100 gravações a cada 5 segundos
- x 12 gravações/minuto = 1.200 gravações
- x 60 minutos/hora = 72.000 gravações
- x 24 horas/dia = 1.728.000 gravações
- x 30 dias/mês = 51.840.000 gravações

Total de gravações mensais = 51.840.000

Agrupando eventos de séries temporais em uma gravação

Como cada gravação é medida em unidades de 1 KB, uma gravação pode conter um lote de 12 eventos de série temporal (998 bytes) e o total de gravações mensais é calculado como:

- 100 eventos de séries temporais = 9 gravações (12 eventos de séries temporais por gravação) a cada 5 segundos
- x 12 gravações/minuto = 108 gravações
- x 60 minutos/hora = 6.480 gravações
- x 24 horas/dia = 155.520 gravações
- x 30 dias/mês = 4.665.600 gravações

Total de gravações mensais = 4.665.600

Agrupando eventos de séries temporais e usando atributos comuns em uma gravação

Se a região, az, vpc e nome da medida forem comuns em 100 EC2 instâncias, os valores comuns poderão ser especificados apenas uma vez por gravação e serão chamados de atributos comuns. Nesse caso, o tamanho dos atributos comuns é 52 bytes e o tamanho dos eventos da série temporal é 27 bytes. Como cada gravação é medida em unidades de 1 KiB, uma gravação pode conter 36 eventos de séries temporais e atributos comuns, e o total de gravações mensais é calculado da seguinte forma:

- 100 eventos de séries temporais = 3 gravações (36 eventos de séries temporais por gravação) a cada 5 segundos
- x 12 gravações/minuto = 36 gravações
- x 60 minutos/hora = 2.160 gravações
- x 24 horas/dia = 51.840 gravações
- x 30 dias/mês = 1.555.200 gravações

Total de gravações mensais = 1.555.200

> **Note**
>
> Devido ao uso de lotes, atributos comuns e arredondamento das gravações para unidades de 1 KB, o tamanho de armazenamento dos eventos da série temporal pode ser diferente do tamanho da gravação.

Armazenamento

O tamanho de armazenamento de cada evento de série temporal no armazenamento de memória e no armazenamento magnético é calculado como a soma do tamanho do timestamp, nomes de dimensões, valores de dimensão, nomes de medidas e valores de medida. O tamanho do timestamp é de 8 bytes. O tamanho dos nomes das dimensões, dos valores das dimensões e dos nomes das medidas são o comprimento dos UTF -8 bytes codificados de cada string representando o nome da dimensão, o valor da dimensão e o nome da medida. O tamanho do valor da medida depende do tipo de dados. É 1 byte para tipos de dados booleanos, 8 bytes para bigint e double e o comprimento dos UTF -8 bytes codificados para strings. Cada medida é armazenada como um registro separado no Amazon Timestream LiveAnalytics para, ou seja, se seu evento de série temporal tiver quatro medidas, haverá quatro registros desse evento de série temporal armazenados.

Considerando o exemplo do evento de série temporal que representa a CPU utilização de uma EC2 instância (consulteCálculo do tamanho de gravação de um evento de série temporal), o tamanho de armazenamento do evento de série temporal é calculado como:

- tempo = 8 bytes
- primeira dimensão = 15 bytes (`region+us-east-1`)
- segunda dimensão = 4 bytes (`az+1d`)
- terceira dimensão = 15 bytes (`vpc+vpc-1a2b3c4d`)
- quarta dimensão = 18 bytes (`hostname+host-24Gju`)
- nome da medida = 15 bytes (`cpu_utilization`)
- valor da medida = 8 bytes

Tamanho de armazenamento do evento da série temporal = 83 bytes

> **Note**
>
> O armazenamento de memória é medido em GB-hora e o armazenamento magnético é medido em GB-mês.

Consultas

As consultas são cobradas com base na duração das unidades computacionais do Timestream (TCUs) usadas pelo seu aplicativo em TCU -horas, conforme especificado na página de preços do Amazon Timestream. O Amazon Timestream LiveAnalytics for 'query engine remove dados irrelevantes ao processar uma consulta. Consultas com projeções e predicados, incluindo intervalos de tempo, nomes de medidas e/ou nomes de dimensões, permitem que o mecanismo de processamento de consultas elimine uma quantidade significativa de dados e ajude a reduzir os custos de consulta.

Otimização de custo

Para otimizar o custo de gravações, armazenamento e consultas, use as seguintes melhores práticas com o Amazon Timestream para: LiveAnalytics

- Batch em lote vários eventos de séries temporais por gravação para reduzir o número de solicitações de gravação.

- Considere o uso de registros de várias medidas, que permitem gravar várias medidas de séries temporais em uma única solicitação de gravação e armazenar seus dados de maneira mais compacta. Isso reduz o número de solicitações de gravação, bem como o custo de armazenamento de dados e o custo de consulta.
- Use atributos comuns com lotes para agrupar mais eventos de séries temporais por gravação para reduzir ainda mais o número de solicitações de gravação.
- Defina a retenção de dados do armazenamento de memória de acordo com os requisitos do seu aplicativo para processar dados que chegam tardiamente. Os dados de chegada tardia são dados recebidos com um registro de data e hora anterior à hora atual e fora do período de retenção do armazenamento de memória.
- Defina a retenção de dados do armazenamento magnético de acordo com seus requisitos de armazenamento de dados de longo prazo.
- Ao escrever consultas, inclua somente os nomes de medidas e dimensões essenciais para a consulta. Adicionar colunas estranhas aumentará as varreduras de dados e, portanto, também aumentará o custo da consulta. Recomendamos que você analise os insights da consulta para avaliar a eficiência de poda das dimensões e medidas incluídas.
- Sempre que possível, inclua um intervalo de tempo na WHERE cláusula da sua consulta. Por exemplo, se você precisar apenas da última hora de dados em seu conjunto de dados, inclua um predicado de tempo como. `time > ago(1h)`
- Quando uma consulta acessa um subconjunto de medidas em uma tabela, sempre inclua os nomes das medidas na WHERE cláusula da consulta.
- Se você começou a executar uma consulta e percebeu que a consulta não retornará os resultados que você está procurando, cancele a consulta para economizar custos.

Monitoramento com a Amazon CloudWatch

Você pode monitorar o Timestream para usar a LiveAnalytics Amazon CloudWatch, que coleta e processa dados brutos do Timestream em métricas legíveis. LiveAnalytics near-real-time Essas estatísticas são mantidas por duas semanas, de maneira que você possa acessar informações históricas e ter uma perspectiva melhor do desempenho do aplicativo web ou do serviço. Por padrão, o Timestream para dados LiveAnalytics métricos é enviado automaticamente CloudWatch em períodos de 1 ou 15 minutos. Para obter mais informações, consulte O que é a Amazon CloudWatch? no Guia do CloudWatch usuário da Amazon.

Tópicos

- Como faço para usar o Timestream para LiveAnalytics métricas?
- Cronograma para LiveAnalytics métricas e dimensões
- Criação de CloudWatch alarmes para monitorar o Timestream para LiveAnalytics

Como faço para usar o Timestream para LiveAnalytics métricas?

As métricas relatadas pelo Timestream LiveAnalytics fornecem informações que você pode analisar de maneiras diferentes. A lista a seguir mostra alguns usos comuns para as métricas. Essas são sugestões para você começar, e não uma lista abrangente.

Como?	Métricas relevantes
`How can I determine if any system errors occurred?`	Você pode monitorar `SystemErrors` para saber se todas as solicitações resultaram em um código de erro de servidor. Normalmente, essa métrica deve ser igual a zero. Se não for o caso, você deve investigar.
`How can I monitor the amount of data in the memory store?`	Você pode monitorar `MemoryCumulativeBytesMetered` durante o período de tempo especificado para monitorar a quantidade de dados armazenados na memória armazenada em bytes. Essa métrica é emitida a cada hora e você pode rastrear os bytes armazenados em uma conta, bem como a granularidade do banco de dados. O armazenamento de memória é medido em GB/hora (o custo de armazenar 1 GB de dados por uma hora). Portanto, multiplicar o valor por hora pelo preço em GB/hora em sua região resultará no custo incorrido por hora. `MemoryCumulativeBytesMetered` Dimensões: Operação (armazenamento) DatabaseName, Nome da métrica
`How can I monitor the amount of data in the magnetic store?`	Você pode monitorar `MagneticCumulativeBytesMetered` durante o período de tempo especificado para monitorar a quantidade de dados armazenados no armazenamento magnético em bytes. Essa métrica é emitida a cada hora e você pode rastrear os bytes armazenados em uma conta, bem como a granularidade do banco de dados. O armazenamento

Como?	Métricas relevantes
	de memória é medido em GB por mês (o custo de armazenar 1 GB de dados por um mês). Portanto, multiplicar o valor por hora pelo preço em GB por mês em sua região resultará no custo incorrido por hora. `MagneticCumulativeBytesMetered` Por exemplo, se o valor de `MagneticCumulativeBytesMetered` for 107374182400 bytes (100 GB), a cobrança horária de 1 GB de dados no armazenamento magnético = (0,03) (preço us-east-1)/(30,4*24). Multiplicar esse valor pelo `MagneticCumulativeBytesMetered` em GB resultará em ~ $0,004 por essa hora. Dimensões: Operação (Armazenamento) DatabaseName, Nome da métrica
`How can I monitor the data scanned by queries?`	Você pode monitorar `CumulativeBytesMetered` durante o período de tempo especificado para monitorar os dados digitalizados pelas consultas (em bytes) enviadas ao Timestream for. LiveAnalytics Essa métrica é emitida após a execução da consulta e você pode rastrear os dados digitalizados na granularidade da conta e do banco de dados. Você pode calcular o custo da consulta para um determinado período multiplicando o valor da métrica pelo preço escaneado por GB em sua região. Os bytes verificados por consultas agendadas são contabilizados nessa métrica. Dimensões: Operação (Consulta) DatabaseName, Nome da métrica

Como?	Métricas relevantes
`How can I monitor the data scanned by scheduled queries?`	Você pode monitorar `CumulativeBytesMetered` durante o período de tempo especificado para monitorar os dados digitalizados por consultas agendadas (em bytes) executadas pelo Timestream for. LiveAnalytics Essa métrica é emitida após a execução da consulta e você pode rastrear os dados digitalizados na granularidade da conta e do banco de dados. Você pode calcular o custo da consulta para um determinado período multiplicando o valor da métrica pelo preço escaneado por GB em sua região. **Note** Os bytes medidos também são contabilizados na consulta. `CumulativeBytesMetered` Dimensões: Operação (TriggeredScheduledQuery), DatabaseName, Nome da métrica

Como?	Métricas relevantes
How can I monitor the number of records ingested?	Você pode monitorar `NumberOfRecords` durante o período especificado para monitorar o número de registros ingeridos. Você pode rastrear os bytes armazenados em uma conta, bem como a granularidade do banco de dados. Você também pode usar essa métrica para monitorar as gravações feitas pelas consultas agendadas quando os resultados da consulta são gravados em uma tabela separada. Ao usar o `WriteRecords` API, a métrica é emitida para cada `WriteRecords` solicitação, com a dimensão CloudWatch Operação sendo`WriteRecords` . Ao usar o `BatchLoad` ou `ScheduledQuery` APIs, a métrica é emitida em intervalo s determinados pelo serviço até que a tarefa seja concluída . A dimensão de CloudWatch operação para essa métrica é `BatchLoad` ou`ScheduledQuery` , dependendo de qual API é usada. Dimensões: Operação (WriteRecords, BatchLoad, ou Scheduled Query), DatabaseName, Nome da métrica

Como?	Métricas relevantes
How can I monitor the cost of records ingested?	Você pode monitorar CumulativeBytesMetered para monitorar o número de bytes ingeridos que geram custos. Você pode rastrear os bytes armazenados em uma conta, bem como a granularidade do banco de dados. Os registros ingeridos são medidos em bytes cumulativos. Multiplicar o valor do preço CumulativeBytesMetered by Writes em sua região fornece o custo de ingestão incorrido. Ao usar o WriteRecords API, essa métrica é emitida para cada WriteRecords solicitação, com a dimensão CloudWatch Operação sendoWriteRecords . Ao usar o BatchLoad ou ScheduledQuery API, a métrica é emitida em intervalos determinados pelo serviço até que a tarefa seja concluída . A dimensão de CloudWatch operação dessa métrica é BatchLoad ou ScheduledQuery depende de qual API é usada. Dimensões: Operação (WriteRecords, BatchLoad, ou Scheduled Query), DatabaseName, Nome da métrica
How can I monitor the Timestream Compute Units (TCUs) used in my account?	Você pode monitorar QueryTCU durante o período especificado para monitorar as unidades computacionais consumidas para a carga de trabalho de consulta na conta. Essa métrica é emitida com unidades computacionais máximas e mínimas para cada minuto durante a carga de trabalho de consulta ativa da conta. Unidades: Count Estatísticas válidas: mínimo, máximo Métrica: ResourceCount Dimensões:Service: Timestream ,Resource: QueryTCU,Type: Resource, Class: OnDemand

Cronograma para LiveAnalytics métricas e dimensões

Quando você interage com o Timestream for LiveAnalytics, ele envia as seguintes métricas e dimensões para a Amazon. CloudWatch Todas as métricas são agregadas e relatadas a cada minuto. Você pode usar os procedimentos a seguir para visualizar as métricas do Timestream for. LiveAnalytics

Para visualizar métricas usando o CloudWatch console

As métricas são agrupadas primeiro pelo namespace do serviço e, em seguida, por várias combinações de dimensão dentro de cada namespace.

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. Se necessário, altere a região da . Na barra de navegação, escolha a região em que seus AWS recursos residem. Para obter mais informações, consulte Endpoints de serviço do AWS.
3. No painel de navegação, selecione Métricas.
4. Na guia Todas as métricas, selecione `AWS/Timestream for LiveAnalytics.`.

Para visualizar métricas usando o AWS CLI

- Em um prompt de comando, use o seguinte comando.

```
aws cloudwatch list-metrics --namespace "AWS/Timestream"
```

Dimensões do Timestream para métricas LiveAnalytics

As métricas do Timestream for LiveAnalytics são qualificadas pelos valores da conta, nome da tabela ou operação. Você pode usar o CloudWatch console para recuperar o Timestream para LiveAnalytics dados em qualquer uma das dimensões na tabela a seguir:

Dimensão	Descrição
`DatabaseName`	Essa dimensão limita os dados a um Timestream específico para LiveAnalytics o banco de dados. Esse valor pode ser qualquer banco de dados na região atual e na AWS conta atual

Dimensão	Descrição
`Operation`	Essa dimensão limita os dados a um dos Timestream para LiveAnalytics operações, como`Storage`, `WriteRecords` `BatchLoad` , ou. `ScheduledQuery` Consulte o Timestream for LiveAnalytics Query API Reference para obter uma lista dos valores disponíveis.
`TableName`	Essa dimensão limita os dados a uma tabela específica em um Timestream para LiveAnalyticss banco de dados.

Important

`CumulativeBytesMetered`, `UserErrors` e `SystemErrors` as métricas só têm a `Operation` dimensão. `SuccessfulRequestLatency`as métricas sempre têm `Operation` dimensão, mas também podem ter as `TableName` dimensões `DatabaseName` e, dependendo do valor de`Operation`. Isso ocorre porque o Timestream para operações em LiveAnalytics nível de tabela tem `DatabaseName` e `TableName` como dimensões, mas as operações em nível de conta não.

Cronograma para métricas LiveAnalytics

Note

A Amazon CloudWatch agrega todo o Timestream a seguir para LiveAnalytics métricas em intervalos de um minuto.

Métricas gerais

Métrica	Descrição
`SuccessfulRequestLatency`	As solicitações bem-sucedidas para o Timestream LiveAnalytics durante o período de tempo especificado. SuccessfulRequestL

Métrica	Descrição
	atency pode fornecer dois tipos diferentes de informações: • O tempo decorrido para solicitações bem-sucedidas (mínimo, máximo, soma ou média). • O número de solicitações bem-sucedidas (SampleCount). SuccessfulRequestLatency reflete a atividade somente no Timestream LiveAnalytics e não leva em consideração a latência da rede ou a atividade do lado do cliente. Unidades: `Milliseconds` Dimensões • `DatabaseName` • `TableName` • `Operation` Estatística válida: • `Minimum` • `Maximum` • `Average` • `SampleCount` • `P10` • `p50` • `p90` • `p95` • `p99`

Métricas de escrita e armazenamento

Métrica	Descrição
`MagneticStoreRejectedRecordCount`	O número de registros gravados no armazenamento magnético que foram rejeitados de forma assíncrona. Isso pode acontecer se o novo registro tiver uma versão menor que a versão atual ou se o novo registro tiver uma versão igual à versão atual, mas tiver dados diferentes. Unidades: `Count` Dimensões • `DatabaseName` • `TableName` • `Operation` Estatística válida: • `Sum` • `SampleCount`
`MagneticStoreRejectedUploadUserFailures`	O número de relatórios de registros rejeitados pela loja magnética que não foram enviados devido a erros do usuário. Isso pode ser devido a IAM permissões não configuradas corretamente ou a um bucket do S3 excluído. Unidades: `Count` Dimensões • `DatabaseName` • `TableName` • `Operation`

Métrica	Descrição
	Estatística válida: • `Sum` • `SampleCount`
`MagneticStoreRejectedUploadSystemFailures`	O número de relatórios de registros rejeitados pelo Magnetic Store que não foram enviados devido a erros do sistema. Unidades: `Count` Dimensões • `DatabaseName` • `TableName` • `Operation` Estatística válida: • `Sum` • `SampleCount`

Métrica	Descrição
`ActiveMagneticStorePartitions`	O número de partições magnéticas de armazenamento que ingerem dados ativamente em um determinado momento. Unidades: `Count` Dimensões • `DatabaseName` • `Operation` Estatística válida: • `Sum` • `SampleCount`

Métrica	Descrição
`MagneticStorePendingRecordsLatency`	Os mais antigos gravam em um repositór io magnético que não está disponível para consulta. Os registros gravados na loja magnética estarão disponíveis para consulta em 6 horas. Unidades: `Milliseconds` Dimensões • `DatabaseName` • `TableName` • `Operation` Estatística válida: • `Minimum` • `Maximum` • `Average` • `SampleCount` • `P10` • `p50` • `p90` • `p95` • `p99`

Métrica	Descrição
`MemoryCumulativeBytesMetered`	A quantidade de dados armazenados no armazenamento de memória, em bytes Unidades: `Bytes` Dimensões: `Operation` Estatística válida: • `Average`
`MagneticCumulativeBytesMetered`	A quantidade de dados armazenados no armazenamento magnético, em bytes Unidades: `Bytes` Dimensões: `Operation` Estatística válida: • `Average`
`CumulativeBytesMetered`	A quantidade de dados medidos por ingestão no Timestream for LiveAnalytics, em bytes. Unidades: `Bytes` Dimensões: `Operation` Estatística válida: `Sum`
`NumberOfRecords`	O número de registros ingeridos no Timestrea m para. LiveAnalytics Unidades: `Count` Dimensões: `Operation` Estatística válida: `Sum`

Métricas de consulta

Métrica	Descrição
`CumulativeBytesMetered`	A quantidade de dados digitalizados pelas consultas enviadas ao Timestream em, em bytes. LiveAnalytics Unidades: `Bytes` Dimensões: `Operation` Estatística válida: • `Sum`
`ResourceCount`	As unidades de computação do Timestream (TCUs) consumidas para a carga de trabalho de consulta na conta. Essa métrica é emitida com unidades computacionais máximas e mínimas para cada minuto durante a carga de trabalho de consulta ativa da conta. Unidades: `Count` Estatísticas válidas: mínimo, máximo Dimensões:`Service: Timestrea m` ,`Resource: QueryTCU`,`Type: Resource`, `Class: OnDemand`

Métricas de erro

Métrica	Descrição
`SystemErrors`	As solicitações ao Timestream para LiveAnaly tics isso geram um SystemError durante o período de tempo especificado. A SystemError geralmente indica um erro interno de serviço.

Métrica	Descrição
	Unidades: `Count` Dimensões: `Operation` Estatística válida: • `Sum` • `SampleCount`
`UserErrors`	Solicitações ao Timestream para LiveAnaly tics isso geram um InvalidRequest erro durante o período especificado. InvalidRequest Geralmente, indica um erro do lado do cliente, como uma combinação inválida de parâmetros, uma tentativa de atualizar uma tabela inexisten te ou uma assinatura de solicitação incorreta. UserErrors representa o agregado de solicitaç ões inválidas para a AWS região atual e a conta atual AWS . Unidades: `Count` Dimensões: `Operation` Estatística válida: • `Sum` • `SampleCount`

Important

Nem todas as estatísticas, como `Average` ou `Sum`, são aplicáveis a todas as métricas. No entanto, todos esses valores estão disponíveis por meio do Timestream para LiveAnalytics console, ou usando o CloudWatch console AWS CLI, ou AWS SDKs para todas as métricas.

Criação de CloudWatch alarmes para monitorar o Timestream para LiveAnalytics

Você pode criar um CloudWatch alarme da Amazon para o Timestream para LiveAnalytics que envie uma mensagem do Amazon Simple Notification Service SNS (Amazon) quando o alarme mudar de estado. Um alarme observa uma única métrica por um período tempo que você especifica. Ele executa uma ou mais ações com base no valor da métrica em relação a um limite especificado ao longo de vários períodos. A ação é uma notificação enviada para um SNS tópico da Amazon ou uma política de Auto Scaling.

Os alarmes invocam ações somente para mudanças de estado sustentadas. CloudWatch os alarmes não invocam ações simplesmente porque estão em um estado específico. O estado deve ter sido alterado e mantido por uma quantidade especificada de períodos.

Para obter mais informações sobre a criação de CloudWatch alarmes, consulte Usando CloudWatch alarmes da Amazon no Guia CloudWatch do usuário da Amazon.

Solução de problemas

Esta seção contém informações sobre a solução de problemas do Timestream para. LiveAnalytics

Tópicos

- Manipulação de WriteRecords aceleradores
- Lidando com registros rejeitados
- Solução de problemas UNLOAD do Timestream para LiveAnalytics
- Timestream para códigos de erro LiveAnalytics específicos

Manipulação de WriteRecords aceleradores

As solicitações de gravação do armazenamento de memória para o Timestream podem ser limitadas à medida que o Timestream é dimensionado para se adaptar às necessidades de ingestão de dados do seu aplicativo. Se seus aplicativos encontrarem exceções de limitação, você deverá continuar enviando dados com a mesma taxa de transferência (ou maior) para permitir que o Timestream seja escalado automaticamente de acordo com as necessidades do seu aplicativo.

Suas solicitações de gravação de armazenamento magnético no Timestream podem ser limitadas se o limite máximo de partições de armazenamento magnético receberem ingestão. Você verá uma mensagem de aceleração orientando você a verificar a métrica do `ActiveMagneticStorePartitions` Cloudwatch para esse banco de dados. Esse acelerador

pode levar até 6 horas para ser resolvido. Para evitar essa aceleração, você deve usar o armazenamento de memória para qualquer carga de trabalho de ingestão de alta taxa de transferência. Para a ingestão de armazenamento magnético, você pode direcionar a ingestão em menos partições limitando quantas séries e o tempo de duração da ingestão são inseridas

Para obter mais informações sobre as melhores práticas de ingestão de dados, consulteEscreve.

Lidando com registros rejeitados

Se o Timestream rejeitar registros, você receberá um `RejectedRecordsException` com detalhes sobre a rejeição. Consulte Como lidar com a falha de gravação para obter mais informações sobre como extrair essas informações da WriteRecords resposta.

Todas as rejeições serão incluídas nesta resposta, com exceção das atualizações no armazenamento magnético em que a versão do novo registro é menor ou igual à versão do registro existente. Nesse caso, o Timestream não atualizará o registro existente que tem a versão superior. O Timestream rejeitará o novo registro com uma versão inferior ou igual e gravará esses erros de forma assíncrona em seu bucket do S3. Para receber esses relatórios de erros assíncronos, você deve definir a `MagneticStoreRejectedDataLocation` propriedade `MagneticStoreWriteProperties` em sua tabela.

Solução de problemas UNLOAD do Timestream para LiveAnalytics

A seguir estão as orientações para solução de problemas relacionados ao UNLOAD comando.

Categoria	Mensagem de erro	Como solucionar problemas
Comprimento da chave S3	UNLOADa chave do arquivo de resultado ao usar o prefixo S3 [%s] fornecido no destino excederá o tamanho de chave permitido para o S3. Consulte a documentação para obter mais detalhes.	Ao exportar os resultados da consulta usando a `UNLOAD` instrução, o tamanho da chave do S3, composto pela soma do tamanho do nome e do prefixo do bucket do S3, excede o tamanho máximo da chave do S3 suportado . Recomendamos reduzir o tamanho do prefixo ou do nome do bucket.

Categoria	Mensagem de erro	Como solucionar problemas
	UNLOADa chave do arquivo de resultado ao usar partitioned_by [%s] excederá o tamanho de chave permitido pelo S3. Consulte a documentação para obter mais detalhes.	Ao exportar os resultados da consulta usando a UNLOAD instrução, o tamanho da chave S3 usando a coluna partitioned_by excede o tamanho máximo da chave S3 suportado. Recomenda mos particionar com uma coluna alternativa ou reduzir o comprimento da coluna_pa rticionada (se possível).
	UNLOADa chave do arquivo de resultado ao usar o prefixo S3 [%s] junto com o partition ed_by [%s] excederá o tamanho de chave permitido do S3. Consulte a documenta ção para obter mais detalhes.	Ao exportar os resultados da consulta usando a UNLOAD instrução, o comprimento da chave S3, composto pela soma do tamanho do nome do bucket do S3, do prefixo e do nome da coluna partition ed_by, excede o tamanho máximo suportado da chave do S3. Recomendamos reduzir o prefixo, o tamanho do nome do bucket ou usar uma coluna alternativa para particionar seus dados.

Categoria	Mensagem de erro	Como solucionar problemas
	A chave de objeto S3 gerada: %s é muito longa. Consulte a documentação para obter mais detalhes.	Ao processar sua consulta usando a UNLOAD instrução , um dos valores na coluna particionada excede o tamanho máximo de chave S3 suportado. A coluna e o valor da partição podem ser encontrados na chave do objeto gerada.
Aceleradores S3	Detectamos que o Amazon S3 está limitando as gravações do comando. UNLOAD Consulte a documentação do Amazon Timestream para obter mais informações	Consulte a documentação do S3 aqui. A taxa de API chamadas do S3 pode ser reduzida quando vários leitores/gravadores acessam a mesma pasta. Audite o volume de chamadas para o bucket fornecido. Se você estiver usando o mesmo bucket para várias UNLOAD consultas simultâneas, tente usar buckets diferentes para o mesmo. Se você estiver usando o mesmo intervalo para várias operações além do Timestream LiveAnalytics UNLOAD, considere mover UNLOAD os resultados para um intervalo separado.

Timestream para códigos de erro LiveAnalytics específicos

Esta seção contém os códigos de erro específicos do Timestream for. LiveAnalytics

Timestream para LiveAnalytics erros de gravação API

InternalServerException

HTTPCódigo de status: 500

ThrottlingException

HTTPCódigo de status: 429

ValidationException

HTTPCódigo de status: 400

ConflictException

HTTPCódigo de status: 409

AccessDeniedException

Você não tem acesso suficiente para executar essa ação.

HTTPCódigo de status: 403

ServiceQuotaExceededException

HTTPCódigo de status: 402

ResourceNotFoundException

HTTPCódigo de status: 404

RejectedRecordsException

HTTPCódigo de status: 419

InvalidEndpointException

HTTPCódigo de status: 421

Cronograma para LiveAnalytics erros de consulta API

ValidationException

HTTPCódigo de status: 400

QueryExecutionException

HTTPCódigo de status: 400

ConflictException

HTTPCódigo de status: 409

ThrottlingException

HTTPCódigo de status: 429

InternalServerException

HTTPCódigo de status: 500

InvalidEndpointException

HTTPCódigo de status: 421

Cotas

Este tópico descreve as cotas atuais, também chamadas de limites, dentro do Amazon Timestream for. LiveAnalytics Salvo indicação em contrário, cada cota aplica-se por região.

Tópicos

- Cotas padrão
- Limites do serviço
- Tipos de dados compatíveis
- Carregamento em lote
- Restrições de nomenclatura
- Palavras-chave reservadas
- Identificadores do sistema
- UNLOAD

Cotas padrão

A tabela a seguir contém o Timestream para LiveAnalytics cotas e os valores padrão.

displayName	Descrição	defaultValue
Bancos de dados por conta	O número máximo de bancos de dados que você pode criar por Conta da AWS.	500
Tabelas por conta	O número máximo de tabelas que você pode criar por Conta da AWS.	50000
Taxa de aceleração para CRUD APIs	O número máximo de API solicitações de Create/Update/List/Describe/Delete database/table/scheduled consulta permitidas por segundo por conta, na região atual.	1
Consultas agendadas por conta da	O número máximo de consultas agendadas que você pode criar por. Conta da AWS	10000
Contagem máxima de partições de armazenamento magnético ativas	O número máximo de partições de armazenamento magnético ativas por banco de dados. Uma partição pode permanecer ativa por até seis horas após o recebimento da ingestão.	250
maxQueryTCU	A consulta máxima TCUs que você pode definir para sua conta.	1000

Limites do serviço

A tabela a seguir contém o Timestream para limites LiveAnalytics de serviço e os valores padrão. Para editar a retenção de dados de uma tabela no console, consulte Editar uma tabela.

displayName	Descrição	defaultValue
Período futuro de ingestão em minutos	O lead time máximo (em minutos) para seus dados de séries temporais em comparação com o horário atual do sistema. Por exemplo, se o período de ingestão futura for de 15 minutos, o Timestream for LiveAnalytics aceitará dados com até 15 minutos de antecedência em relação à hora atual do sistema.	15
Período mínimo de retenção para armazenamento de memória em horas	A duração mínima (em horas) pela qual os dados precisam ser retidos no armazenamento de memória por tabela.	1
Período máximo de retenção para armazenamento de memória em horas	A duração máxima (em horas) pela qual os dados podem ser retidos no armazenamento de memória por tabela.	876
Período mínimo de retenção para armazenamento magnético em dias	A duração mínima (em dias) pela qual os dados podem ser retidos no armazenamento magnético por tabela.	1
Período de retenção para armazenamento magnético em dias	A duração máxima (em dias) pela qual os dados podem ser retidos no armazenam	73000

displayName	Descrição	defaultValue
	ento magnético. Esse valor é equivalente a 200 anos.	
Período de retenção padrão para armazenamento magnético em dias	O valor padrão (em dias) para o qual os dados são retidos no armazenamento magnético por tabela. Esse valor é equivalente a 200 anos.	73000
Período de retenção padrão para armazenamento de memória em horas	A duração padrão (em horas) pela qual os dados são retidos no armazenamento de memória.	6
Dimensões por tabela	O número máximo de dimensões por tabela.	128
Nomes de medidas por tabela	O número máximo de nomes de medidas exclusivos por tabela.	8192
Nome da dimensão, valor da dimensão, par, tamanho por série	Par tamanho máximo do nome da dimensão e valor da dimensão por série.	2 kilobytes
Tamanho máximo do registro	O tamanho máximo de um registro.	2 kilobytes
Registros por WriteRecords API solicitação	O número máximo de registros em uma WriteRecords API solicitação.	100
Comprimento do nome da dimensão	O número máximo de bytes para um nome de Dimensão.	60 bytes

displayName	Descrição	defaultValue
Medir o comprimento do nome	O número máximo de bytes para o nome de uma medida.	256 bytes
Tamanho do nome do banco de dados	O número máximo de bytes para um nome de banco de dados.	256 bytes
Comprimento do nome da tabela	O número máximo de bytes para um nome de tabela.	256 bytes
QueryString comprimento em KiB	O tamanho máximo (em KiB) de uma string de consulta em UTF -8 caracteres codificados para uma consulta.	256
Duração da execução das consultas em horas	A duração máxima de execução (em horas) de uma consulta. As consultas que demoram mais atingirão o tempo limite.	1
Insights de consulta	O número máximo de API solicitações de consulta permitidas com insights de consulta ativados por segundo por conta, na região atual.	1
Tamanho dos metadados para o resultado da consulta	O tamanho máximo de metadados de um resultado de consulta.	100 quilobytes
Tamanho dos dados para o resultado da consulta	O tamanho máximo de dados de um resultado de consulta.	5 Gigabytes
Medidas por registro de várias medidas	Número máximo das medidas por registro de várias medidas.	256

displayName	Descrição	defaultValue
Meça o tamanho do valor por registro de várias medidas	Tamanho máximo dos valores de medida por registro de várias medidas.	2048
Medidas exclusivas em registros de várias medidas por tabela	As medidas exclusivas em todos os registros de várias medidas definidos em uma única tabela.	1024
Unidades de computação de fluxo de tempo () TCUs por conta	O máximo padrão TCUs por conta.	200

Tipos de dados compatíveis

A tabela a seguir descreve os tipos de dados suportados para valores de medida e dimensão.

Descrição	Cronograma de valor LiveAnalytics
Tipos de dados compatíveis para valores de medida.	Big int, double, string, booleanoMULTI, timestamp
Tipos de dados compatíveis para valores de dimensão.	String

Carregamento em lote

As cotas atuais, também chamadas de limites, dentro do carregamento em lote são as seguintes.

Descrição	Cronograma de valor LiveAnalytics
Tamanho máximo da tarefa de carregamento em lote	O tamanho máximo da tarefa de carregamento em lote não pode exceder 100 GB.
Quantidade de arquivos	Uma tarefa de carregamento em lote não pode ter mais de 100 arquivos.
Tamanho máximo do arquivo	O tamanho máximo do arquivo em uma tarefa de carregamento em lote não pode exceder 5 GB.
CSVtamanho da linha do arquivo	Uma linha em um CSV arquivo não pode exceder 16 MB. Esse é um limite rígido que não pode ser aumentado.
Tarefas ativas de carregamento em lote	Uma tabela não pode ter mais de 5 tarefas ativas de carregame nto em lote e uma conta não pode ter mais de 10 tarefas ativas de carregamento em lote. O Timestream for LiveAnalytics acelerará novas tarefas de carregamento em lote até que mais recursos estejam disponíveis.

Restrições de nomenclatura

A tabela a seguir descreve as restrições de nomenclatura.

Descrição	Cronograma de valor LiveAnalytics
O comprimento máximo do nome de uma dimensão.	60 bytes
O tamanho máximo do nome de uma medida.	256 bytes
O tamanho máximo de um nome de tabela ou nome de banco de dados.	256 bytes

Descrição	Cronograma de valor LiveAnalytics
Nome da tabela e do banco de dados	• Recomendamos que você não useIdentificadores do sistema. • Pode conter a-z A-Z 0-9 _ (sublinhado) - (traço). (ponto). • Todos os nomes devem ser codificados como UTF -8 e diferenci am maiúsculas de minúsculas. **Note** Os nomes da tabela e do banco de dados são comparados usando a representação binária UTF -8. Isso significa que a comparação de ASCII caracteres diferencia maiúsculas de minúsculas.
Nome da medida	• Não deve conter Identificadores do sistema nem dois pontos ':'. • Não deve começar com um prefixo reservado (`ts_`,`measure_value`). **Note** Os nomes da tabela e do banco de dados são comparados usando a representação binária UTF -8. Isso significa que a comparação de ASCII caracteres diferencia maiúsculas de minúsculas.

Descrição	Cronograma de valor LiveAnalytics
Nome da dimensão	• Não deve conter Identificadores do sistema dois pontos ':' ou aspas duplas (“). • Não deve começar com um prefixo reservado (`ts_`,`measure_value`). • Não deve conter caracteres Unicode [0,31] listados aqui ou “\ u2028" ou “\ u2029". **Note** Os nomes de dimensões e medidas são comparados usando a representação binária UTF -8. Isso significa que a comparação de ASCII caracteres diferencia maiúsculas de minúsculas.
Todos os nomes das colunas	Os nomes das colunas não podem ser duplicados. Como os registros de várias medidas representam dimensões e medidas como colunas, o nome de uma dimensão não pode ser igual ao nome de uma medida. Os nomes diferenciam letras maiúsculas de minúsculas.

Palavras-chave reservadas

Todas as seguintes são palavras-chave reservadas:

- 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
- OU
- ORDER
- OUTER
- PREPARE
- RECURSIVE
- RIGHT
- ROLLUP
- SELECT
- TABLE
- THEN
- TRUE
- UESCAPE
- UNION
- UNNEST
- USING
- VALUES
- WHEN

- WHERE
- WITH

Identificadores do sistema

Reservamos os nomes das colunas “measure_value”, “ts_non_existent_col” e “time” como Timestream para identificadores do sistema. LiveAnalytics Além disso, os nomes das colunas não podem começar com “ts_” ou “measure_name”. Os identificadores do sistema diferenciam maiúsculas de minúsculas. Identificadores comparados usando representação binária UTF -8. Isso significa que a comparação de identificadores diferencia maiúsculas de minúsculas.

> **Note**
>
> Os identificadores do sistema não podem ser usados para nomes de dimensões ou medidas. Recomendamos que você não use identificadores de sistema para nomes de bancos de dados ou tabelas.

UNLOAD

Para ver os limites relacionados ao `UNLOAD` comando, consulte Usando UNLOAD para exportar os resultados da consulta para o S3 a partir do Timestream.

Referência da linguagem de consulta

> **Note**
>
> Essa referência de linguagem de consulta inclui a seguinte documentação de terceiros da Trino Software Foundation (antiga Presto Software Foundation), que é licenciada sob a Licença Apache, versão 2.0. Você não pode usar esse arquivo, exceto em conformidade com esta licença. Para obter uma cópia da Licença Apache, versão 2.0, consulte o site da Apache.

O Timestream for LiveAnalytics suporta uma linguagem de consulta avançada para trabalhar com seus dados. Você pode ver os tipos de dados, operadores, funções e construções disponíveis abaixo.

Você também pode começar imediatamente com a linguagem de consulta do Timestream na Consultas de exemplo seção.

Tópicos

- Tipos de dados compatíveis
- Funcionalidade de série temporal integrada
- SQLapoio
- Operadores lógicos
- Operadores de comparação
- Funções de comparação
- Expressões condicionais
- Funções de conversão
- Operadores matemáticos
- Funções matemáticas
- Operadores de string
- Funções de string
- Operadores de matriz
- Funções de array
- Funções bitwise
- Funções de expressões regulares
- Operadores de data/hora
- Funções de data/hora
- Funções agregadas
- Funções de janela
- Consultas de exemplo

Tipos de dados compatíveis

A linguagem de consulta LiveAnalytics do Timestream for oferece suporte aos seguintes tipos de dados.

> **Note**
>
> Os tipos de dados compatíveis com gravações são descritos em Tipos de dados.

Tipo de dados	Descrição
`int`	Representa um número inteiro de 32 bits.
`bigint`	Representa um inteiro assinado de 64 bits.
`boolean`	Um dos dois valores verdadeiros da lógica, `True` `False` e.
`double`	Representa um tipo de dados de precisão variável de 64 bits. Implementa o IEEEpadrão 754 para aritmética binária de ponto flutuante. **Note** A linguagem de consulta é para leitura de dados. Existem funções para `Infinity` e valores `NaN` duplos que podem ser usados em consultas. Mas você não pode gravar esses valores no Timestream.
`varchar`	Dados de caracteres de comprimento variável com tamanho máximo de 2 KB.
`array[T,...]`	Contém um ou mais elementos de um tipo de dados especific ado *T*em que, *T* pode ser qualquer um dos tipos de dados compatíveis com o Timestream.
`row(T,...)`	Contém um ou mais campos nomeados do tipo de dados *T*. Os campos podem ser de qualquer tipo de dados suportado pelo Timestream e são acessados com o operador de referência de campo de pontos: `.`

Tipo de dados	Descrição
`date`	Representa uma data no *YYYY*-*MM*-*DD* formulário. onde *YYYY* é o ano, *MM* é o mês, e *DD* é o dia, respectivamente. O intervalo suportado é `1970-01-01` de `2262-04-11` a. Exemplo: `1971-02-03`
`time`	Representa a hora do dia em UTC. O `time` tipo de dados é representado no formato *HH*`.`*MM*`.`*SS*`.`*sssssssss* `.` Suporta precisão de nanossegundos. Exemplo: `17:02:07.496000000`
`timestamp`	Representa uma instância no tempo usando tempo de precisão de nanossegundos. UTC *YYYY*-*MM*-*DD* *hh*:*mm*:*ss*.*sssssssss* O Query oferece suporte a carimbos de data/hora no intervalo de. `1677-09-21 00:12:44.000000000` `2262-04-11 23:47:16.854775807`

Tipo de dados	Descrição
`interval`	Representa um intervalo de tempo como uma string literal*`Xt`*, composta por duas partes, *`X`* e *`t`*. *`X`* é um valor numérico maior ou igual a`0`, e *`t`* é uma unidade de tempo, como segundo ou hora. A unidade não está pluralizada. A unidade de tempo *`t`* is deve ser um dos seguintes literais de string: • `nanosecond` • `microsecond` • `millisecond` • `second` • `minute` • `hour` • `day` • `ns`(o mesmo que`nanosecond`) • `us`(o mesmo que`microsecond`) • `ms`(o mesmo que`millisecond`) • `s`(o mesmo que`second`) • `m`(o mesmo que`minute`) • `h`(o mesmo que`hour`) • `d`(o mesmo que`day`) Exemplos: `17s` `12second` `21hour`

Tipo de dados	Descrição
	`2d`
`timeseries[row(timestamp, T,...)]`	Representa os valores de uma medida registrada em um intervalo de tempo como uma `array` composição de `row` objetos. Cada um `row` contém um `timestamp` e um ou mais valores de medida do tipo de dados *T*em que, *T* pode ser qualquer um dos`bigint`,`boolean`,`double`, ou`varchar`. As linhas são classificadas em ordem crescente por. `timestamp` O tipo de dados da série temporal representa os valores de uma medida ao longo do tempo.
`unknown`	Representa dados nulos.

Funcionalidade de série temporal integrada

O Timestream for LiveAnalytics fornece funcionalidade de série temporal integrada que trata os dados de séries temporais como um conceito de primeira classe.

A funcionalidade de série temporal integrada pode ser dividida em duas categorias: visualizações e funções.

Você pode ler sobre cada construção abaixo.

Tópicos

- Visualizações da série temporal
- Funções de séries temporais

Visualizações da série temporal

O Timestream for LiveAnalytics suporta as seguintes funções para transformar seus dados no tipo de dados: `timeseries`

Tópicos

- CREATE_TIME_SERIES
- UNNEST

CREATE_TIME_SERIES

CREATE_ TIME _ SERIES é uma função de agregação que obtém todas as medições brutas de uma série temporal (valores de tempo e medida) e retorna um tipo de dados de série temporal. A sintaxe dessa função é a seguinte:

```
CREATE_TIME_SERIES(time, measure_value::<data_type>)
```

onde `<data_type>` é o tipo de dados do valor da medida e pode ser bigint, boolean, double ou varchar. O segundo parâmetro não pode ser nulo.

Considere a CPU utilização de EC2 instâncias armazenadas em uma tabela chamada métricas, conforme mostrado abaixo:

Tempo	região	az	vpc	instance_id	nome_medida	valor_medida::duplo
2019-12-04 19:00:00.000 000000	us-east-1	leste dos EUA - 1d	vpc-1a2b3c4d	i-1234567890abcdef0	utilização_da CPU	35,0
2019-12-04 19:00:01.000 000000	us-east-1	leste dos EUA - 1d	vpc-1a2b3c4d	i-1234567890abcdef0	utilização_da CPU	38,2
2019-12-04 19:00:02.000 000000	us-east-1	leste dos EUA - 1d	vpc-1a2b3c4d	i-1234567890abcdef0	utilização_da CPU	45,3
2019-12-04 19:00:00.000 000000	us-east-1	leste dos EUA - 1d	vpc-1a2b3c4d	i-1234567890abcdef1	utilização_da CPU	54.1

Tempo	região	az	vpc	instance_id	nome_medida	valor_medida::duplo
2019-12-04 19:00:01.000 000000	us-east-1	leste dos EUA - 1d	vpc-1a2b3c4d	i-1234567890abcdef1	utilização_da CPU	42,5
2019-12-04 19:00:02.000 000000	us-east-1	leste dos EUA - 1d	vpc-1a2b3c4d	i-1234567890abcdef1	utilização_da CPU	3.7

Executando a consulta:

```
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
```

retornará todas as séries que tenham `cpu_utilization` como medida um valor. Nesse caso, temos duas séries:

região	az	vpc	instance_id	utilização_da CPU
us-east-1	leste dos EUA - 1d	vpc-1a2b3c4d	i-1234567890abcdef0	[{time: 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:

região	az	vpc	instance_id	utilização_da CPU
				38.2}, {time: 2019-12-0 4 19:00:02. 000 000000, value_mea sure: :double: 45.3}]
us-east-1	leste dos EUA - 1d	vpc-1a2b3c4d	i-1234567 890abcdef1	[{time: 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, value_mea sure: :double: 45.7}]

UNNEST

`UNNEST`é uma função de tabela que permite transformar `timeseries` dados no modelo plano. A sintaxe é a seguinte:

`UNNEST`transforma a `timeseries` em duas colunas, a saber, `time` e. `value` Você também pode usar aliases UNNEST conforme mostrado abaixo:

```
UNNEST(timeseries) AS <alias_name> (time_alias, value_alias)
```

onde <alias_name> é o alias da tabela simples, `time_alias` é o alias da `time` coluna e `value_alias` é o alias da coluna. `value`

Por exemplo, considere o cenário em que algumas das EC2 instâncias da sua frota estão configuradas para emitir métricas em um intervalo de 5 segundos, outras emitem métricas em um intervalo de 15 segundos, e você precisa das métricas médias de todas as instâncias em uma granularidade de 10 segundos nas últimas 6 horas. Para obter esses dados, você transforma suas métricas no modelo de séries temporais usando CREATE_ TIME _ SERIES. Em seguida, você pode usar INTERPOLATE_ LINEAR para obter os valores ausentes na granularidade de 10 segundos. Em seguida, você transforma os dados de volta no modelo plano usando e UNNEST, em seguida, usa AVGpara obter as métricas médias em todas as instâncias.

```
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
```

A consulta acima demonstra o uso de UNNESTcom um alias. Abaixo está um exemplo da mesma consulta sem usar um alias para 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
```

Funções de séries temporais

O Amazon Timestream LiveAnalytics for oferece suporte a funções de séries temporais, como derivadas, integrais e correlações, além de outras, para obter insights mais profundos de seus dados de séries temporais. Esta seção fornece informações de uso para cada uma dessas funções, bem como exemplos de consultas. Selecione um tópico abaixo para saber mais.

Tópicos

- Funções de interpolação
- Funções derivadas
- Funções integrais
- Funções de correlação
- Funções de filtragem e redução

Funções de interpolação

Se seus dados de série temporal não tiverem valores para eventos em determinados momentos, você poderá estimar os valores desses eventos ausentes usando a interpolação. O Amazon Timestream suporta quatro variantes de interpolação: interpolação linear, interpolação por spline cúbica, interpolação por última observação transportada (locf) e interpolação constante. Esta seção fornece informações de uso do Timestream para funções de LiveAnalytics interpolação, bem como exemplos de consultas.

Informações de uso

Função	Tipo de dados de saída	Descrição
`interpolate_linear (timeseries, array[timestamp])`	série temporal	Preenche os dados ausentes usando interpolação linear.

Função	Tipo de dados de saída	Descrição
`interpolate_linear (timeseries, timestamp)`	double	Preenche os dados ausentes usando interpolação linear.
`interpolate_spline _cubic(timeseries, array[timestamp])`	série temporal	Preenche os dados ausentes usando a interpolação de spline cúbica.
`interpolate_spline _cubic(timeseries, timestamp)`	double	Preenche os dados ausentes usando a interpolação de spline cúbica.
`interpolate_locf(t imeseries, array[tim estamp])`	série temporal	Preenche os dados ausentes usando o último valor amostrado.
`interpolate_locf(t imeseries, timestamp )`	double	Preenche os dados ausentes usando o último valor amostrado.
`interpolate_fill(t imeseries, array[tim estamp], double)`	série temporal	Preenche os dados ausentes usando um valor constante.
`interpolate_fill(t imeseries, timestamp , double)`	double	Preenche os dados ausentes usando um valor constante.

Exemplos de consulta

Example

Encontre a CPU utilização média armazenada em intervalos de 30 segundos para um EC2 host específico nas últimas 2 horas, preenchendo os valores ausentes usando a interpolação linear:

```
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

Encontre a CPU utilização média armazenada em intervalos de 30 segundos para um EC2 hospedeiro específico nas últimas 2 horas, preenchendo os valores ausentes usando a interpolação com base na última observação realizada:

```
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)
```

Funções derivadas

Os derivativos são usados para calcular a taxa de variação de uma determinada métrica e podem ser usados para responder proativamente a um evento. Por exemplo, suponha que você calcule a derivada da CPU utilização de EC2 instâncias nos últimos 5 minutos e observe uma derivada positiva significativa. Isso pode ser indicativo do aumento da demanda em sua carga de trabalho, então você pode decidir ativar mais EC2 instâncias para lidar melhor com sua carga de trabalho.

O Amazon Timestream oferece suporte a duas variantes de funções derivadas. Esta seção fornece informações de uso do Timestream para funções LiveAnalytics derivadas, bem como exemplos de consultas.

Informações de uso

Função	Tipo de dados de saída	Descrição
`derivative_linear(timeseries, interval)`	série temporal	Calcula a derivada de cada ponto no `timeseries` para o especificado. `interval`
`non_negative_derivative_linear(timeseries, interval)`	série temporal	O mesmo que `derivative_linear(timeseries, interval)`, mas retorna apenas valores positivos.

Exemplos de consulta

Example

Encontre a taxa de variação na CPU utilização a cada 5 minutos na última 1 hora:

```
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

Calcule a taxa de aumento nos erros gerados por um ou mais microsserviços:

```
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
```

Funções integrais

Você pode usar integrais para encontrar a área sob a curva por unidade de tempo para seus eventos de série temporal. Por exemplo, suponha que você esteja monitorando o volume de solicitações recebidas pelo seu aplicativo por unidade de tempo. Nesse cenário, você pode usar a função integral para determinar o volume total de solicitações atendidas por intervalo especificado em um período específico.

O Amazon Timestream oferece suporte a uma variante de funções integrais. Esta seção fornece informações de uso do Timestream para a função LiveAnalytics integral, bem como exemplos de consultas.

Informações de uso

Função	Tipo de dados de saída	Descrição
`integral_trapezoidal(timeseries(double))`	double	Aproxima a integral de acordo com a especificada `interval day to second` para a

Função	Tipo de dados de saída	Descrição
`integral_trapezoidal(timeseries(double), interval day to second)` `integral_trapezoidal(timeseries(bigint))` `integral_trapezoidal(timeseries(bigint), interval day to second)` `integral_trapezoidal(timeseries(integer), interval day to second)` `integral_trapezoidal(timeseries(integer))`		`timeseries` fornecida, usando a regra trapezoidal. O parâmetro de intervalo de dia para segundo é opcional e o padrão é `1s`. Para obter mais informações sobre intervalos, consulte Intervalo e duração.

Exemplos de consulta

Example

Calcule o volume total de solicitações atendidas por cinco minutos na última hora por um host específico:

```
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
```

Funções de correlação

Dadas duas séries temporais de duração semelhante, as funções de correlação fornecem um coeficiente de correlação, o que explica como as duas séries temporais tendem ao longo do tempo. O coeficiente de correlação varia de a. `-1.0` `1.0` `-1.0`indica que as duas séries temporais tendem em direções opostas na mesma taxa, enquanto `1.0` indica que as duas séries temporais tendem na mesma direção e na mesma taxa. Um valor de `0` indica que não há correlação entre as duas séries temporais. Por exemplo, se o preço do petróleo aumentar e o preço das ações de uma empresa petrolífera aumentar, a tendência do aumento do preço do petróleo e o aumento do preço da companhia petrolífera terão um coeficiente de correlação positivo. Um alto coeficiente de correlação positivo indicaria que os dois preços tendem a uma taxa semelhante. Da mesma forma, o coeficiente de correlação entre os preços dos títulos e os rendimentos dos títulos é negativo, indicando que esses dois valores tendem na direção oposta ao longo do tempo.

O Amazon Timestream oferece suporte a duas variantes de funções de correlação. Esta seção fornece informações de uso do Timestream para funções de LiveAnalytics correlação, bem como exemplos de consultas.

Informações de uso

Função	Tipo de dados de saída	Descrição
`correlate_pearson( timeseries, timeseries)`	double	Calcula o coeficiente de correlação de Pearson para os dois. `timeseries` A série temporal deve ter os mesmos timestamps.
`correlate_spearman (timeseries, timeseries)`	double	Calcula o coeficiente de correlação de Spearman para os dois. `timeseries` A série temporal deve ter os mesmos timestamps.

Exemplos de consulta

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
```

Funções de filtragem e redução

O Amazon Timestream oferece suporte a funções para realizar operações de filtragem e redução em dados de séries temporais. Esta seção fornece informações de uso do Timestream para funções de LiveAnalytics filtragem e redução, bem como exemplos de consultas.

Informações de uso

Função	Tipo de dados de saída	Descrição
`filter(timeseries(T), function(T, Boolean))`	série temporal (T)	Constrói uma série temporal a partir de uma série temporal de entrada, usando valores para os quais o passado `function` retorna `true`.

Função	Tipo de dados de saída	Descrição
`reduce(timeseries(T), initialState S, inputFunction(S, T, S), outputFunction(S, R))`	R	Retorna um valor único, reduzido da série temporal. O `inputFunction` será invocado em cada elemento na série temporal em ordem. Além de pegar o elemento atual, inputFunction pega o estado atual (inicialm ente`initialState`) e retorna o novo estado. O `outputFunction` será invocado para transform ar o estado final no valor do resultado. `outputFun ction` Pode ser uma função de identidade.

Exemplos de consulta

Example

Construa uma série temporal de CPU utilização de um hospedeiro e de pontos de filtro com medição maior que 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

Construa uma série temporal de CPU utilização de um hospedeiro e determine a soma quadrada das medições:

```
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

Crie uma série temporal de CPU utilização de um hospedeiro e determine a fração de amostras que estão acima do CPU limite:

```
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
```

SQLapoio

O Timestream for LiveAnalytics suporta algumas construções comunsSQL. Você pode ler mais abaixo.

Tópicos

- SELECT
- Suporte para subconsultas
- SHOWdeclarações
- DESCRIBEdeclarações
- UNLOAD

SELECT

SELECTinstruções podem ser usadas para recuperar dados de uma ou mais tabelas. A linguagem de consulta do Timestream suporta a seguinte sintaxe para declarações: SELECT

```
[ 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)`é uma das funções de janela suportadas.
- `partition_expr_list` é:

```
expression | column_name [, expr_list ]
```

- `order_list` é:

```
expression | column_name [ ASC | DESC ]
[ NULLS FIRST | NULLS LAST ]
[, order_list ]
```

- `frame_clause` é:

```
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`é um dos seguintes:

```
table_name [ [ AS ] alias [ ( column_alias [, ...] ) ] ]
from_item join_type from_item [ ON join_condition | USING ( join_column [, ...] ) ]
```

- `join_type`é um dos seguintes:

```
[ INNER ] JOIN
LEFT [ OUTER ] JOIN
RIGHT [ OUTER ] JOIN
FULL [ OUTER ] JOIN
```

- `grouping_element`é um dos seguintes:

```
()
expression
```

Suporte para subconsultas

O Timestream suporta subconsultas e predicados. `EXISTS` `IN` O `EXISTS` predicado determina se uma subconsulta retorna alguma linha. O `IN` predicado determina se os valores produzidos pela

subconsulta correspondem aos valores ou à expressão da cláusula IN. A linguagem de consulta Timestream oferece suporte a subconsultas correlacionadas e outras.

```
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

SHOWdeclarações

Você pode visualizar todos os bancos de dados em uma conta usando o `SHOW DATABASES` extrato. A sintaxe é a seguinte:

```
SHOW DATABASES [LIKE pattern]
```

onde a `LIKE` cláusula pode ser usada para filtrar nomes de bancos de dados.

Você pode ver todas as tabelas em uma conta usando o `SHOW TABLES` extrato. A sintaxe é a seguinte:

```
SHOW TABLES [FROM database] [LIKE pattern]
```

onde a `FROM` cláusula pode ser usada para filtrar nomes de bancos de dados e a `LIKE` cláusula pode ser usada para filtrar nomes de tabelas.

Você pode visualizar todas as medidas de uma tabela usando a `SHOW MEASURES` declaração. A sintaxe é a seguinte:

```
SHOW MEASURES FROM database.table [LIKE pattern]
```

onde a `FROM` cláusula será usada para especificar o nome do banco de dados e da tabela e a `LIKE` cláusula poderá ser usada para filtrar nomes de medidas.

DESCRIBEdeclarações

Você pode visualizar os metadados de uma tabela usando a `DESCRIBE` instrução. A sintaxe é a seguinte:

```
DESCRIBE database.table
```

onde `table` contém o nome da tabela. A instrução describe retorna os nomes das colunas e os tipos de dados da tabela.

UNLOAD

Timestream for LiveAnalytics suporta um `UNLOAD` comando como uma extensão de seu SQL suporte. Os tipos de dados suportados pelo `UNLOAD` estão descritos emTipos de dados compatíveis. Os `unknown` tipos `time` e não se aplicam `UNLOAD` a.

```
UNLOAD (SELECT statement)
 TO 's3://bucket-name/folder'
 WITH ( option = expression [, ...] )
```

onde a opção é

```
{ 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>'
}
```

SELECTdeclaração

A instrução de consulta usada para selecionar e recuperar dados de um ou mais Timestream para tabelas. LiveAnalytics

```
(SELECT column 1, column 2, column 3 from database.table
      where measure_name = "ABC" and timestamp between ago (1d) and now() )
```

Cláusula TO

```
TO 's3://bucket-name/folder'
```

ou

```
TO 's3://access-point-alias/folder'
```

A TO cláusula na UNLOAD instrução especifica o destino para a saída dos resultados da consulta. Você precisa fornecer o caminho completo, incluindo o nome do bucket do Amazon S3 ou o Amazon S3 com a access-point-alias localização da pasta no Amazon S3, onde o Timestream grava os objetos do arquivo de saída. LiveAnalytics O bucket do S3 deve pertencer à mesma conta e estar na mesma região. Além do conjunto de resultados da consulta, o Timestream for LiveAnalytics grava os arquivos de manifesto e metadados na pasta de destino especificada.

PARTITIONEDCláusula _BY

```
partitioned_by = ARRAY [col_name[,…] , (default: none)
```

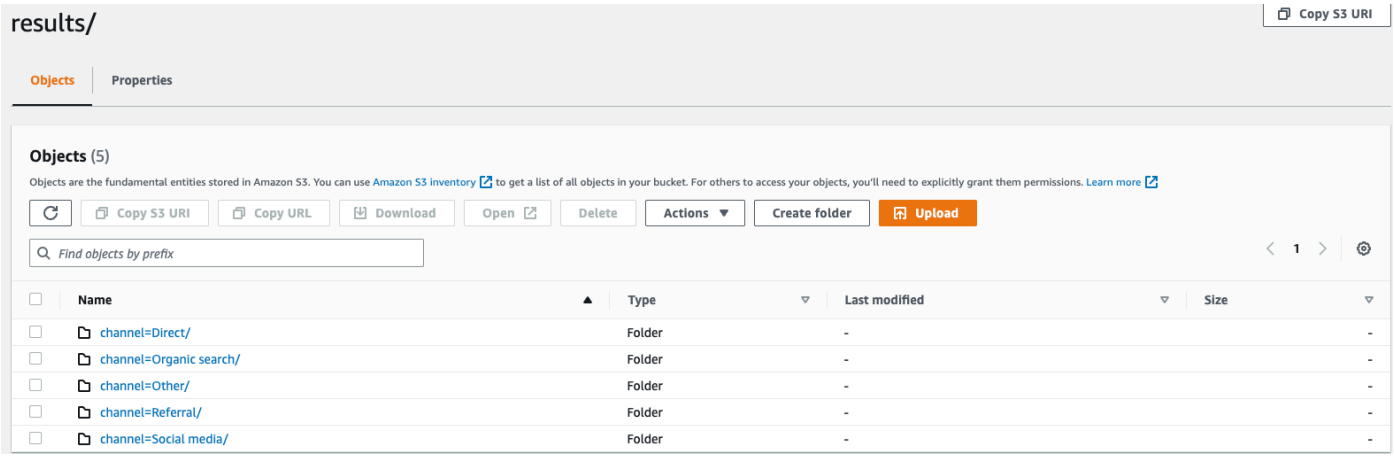
A `partitioned_by` cláusula é usada em consultas para agrupar e analisar dados em um nível granular. Ao exportar os resultados da consulta para o bucket do S3, você pode optar por particionar os dados com base em uma ou mais colunas na consulta selecionada. Ao particionar os dados, os dados exportados são divididos em subconjuntos com base na coluna de partição e cada subconjunto é armazenado em uma pasta separada. Na pasta de resultados que contém os dados exportados, uma subpasta `folder/results/partition column = partition value/` é criada automaticamente. No entanto, observe que as colunas particionadas não estão incluídas no arquivo de saída.

`partitioned_by`não é uma cláusula obrigatória na sintaxe. Se você optar por exportar os dados sem nenhum particionamento, poderá excluir a cláusula na sintaxe.

Example

Supondo que você esteja monitorando os dados do fluxo de cliques do seu site e tenha 5 canais de tráfego`direct`, a saber,`Social Media`,`Organic Search`, e. `Other` `Referral` Ao exportar os dados, você pode optar por particioná-los usando a coluna`Channel`. Em sua pasta de dados`s3://bucketname/results`, você terá cinco pastas, cada uma com o nome do respectivo canal. Por exemplo, `s3://bucketname/results/channel=Social Media/`. nessa pasta, você encontrará os dados de todos os clientes que acessaram seu site por meio do `Social Media` canal. Da mesma forma, você terá outras pastas para os canais restantes.

Dados exportados particionados pela coluna Canal



FORMAT

```
format = [ '{ CSV | PARQUET }' , default: CSV
```

As palavras-chave para especificar o formato dos resultados da consulta gravados em seu bucket do S3. Você pode exportar os dados como um valor separado por vírgula (CSV) usando uma vírgula (,) como delimitador padrão ou no formato Apache Parquet, um formato de armazenamento em colunas aberto eficiente para análise.

COMPRESSION

```
compression = [ '{ GZIP | NONE }' ], default: GZIP
```

Você pode compactar os dados exportados usando o algoritmo de compactação GZIP ou descompactá-los especificando a opção. NONE

ENCRYPTION

```
encryption = [ '{ SSE_KMS | SSE_S3 }' ], default: SSE_S3
```

Os arquivos de saída no Amazon S3 são criptografados usando a opção de criptografia selecionada. Além dos seus dados, os arquivos de manifesto e metadados também são criptografados com base na opção de criptografia selecionada. Atualmente, oferecemos suporte à SSE criptografia _S3 e SSE _KMS. SSE_S3 é uma criptografia do lado do servidor com o Amazon S3 criptografando os dados usando criptografia padrão de criptografia avançada () de 256 bits. AES SSE_ KMS é uma criptografia do lado do servidor para criptografar dados usando chaves gerenciadas pelo cliente.

KMS_KEY

```
kms_key = '<string>'
```

KMSA chave é uma chave definida pelo cliente para criptografar os resultados da consulta exportada. KMSA chave é gerenciada com segurança pelo AWS Key Management Service (AWS KMS) e usada para criptografar arquivos de dados no Amazon S3.

FIELD_DELIMITER

```
field_delimiter ='<character>' , default: (,)
```

Ao exportar os dados em CSV formato, esse campo especifica um único ASCII caractere usado para separar campos no arquivo de saída, como caractere vertical (|), vírgula (,) ou tabulação (/t). O delimitador padrão para CSV arquivos é um caractere de vírgula. Se um valor em seus dados contiver o delimitador escolhido, o delimitador será citado com um caractere de aspa. Por exemplo, se o valor em seus dados contiver`Time,stream`, esse valor será cotado como `"Time,stream"` nos dados exportados. O caractere de aspas usado pelo Timestream para LiveAnalytics são aspas duplas (“).

Evite especificar o caractere de retorno do carro (ASCII13, hexadecimal`0D`, texto '\ r') ou o caractere de quebra de linha (ASCII10, hexadecimal 0A, texto'\n') como `FIELD_DELIMITER` se você quiser incluir cabeçalhos noCSV, pois isso impedirá que muitos analisadores consigam analisar os cabeçalhos corretamente na saída resultante. CSV

ESCAPED_POR

```
escaped_by = '<character>', default: (\)
```

Ao exportar os dados em CSV formato, esse campo especifica o caractere que deve ser tratado como um caractere de escape no arquivo de dados gravado no bucket do S3. A fuga acontece nos seguintes cenários:

1. Se o valor em si contiver o caractere de aspa (“), ele será escapado usando um caractere de escape. Por exemplo, se o valor for`Time"stream`, onde (\) é o caractere de escape configurado, ele será escapado como`Time\"stream`.
2. Se o valor contiver o caractere de escape configurado, ele será escapado. Por exemplo, se o valor for`Time\stream`, ele será escapado como`Time\\stream`.

> **Note**
>
> Se a saída exportada contiver tipos de dados complexos, como matrizes, linhas ou séries temporais, ela será serializada como uma string. JSON Veja um exemplo a seguir.

Tipo de dados	Valor real	Como o valor é escapado no CSV formato [string serializa daJSON]
Array	`[ 23,24,25 ]`	`"[23,24,25]"`

Tipo de dados	Valor real	Como o valor é escapado no CSV formato [string serializa daJSON]
Linha	`( x=23.0, y=hello )`	`"{\"x\":23.0,\"y\": \"hello\"}"`
Série temporal	`[ ( 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\",\"val ue\":100.0},{\"tim e\":\"1970-01-01 00:00:00.000000012 Z\",\"value\":120. 0}]"`

INCLUDE_HEADER

```
include_header = 'true' , default: 'false'
```

Ao exportar os dados em CSV formato, esse campo permite incluir nomes de colunas como a primeira linha dos arquivos de CSV dados exportados.

Os valores aceitos são 'verdadeiro' e 'falso' e o valor padrão é 'falso'. Opções de transformação de texto, como `escaped_by` e também `field_delimiter` se aplicam aos cabeçalhos.

> **Note**
>
> Ao incluir cabeçalhos, é importante que você não selecione um caractere de retorno de carro (ASCII13, hexadecimal 0D, texto '\ r') ou um caractere de quebra de linha (ASCII10, hexadecimal 0A, texto'\n') como o`FIELD_DELIMITER`, pois isso impedirá que muitos analisadores consigam analisar os cabeçalhos corretamente na saída resultante. CSV

MAX_FILE_SIZE

```
max_file_size = 'X[MB|GB]' , default: '78GB'
```

Esse campo especifica o tamanho máximo dos arquivos que a UNLOAD declaração cria no Amazon S3. A UNLOAD declaração pode criar vários arquivos, mas o tamanho máximo de cada arquivo gravado no Amazon S3 será aproximadamente o especificado nesse campo.

O valor do campo deve estar entre 16 MB e 78 GB, inclusive. Você pode especificá-lo em números inteiros12GB, como, ou em decimais, como ou. 0.5GB 24.7MB O valor padrão é 78 GB.

O tamanho real do arquivo é aproximado quando o arquivo está sendo gravado, portanto, o tamanho máximo real pode não ser exatamente igual ao número especificado.

Operadores lógicos

O Timestream for LiveAnalytics suporta os seguintes operadores lógicos.

Operador	Descrição	Exemplo
AND	Verdadeiro se ambos os valores forem verdadeiros	a AND b
OU	Verdadeiro se qualquer um dos valores for verdadeiro	a OU b
NOT	Verdadeiro se o valor for falso	NOTuma

- O resultado de uma AND comparação pode ser NULL se um ou ambos os lados da expressão foremNULL.
- Se pelo menos um lado de um AND operador estiver, FALSE a expressão será avaliada como. FALSE
- O resultado de uma 0R comparação pode ser NULL se um ou ambos os lados da expressão foremNULL.
- Se pelo menos um lado de um 0R operador estiver, TRUE a expressão será avaliada como. TRUE
- O complemento lógico de NULL éNULL.

A tabela de verdades a seguir demonstra o tratamento de NULL in AND e0R:

A	B	A e b	A ou b
nulo	nulo	nulo	nulo
false	nulo	false	nulo
nulo	false	false	nulo
verdadeiro	nulo	nulo	verdadeiro
nulo	verdadeiro	nulo	verdadeiro
false	false	false	false
verdadeiro	false	false	verdadeiro
false	verdadeiro	false	verdadeiro
verdadeiro	verdadeiro	verdadeiro	verdadeiro

A tabela de verdades a seguir demonstra o tratamento de NULL in: NOT

A	Não é um
nulo	nulo
verdadeiro	false
false	verdadeiro

Operadores de comparação

O Timestream for LiveAnalytics suporta os seguintes operadores de comparação.

Operador	Descrição
<	Menor que

Operador	Descrição
>	Maior que
<=	Menor ou igual a
>=	Maior ou igual a
=	Equal
<>	Not equal
!=	Not equal

Note

- O `BETWEEN` operador testa se um valor está dentro de um intervalo especificado. A sintaxe é a seguinte:

  ```
  BETWEEN min AND max
  ```

  A presença de `NULL` em uma `NOT BETWEEN` declaração `BETWEEN` ou resultará na avaliação da declaração. `NULL`
- `IS NULL` e `IS NOT NULL` os operadores testam se um valor é nulo (indefinido). Usar `NULL` com é `IS NULL` avaliado como verdadeiro.
- EmSQL, um `NULL` valor significa um valor desconhecido.

Funções de comparação

O Timestream for LiveAnalytics suporta as seguintes funções de comparação.

Tópicos

- o melhor ()
- pelo menos ()
- ALL(), ANY () e SOME ()

o melhor ()

A função greatest () retorna o maior dos valores fornecidos. Ele retorna `NULL` se algum dos valores fornecidos for`NULL`. A sintaxe é a seguinte.

```
greatest(value1, value2, ..., valueN)
```

pelo menos ()

A função least () retorna o menor dos valores fornecidos. Ele retorna `NULL` se algum dos valores fornecidos for`NULL`. A sintaxe é a seguinte.

```
least(value1, value2, ..., valueN)
```

ALL(), ANY () e SOME ()

Os `SOME` quantificadores`ALL`, `ANY` e podem ser usados junto com operadores de comparação da seguinte maneira.

Expressão	Significado
A = ALL (...)	É avaliado como verdadeiro quando A é igual a todos os valores.
Um <> ALL (...)	É avaliado como verdadeiro quando A não corresponde a nenhum valor.
A < ALL (...)	É avaliado como verdadeiro quando A é menor que o menor valor.
A = ANY (...)	É avaliado como verdadeiro quando A é igual a qualquer um dos valores.
Um <> ANY (...)	É avaliado como verdadeiro quando A não corresponde a um ou mais valores.
A < ANY (...)	É avaliado como verdadeiro quando A é menor que o maior valor.

Exemplos e notas de uso

> **Note**
>
> Ao usar`ALL`, `ANY` ou`SOME`, a palavra-chave `VALUES` deve ser usada se os valores de comparação forem uma lista de literais.

Exemplo: **`ANY()`**

Um exemplo `ANY()` em uma declaração de consulta é o seguinte.

```
SELECT 11.7 = ANY (VALUES 12.0, 13.5, 11.7)
```

Uma sintaxe alternativa para a mesma operação é a seguinte.

```
SELECT 11.7 = ANY (SELECT 12.0 UNION ALL SELECT 13.5 UNION ALL SELECT 11.7)
```

Nesse caso, `ANY()` avalia como. `True`

Exemplo: **`ALL()`**

Um exemplo `ALL()` em uma declaração de consulta é o seguinte.

```
SELECT 17 < ALL (VALUES 19, 20, 15);
```

Uma sintaxe alternativa para a mesma operação é a seguinte.

```
SELECT 17 < ALL (SELECT 19 UNION ALL SELECT 20 UNION ALL SELECT 15);
```

Nesse caso, `ALL()` avalia como. `False`

Exemplo: **`SOME()`**

Um exemplo `SOME()` em uma declaração de consulta é o seguinte.

```
SELECT 50 >= SOME (VALUES 53, 77, 27);
```

Uma sintaxe alternativa para a mesma operação é a seguinte.

```
SELECT 50 >= SOME (SELECT 53 UNION ALL SELECT 77 UNION ALL SELECT 27);
```

Nesse caso, `SOME()` avalia como. `True`

Expressões condicionais

O Timestream for LiveAnalytics suporta as seguintes expressões condicionais.

Tópicos

- A CASE declaração
- A declaração do IF
- A COALESCE declaração
- A NULLIF declaração
- A TRY declaração

A CASE declaração

A CASEinstrução pesquisa cada expressão de valor da esquerda para a direita até encontrar uma que seja igual`expression`. Se encontrar uma correspondência, o resultado do valor correspondente será retornado. Se nenhuma correspondência for encontrada, o resultado da `ELSE` cláusula será retornado se existir; caso contrário`null`, será retornado. A sintaxe é a seguinte:

```
CASE expression
    WHEN value THEN result
    [ WHEN ... ]
    [ ELSE result ]
END
```

O Timestream também oferece suporte à seguinte sintaxe para declarações. CASE Nessa sintaxe, o formulário “pesquisado” avalia cada condição booleana da esquerda para a direita até que ela exista `true` e retorna o resultado correspondente. Se não houver condições`true`, o resultado da `ELSE` cláusula será retornado se existir; caso contrário`null`, será retornado. Veja abaixo a sintaxe alternativa:

```
CASE
    WHEN condition THEN result
    [ WHEN ... ]
```

```
    [ ELSE result ]
END
```

A declaração do IF

A instrução IF avalia uma condição como verdadeira ou falsa e retorna o valor apropriado. O Timestream suporta as duas representações de sintaxe a seguir para IF:

```
if(condition, true_value)
```

Essa sintaxe avalia e retorna `true_value` se a condição for`true`; caso contrário, é `null` retornada e não `true_value` é avaliada.

```
if(condition, true_value, false_value)
```

Essa sintaxe avalia e retorna `true_value` se a condição for`true`, caso contrário, avalia e retorna. `false_value`

Exemplos

```
SELECT
  if(true, 'example 1'),
  if(false, 'example 2'),
  if(true, 'example 3 true', 'example 3 false'),
  if(false, 'example 4 true', 'example 4 false')
```

_col0	_col1	_col2	_col3
example 1	-	example 3 true	example 4 false
	null		

A COALESCE declaração

COALESCEretorna o primeiro valor não nulo em uma lista de argumentos. A sintaxe é a seguinte:

```
coalesce(value1, value2[,...])
```

A NULLIF declaração

A instrução IF avalia uma condição como verdadeira ou falsa e retorna o valor apropriado. O Timestream suporta as duas representações de sintaxe a seguir para IF:

NULLIFretorna null se for `value1` igual`value2`; caso contrário, ele retornará. `value1` A sintaxe é a seguinte:

```
nullif(value1, value2)
```

A TRY declaração

A TRYfunção avalia uma expressão e manipula certos tipos de erros retornando`null`. A sintaxe é a seguinte:

```
try(expression)
```

Funções de conversão

O Timestream for LiveAnalytics suporta as seguintes funções de conversão.

Tópicos

- cast()
- try_cast ()

cast()

A sintaxe da função de conversão para converter explicitamente um valor como um tipo é a seguinte.

```
cast(value AS type)
```

try_cast ()

O Timestream for LiveAnalytics também suporta a função try_cast, que é semelhante à conversão, mas retorna null se a conversão falhar. A sintaxe é a seguinte.

```
try_cast(value AS type)
```

Operadores matemáticos

O Timestream for LiveAnalytics suporta os seguintes operadores matemáticos.

Operador	Descrição
+	Adição
-	Subtração
*	Multiplicação
/	Divisão (a divisão inteira executa o truncamento)
%	Módulo (restante)

Funções matemáticas

O Timestream for LiveAnalytics suporta as seguintes funções matemáticas.

Função	Tipo de dados de saída	Descrição
abdômen (x)	[o mesmo que a entrada]	Retorna o valor absoluto de x.
cbrt (x)	double	Retorna a raiz cúbica de x.
teto (x) ou teto (x)	[o mesmo que a entrada]	Retorna x arredondado para o número inteiro mais próximo.
graus (x)	double	Converte o ângulo x em radianos em graus.
e ()	double	Retorna o número constante de Euler.
exp (x)	double	Retorna o número de Euler elevado à potência de x.

Função	Tipo de dados de saída	Descrição
piso (x)	[o mesmo que a entrada]	Retorna x arredondado para o número inteiro mais próximo.
from_base (string, radix)	bigint	Retorna o valor da string interpretada como um número de base.
ln (x)	double	Retorna o logaritmo natural de x.
registro 2 (x)	double	Retorna o logaritmo de base 2 de x.
registro 10 (x)	double	Retorna o logaritmo de base 10 de x.
modo (n, m)	[o mesmo que a entrada]	Retorna o módulo (resto) de n dividido por m.
torta (1)	double	Retorna a constante Pi.
pow (x, p) ou potência (x, p)	double	Retorna x elevado à potência de p.
radianos (x)	double	Converte o ângulo x em graus em radianos.
rand () ou random ()	double	Retorna um valor pseudo-al eatório no intervalo 0,0 1,0.
aleatório (n)	[o mesmo que a entrada]	Retorna um número pseudo-al eatório entre 0 e n (exclusivo).
redondo (x)	[o mesmo que a entrada]	Retorna x arredondado para o inteiro mais próximo.
redondo (x, d)	[o mesmo que a entrada]	Retorna x arredondado para d casas decimais.

Função	Tipo de dados de saída	Descrição
sinal (x)	[o mesmo que a entrada]	Retorna a função signum de x, ou seja: • 0 se o argumento for 0 • 1 se o argumento for maior que 0 • -1 se o argumento for menor que 0. Para argumentos duplos, a função também retorna: • NaN se o argumento for NaN • 1 se o argumento for +Infinity • -1 se o argumento for -Infinity.
sqrt (x)	double	Retorna a raiz quadrada de x.
to_base (x, radix)	varchar	Retorna a representação da raiz base de x.
truncar (x)	double	Retorna x arredondado para inteiro soltando dígitos após o ponto decimal.
taco (x)	double	Retorna o arco cosseno de x.
código asiático (x)	double	Retorna o arco seno de x.
Satanás (x)	double	Retorna o arco tangente de x.
satan2 (y, x)	double	Retorna o arco tangente de y/x.

Função	Tipo de dados de saída	Descrição
custo (x)	double	Retorna o cosseno de x.
cash (x)	double	Retorna o cosseno hiperbólico de x.
pecado (x)	double	Retorna o seno de x.
bronzeado (x)	double	Retorna a tangente de x.
tânix (x)	double	Retorna a tangente hiperbólica de x.
infinito ()	double	Retorna a constante que representa o infinito positivo.
é_finito (x)	boolean	Determine se x é finito.
é_infinito (x)	boolean	Determine se x é infinito.
é_nan (x)	boolean	Determine se x é not-a-num ber.
homem (2)	double	Retorna a constante represent ando not-a-number.

Operadores de string

Timestream for LiveAnalytics suporta o `||` operador na concatenação de uma ou mais cadeias de caracteres.

Funções de string

> **Note**
>
> O tipo de dados de entrada dessas funções é considerado varchar, a menos que especificado de outra forma.

Função	Tipo de dados de saída	Descrição
chr (n)	varchar	Retorna o ponto de código Unicode n como um varchar.
ponto de código (x)	inteiro	Retorna o ponto de código Unicode do único caractere de str.
concat (x1,..., xN)	varchar	Retorna a concatenação de x1, x2,..., xN.
distância_de hamming (x1, x2)	bigint	Retorna a distância de Hamming de x1 e x2, ou seja, o número de posições nas quais os caracteres correspondentes são diferente s. Observe que as duas entradas varchar devem ter o mesmo comprimento.
comprimento (x)	bigint	Retorna o comprimento de x em caracteres.
distância de levenshtein (x1, x2)	bigint	Retorna a distância de edição de Levenshtein de x1 e x2, ou seja, o número mínimo de edições de um único caractere (inserções, exclusões ou substituições) necessárias para transformar x1 em x2.
inferior (x)	varchar	Converte x em minúsculas.
carga (x1, tamanho do bit, x2)	varchar	Tecla esquerda x1 para dimensionar caracteres com x2. Se o tamanho for menor que o comprimento de x1,

Função	Tipo de dados de saída	Descrição
		o resultado será truncado para caracteres de tamanho. o tamanho não deve ser negativo e x2 não deve estar vazio.
ltrim (x)	varchar	Remove o espaço em branco inicial de x.
substituir (x1, x2)	varchar	Remove todas as instâncias de x2 de x1.
substituir (x1, x2, x3)	varchar	Substitui todas as instâncias de x2 por x3 em x1.
Reverso (x)	varchar	Retorna x com os caracteres na ordem inversa.
estrada (x1, tamanho grande, x2)	varchar	Pressiona com o botão direito x1 para dimensionar caractere s com x2. Se o tamanho for menor que o comprimento de x1, o resultado será truncado para caracteres de tamanho. o tamanho não deve ser negativo e x2 não deve estar vazio.
guarnição (x)	varchar	Remove o espaço em branco à direita de x.
divisão (x1, x2)	array(varchar)	Divide x1 no delimitador x2 e retorna uma matriz.

Função	Tipo de dados de saída	Descrição
divisão (x1, x2, limite de bigint)	array(varchar)	Divide x1 no delimitador x2 e retorna uma matriz. O último elemento na matriz sempre contém tudo o que resta no limite x1. deve ser um número positivo.
split_part (x1, x2, pós grandes)	varchar	Divide x1 no delimitador x2 e retorna o campo varchar em pos. Os índices de campo começam com 1. Se pos for maior que o número de campos, será retornado null.
tiras (x1, x2)	bigint	Retorna a posição inicial da primeira instância de x2 em x1. As posições começam com 1. Se não for encontrado, 0 será retornado.
strpos (instância x1, x2, bigint)	bigint	Retorna a posição da enésima instância de x2 em x1. A instância deve ser um número positivo. As posições começam com 1. Se não for encontrado, 0 será retornado.
estribos (x1, x2)	bigint	Retorna a posição inicial da última instância de x2 em x1. As posições começam com 1. Se não for encontrado, 0 será retornado.

Função	Tipo de dados de saída	Descrição
strerpos (instância x1, x2, bigint)	bigint	Retorna a posição da enésima instância de x2 em x1 a partir do final de x1. a instância deve ser um número positivo. As posições começam com 1. Se não for encontrado, 0 será retornado.
posição (x2 IN x1)	bigint	Retorna a posição inicial da primeira instância de x2 em x1. As posições começam com 1. Se não for encontrado, 0 será retornado.
substr (x, bigint start)	varchar	Retorna o resto de x do início da posição inicial. As posições começam com 1. Uma posição inicial negativa é interpretada como sendo relativa ao final de x.
substr (x, bigint start, bigint lens)	varchar	Retorna uma substring de x de comprimento len do início da posição inicial. As posições começam com 1. Uma posição inicial negativa é interpretada como sendo relativa ao final de x.
guarnição (x)	varchar	Remove os espaços em branco à esquerda e à direita de x.
superior (x)	varchar	Converte x em maiúsculas.

Operadores de matriz

O Timestream for LiveAnalytics suporta os seguintes operadores de matriz.

Operador	Descrição
[]	Acesse um elemento de uma matriz em que o primeiro índice começa em 1.
||	Concatene uma matriz com outra matriz ou elemento do mesmo tipo.

Funções de array

O Timestream for LiveAnalytics suporta as seguintes funções de matriz.

Função	Tipo de dados de saída	Descrição
array_distinct (x)	array	Remova valores duplicados da matriz x. `SELECT array_distinct(ARRAY[1,2,2,3])` Exemplo de resultado: `[ 1,2,3 ]`
interseção de matriz (x, y)	array	Retorna uma matriz dos elementos na interseção de x e y, sem duplicatas. `SELECT array_intersect(ARRAY[1,2,3], ARRAY[3,4,5])` Exemplo de resultado: `[ 3 ]`

Função	Tipo de dados de saída	Descrição
união_matriz (x, y)	array	Retorna uma matriz dos elementos na união de x e y, sem duplicatas. `SELECT array_union(ARRAY[1,2,3], ARRAY[3,4,5])` Exemplo de resultado: `[ 1,2,3,4,5 ]`
array_except (x, y)	array	Retorna uma matriz de elementos em x, mas não em y, sem duplicatas. `SELECT array_except(ARRAY[1,2,3], ARRAY[3,4,5])` Exemplo de resultado: `[ 1,2 ]`
array_join (x, delimitador, substituição_nula)	varchar	Concatena os elementos da matriz fornecida usando o delimitador e uma string opcional para substituir os nulos. `SELECT array_join(ARRAY[1,2,3], ';', '')` Exemplo de resultado: `1;2;3`

Função	Tipo de dados de saída	Descrição
array_max (x)	igual aos elementos da matriz	Retorna o valor máximo da matriz de entrada. `SELECT array_max (ARRAY[1,2,3])` Exemplo de resultado: 3
array_min (x)	igual aos elementos da matriz	Retorna o valor mínimo da matriz de entrada. `SELECT array_min (ARRAY[1,2,3])` Exemplo de resultado: 1
array_position (x, elemento)	bigint	Retorna a posição da primeira ocorrência do elemento na matriz x (ou 0 se não for encontrada). `SELECT array_position(ARRAY[3,4,5,9], 5)` Exemplo de resultado: 3

Função	Tipo de dados de saída	Descrição
array_remove (x, elemento)	array	Remova todos os elementos iguais ao elemento da matriz x. `SELECT array_remove(ARRAY[3,4,5,9], 4)` Exemplo de resultado: `[ 3,5,9 ]`
ordenação_matriz (x)	array	Classifica e retorna a matriz x. Os elementos de x devem ser ordenáveis. Elementos nulos serão colocados no final da matriz retornada. `SELECT array_sort(ARRAY[6,8,2,9,3])` Exemplo de resultado: `[ 2,3,6,8,9 ]`
sobreposição de matrizes (x, y)	boolean	Testa se as matrizes x e y têm algum elemento não nulo em comum. Retorna null se não houver elementos não nulos em comum, mas qualquer uma das matrizes contiver null. `SELECT arrays_overlap(ARRAY[6,8,2,9,3], ARRAY[6,8])` Exemplo de resultado: `true`

Função	Tipo de dados de saída	Descrição
cardinalidade (x)	bigint	Retorna o tamanho da matriz x. `SELECT cardinality(ARRAY[6,8,2,9,3])` Exemplo de resultado: 5
concat (matriz1, matriz2,..., matrizN)	array	Concatena as matrizes array1, array2,..., arrayN. `SELECT concat(ARRAY[6,8,2,9,3], ARRAY[11,32], ARRAY[6,8,2,0,14])` Exemplo de resultado: `[ 6,8,2,9,3,11,32,6,8,2,0,14 ]`
element_at (matriz (E), índice)	E	Retorna o elemento da matriz em determinado índice. Se o índice for < 0, element_at acessa elementos do último ao primeiro. `SELECT element_at(ARRAY[6,8,2,9,3], 1)` Exemplo de resultado: 6

Função	Tipo de dados de saída	Descrição
repetir (elemento, contagem)	array	Repita o elemento para contar os tempos. `SELECT repeat(1, 3)` Exemplo de resultado: `[ 1,1,1 ]`
reverso (x)	array	Retorna uma matriz que tem a ordem inversa da matriz x. `SELECT reverse(ARRAY[6,8,2,9,3])` Exemplo de resultado: `[ 3,9,2,8,6 ]`
sequência (iniciar, parar)	matriz (bigint)	Gere uma sequência de números inteiros do início ao fim, incrementando em 1 se o início for menor ou igual ao fim, caso contrário -1. `SELECT sequence(3, 8)` Exemplo de resultado: `[ 3,4,5,6,7,8 ]`

Função	Tipo de dados de saída	Descrição
sequência (início, parada, etapa)	matriz (bigint)	Gere uma sequência de números inteiros do início ao fim, incrementando por etapa. `SELECT sequence(3, 15, 2)` Exemplo de resultado: `[ 3,5,7,9,11,13,15 ]`
sequência (iniciar, parar)	matriz (carimbo de data/hora)	Gere uma sequência de registros de data e hora da data de início até a data de término, incrementando em 1 dia. `SELECT sequence( '2023-04-02 19:26:12.941000000', '2023-04-06 19:26:12.941000000', 1d)` Exemplo de resultado: `[ 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 ]`

Função	Tipo de dados de saída	Descrição
sequência (início, parada, etapa)	matriz (carimbo de data/hora)	Gere uma sequência de registros de data e hora do início ao fim, incrementando por etapa. O tipo de dados da etapa é intervalo. `SELECT sequence( '2023-04-02 19:26:12.941000000', '2023-04-10 19:26:12.941000000', 2d)` Exemplo de resultado: `[ 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 ]`
embaralhar (x)	array	Gere uma permutação aleatória da matriz dada x. `SELECT shuffle(ARRAY[6,8,2,9,3])` Exemplo de resultado: `[ 6,3,2,9,8 ]`

Função	Tipo de dados de saída	Descrição
fatia (x, início, comprimento)	array	Subdefine a matriz x começando do início do índice (ou começando do final se o início for negativo) com um comprimento de comprimento. `SELECT slice(ARRAY[6,8,2,9,3], 1, 3)` Exemplo de resultado: `[ 6,8,2 ]`
zip (matriz1, matriz2 [,...])	matriz (linha)	Mescla as matrizes fornecidas, elemento a elemento, em uma única matriz de linhas. Se os argumentos tiverem um comprimento desigual, os valores ausentes serão preenchidos comNULL. `SELECT zip(ARRAY[6,8,2,9,3], ARRAY[15, 24])` Exemplo de resultado: `[ ( 6, 15 ),( 8, 24 ),( 2, - ),( 9, - ),( 3, - ) ]`

Funções bitwise

O Timestream for LiveAnalytics suporta as seguintes funções bit a bit.

Função	Tipo de dados de saída	Descrição
bit_count (bigint, bigint)	bigint (complemento de dois)	Retorna a contagem de bits no primeiro parâmetro bigint em que o segundo parâmetro é um inteiro com sinal de bits, como 8 ou 64. `SELECT bit_count(19, 8)` Exemplo de resultado: 3 `SELECT bit_count(19, 2)` Exemplo de resultado: `Number must be representable with the bits specified. 19 can not be represented with 2 bits`
bitwise_and (bigint, bigint)	bigint (complemento de dois)	Retorna o bit a bit AND dos parâmetros bigint. `SELECT bitwise_and(12, 7)` Exemplo de resultado: 4
bitwise_not (bigint)	bigint (complemento de dois)	Retorna o bit a bit NOT do parâmetro bigint. `SELECT bitwise_not(12)` Exemplo de resultado: -13

Função	Tipo de dados de saída	Descrição
bitwise_or (bigint, bigint)	bigint (complemento de dois)	Retorna o OR bit a bit dos parâmetros bigint. `SELECT bitwise_or(12, 7)` Exemplo de resultado: 15
bitwise_xor (bigint, bigint)	bigint (complemento de dois)	Retorna o bit a bit XOR dos parâmetros bigint. `SELECT bitwise_xor(12, 7)` Exemplo de resultado: 11

Funções de expressões regulares

A expressão regular funciona no Timestream para LiveAnalytics dar suporte à sintaxe do padrão Java. O Timestream for LiveAnalytics suporta as seguintes funções de expressão regular.

Função	Tipo de dados de saída	Descrição
regexp_extract_all (string, padrão)	array(varchar)	Retorna a (s) substring (s) correspondente (s) ao padrão de expressão regular na string. `SELECT regexp_extract_all('example expect complex', 'ex\w')` Exemplo de resultado: `[ exa,exp ]`

Função	Tipo de dados de saída	Descrição
regexp_extract_all (string, padrão, grupo)	array(varchar)	Encontra todas as ocorrências do padrão de expressão regular na string e retorna o grupo numérico do grupo de captura. `SELECT regexp_extract_all('example expect complex', '(ex)(\w)', 2)` Exemplo de resultado: `[ a,p ]`
regexp_extract (string, padrão)	varchar	Retorna a primeira substring correspondente ao padrão de expressão regular na string. `SELECT regexp_extract('example expect', 'ex\w')` Exemplo de resultado: `exa`
regexp_extract (string, padrão, grupo)	varchar	Localiza a primeira ocorrência do padrão de expressão regular na string e retorna o grupo numérico do grupo de captura. `SELECT regexp_extract('example expect', '(ex)(\w)', 2)` Exemplo de resultado: a

Função	Tipo de dados de saída	Descrição
regexp_like (string, padrão)	boolean	Avalia o padrão de expressão regular e determina se ele está contido na string. Essa função é semelhante ao LIKE operador, exceto que o padrão só precisa estar contido na string, em vez de precisar corresponder a toda a string. Em outras palavras, isso executa uma operação de contenção em vez de uma operação de correspondência. Você pode combinar a string inteira ancorando o padrão usando ^ e $. `SELECT regexp_like('example', 'ex')` Exemplo de resultado: `true`
regexp_replace (string, padrão)	varchar	Remove todas as instâncias da substring correspondentes ao padrão de expressão regular da string. `SELECT regexp_replace('example expect', 'expect')` Exemplo de resultado: `example`

Função	Tipo de dados de saída	Descrição
regexp_replace (string, padrão, substituição)	varchar	Substitui todas as instâncias da substring correspondidas pelo padrão regex na string por replace. Os grupos de captura podem ser referenci ados em substituição usando $g para um grupo numerado ou $ {name} para um grupo nomeado. Um cifrão ($) pode ser incluído na substituição escapando-o com uma barra invertida (\ $). `SELECT regexp_re place('example expect', 'expect', 'surprise')` Exemplo de resultado: `example surprise`

Função	Tipo de dados de saída	Descrição
regexp_replace (string, padrão, função)	varchar	Substitui todas as instâncias da substring correspondidas pelo padrão de expressão regular na string usando a função. A função de expressão lambda é invocada para cada correspondência com os grupos de captura passados como uma matriz. A captura de números de grupos começa em um; não há grupo para a partida inteira (se você precisar disso, coloque a expressão inteira entre parênteses). `SELECT regexp_replace('example', '(\w)', x -> upper(x[1]))` Exemplo de resultado: `EXAMPLE`
regexp_split (string, padrão)	array(varchar)	Divide a string usando o padrão de expressão regular e retorna uma matriz. As sequências vazias à direita são preservadas. `SELECT regexp_split('example', 'x')` Exemplo de resultado: `[ e,ample ]`

Operadores de data/hora

> **Note**
>
> O Timestream for LiveAnalytics não suporta valores de tempo negativos. Qualquer operação que resulte em tempo negativo resulta em erro.

O Timestream for LiveAnalytics suporta as seguintes operações em `timestamps`, `dates`, e. `intervals`

Operador	Descrição
+	Adição
-	Subtração

Tópicos

- Operações
- Adição
- Subtração

Operações

O tipo de resultado de uma operação é baseado nos operandos. Literais de intervalo, como `1day` e, `3s` podem ser usados.

```
SELECT date '2022-05-21' + interval '2' day
```

```
SELECT date '2022-05-21' + 2d
```

```
SELECT date '2022-05-21' + 2day
```

Exemplo de resultado para cada um: `2022-05-23`

As unidades de intervalo incluem `second` `minutehour`,`day`,`week`,`month`,, `year` e. Mas, em alguns casos, nem todos são aplicáveis. Por exemplo, segundos, minutos e horas não podem ser adicionados ou subtraídos de uma data.

```
SELECT interval '4' year + interval '2' month
```

Exemplo de resultado: `4-2`

```
SELECT typeof(interval '4' year + interval '2' month)
```

Exemplo de resultado: `interval year to month`

O tipo de resultado das operações de intervalo pode ser `'interval year to month'` ou `'interval day to second'` depender dos operandos. Os intervalos podem ser adicionados ou subtraídos de e. `dates` `timestamps` Mas um `date` ou `timestamp` não pode ser adicionado ou subtraído de um `date` ou. `timestamp` Para encontrar intervalos ou durações relacionados a datas ou carimbos de data/hora, consulte `date_diff` e funções relacionadas em. Intervalo e duração

Adição

Example

```
SELECT date '2022-05-21' + interval '2' day
```

Exemplo de resultado: `2022-05-23`

Example

```
SELECT typeof(date '2022-05-21' + interval '2' day)
```

Exemplo de resultado: `date`

Example

```
SELECT interval '2' year + interval '4' month
```

Exemplo de resultado: `2-4`

Example

```
SELECT typeof(interval '2' year + interval '4' month)
```

Exemplo de resultado: `interval year to month`

Subtração

Example

```
SELECT timestamp '2022-06-17 01:00' - interval '7' hour
```

Exemplo de resultado: `2022-06-16 18:00:00.000000000`

Example

```
SELECT typeof(timestamp '2022-06-17 01:00' - interval '7' hour)
```

Exemplo de resultado: `timestamp`

Example

```
SELECT interval '6' day - interval '4' hour
```

Exemplo de resultado: `5 20:00:00.000000000`

Example

```
SELECT typeof(interval '6' day - interval '4' hour)
```

Exemplo de resultado: `interval day to second`

Funções de data/hora

> **Note**
>
> O Timestream for LiveAnalytics não suporta valores de tempo negativos. Qualquer operação que resulte em tempo negativo resulta em erro.

Timestream for LiveAnalytics usa o UTC fuso horário para data e hora. O Timestream suporta as seguintes funções de data e hora.

Tópicos

- Geral e conversão
- Intervalo e duração
- Formatação e análise
- Extração

Geral e conversão

O Timestream for LiveAnalytics suporta as seguintes funções gerais e de conversão para data e hora.

Função	Tipo de dados de saída	Descrição
data_atual	date	Retorna a data atual emUTC. Nenhum parêntese foi usado. `SELECT current_date` Exemplo de resultado: `2022-07-07` **Note** Essa também é uma palavra-chave reservada. Para obter uma lista de palavras-chave reservadas, consulte Palavras-chave reservadas.
horário_atual	horário	Retorna a hora atual emUTC. Nenhum parêntese foi usado.

Função	Tipo de dados de saída	Descrição
		`SELECT current_time` Exemplo de resultado: `17:41:52.827000000` **Note** Essa também é uma palavra-chave reservada. Para obter uma lista de palavras-chave reservadas, consulte Palavras-chave reservadas.
current_timestamp ou now ()	timestamp	Retorna o timestamp atual em. UTC `SELECT current_timestamp` Exemplo de resultado: `2022-07-07 17:42:32.939000000` **Note** Essa também é uma palavra-chave reservada. Para obter uma lista de palavras-chave reservadas, consulte Palavras-chave reservadas.

Função	Tipo de dados de saída	Descrição
fuso horário_atual ()	varchar O valor será 'UTC.'	O Timestream usa o UTC fuso horário para data e hora. `SELECT current_timezone()` Exemplo de resultado: UTC
data (varchar (x)), data (timestamp)	date	`SELECT date(TIMESTAMP '2022-07-07 17:44:43.771000000')` Exemplo de resultado: `2022-07-07`
último dia do mês (registro de data e hora), último dia do mês (data)	date	`SELECT last_day_of_month(TIMESTAMP '2022-07-07 17:44:43.771000000')` Exemplo de resultado: `2022-07-31`
from_iso8601_timestamp (string)	timestamp	Analisa o carimbo de data/hora ISO 8601 em formato de carimbo de data/hora interno. `SELECT from_iso8601_timestamp('2022-06-17T08:04:05.000000000+05:00')` Exemplo de resultado: `2022-06-17 03:04:05.000000000`

Função	Tipo de dados de saída	Descrição
from_iso8601_date (string)	date	Analisa a string de data ISO 8601 em formato de carimbo de data/hora interno para UTC 00:00:00 da data especificada. `SELECT from_iso8601_date('2022-07-17')` Exemplo de resultado: `2022-07-17`
to_iso8601 (timestamp), to_iso8601 (data)	varchar	Retorna uma string formatada ISO 8601 para a entrada. `SELECT to_iso8601(from_iso8601_date('2022-06-17'))` Exemplo de resultado: `2022-06-17`
from_milliseconds (bigint)	timestamp	`SELECT from_milliseconds(1)` Exemplo de resultado: `1970-01-01 00:00:00.001000000`
from_nanoseconds (bigint)	timestamp	`select from_nanoseconds(300000001)` Exemplo de resultado: `1970-01-01 00:00:00.300000001`

Função	Tipo de dados de saída	Descrição
from_unixtime (duplo)	timestamp	Retorna um timestamp que corresponde ao unixtime fornecido. `SELECT from_unixtime(1)` Exemplo de resultado: `1970-01-01 00:00:01.000000000`
hora local	horário	Retorna a hora atual emUTC. Nenhum parêntese foi usado. `SELECT localtime` Exemplo de resultado: `17:58:22.654000000` **Note** Essa também é uma palavra-chave reservada. Para obter uma lista de palavras-chave reservadas, consulte Palavras-chave reservadas.

Função	Tipo de dados de saída	Descrição
carimbo de data/hora local	timestamp	Retorna o timestamp atual em. UTC Nenhum parêntese foi usado. `SELECT localtimestamp` Exemplo de resultado: `2022-07-07 17:59:04.368000000` **Note** Essa também é uma palavra-chave reservada. Para obter uma lista de palavras-chave reservadas, consulte Palavras-chave reservadas.
to_milissegundos (intervalo de dia a segundo), to_milissegundos (carimbo de data/hora)	bigint	`SELECT to_milliseconds(INTERVAL '2' DAY + INTERVAL '3' HOUR)` Exemplo de resultado: `183600000` `SELECT to_milliseconds(TIMESTAMP '2022-06-17 17:44:43.771000000')` Exemplo de resultado: `1655487883771`

Função	Tipo de dados de saída	Descrição
to_nanoseconds (intervalo de dia a segundo), to_nanose conds (timestamp)	bigint	`SELECT to_nanose conds(INTERVAL '2' DAY + INTERVAL '3' HOUR)` Exemplo de resultado: `183600000000000` `SELECT to_nanose conds(TIMESTAMP '2022-06-17 17:44:43. 771000678')` Exemplo de resultado: `1655487883771000678`
to_unixtime (timestamp)	double	Retorna unixtime para o timestamp fornecido. `SELECT to_unixti me('2022-06-17 17:44:43.771000000')` Exemplo de resultado: `1.6554878837710001E9`

Função	Tipo de dados de saída	Descrição
date_trunc (unidade, timestamp)	timestamp	Retorna o timestamp truncado para a unidade, onde a unidade é [segundo, minuto, hora, dia, semana, mês, trimestre ou ano]. `SELECT date_trunc('minute', TIMESTAMP '2022-06-17 17:44:43.771000000')` Exemplo de resultado: `2022-06-17 17:44:00.000000000`

Intervalo e duração

O Timestream for LiveAnalytics suporta as seguintes funções de intervalo e duração para data e hora.

Função	Tipo de dados de saída	Descrição
date_add (unidade, bigint, data), date_add (unidade, bigint, hora), date_add (varchar (x), bigint, timestamp)	timestamp	Adiciona uma grande quantidade de unidades, em que a unidade é uma de [segundo, minuto, hora, dia, semana, mês, trimestre ou ano]. `SELECT date_add('hour', 9, TIMESTAMP '2022-06-17 00:00:00')`

Função	Tipo de dados de saída	Descrição
		Exemplo de resultado: `2022-06-17 09:00:00.000000000`
date_diff (unidade, data, data), date_diff (unidade, hora, hora), date_diff (unidade, timestamp, timestamp)	bigint	Retorna uma diferença, em que a unidade é uma de [segundo, minuto, hora, dia, semana, mês, trimestre ou ano]. `SELECT date_diff('day', DATE '2020-03-01', DATE '2020-03-02')` Exemplo de resultado: 1
parse_duration (string)	interval	Analisa a string de entrada para retornar um `interval` equivalente. `SELECT parse_duration('42.8ms')` Exemplo de resultado: `0 00:00:00.042800000` `SELECT typeof(parse_duration('42.8ms'))` Exemplo de resultado: `interval day to second`

Função	Tipo de dados de saída	Descrição
compartimento (registro de data e hora, intervalo)	timestamp	Arredonda o valor inteiro do `timestamp` parâmetro para o múltiplo mais próximo do valor inteiro do `interval` parâmetro. O significado desse valor de retorno pode não ser óbvio. Ele é calculado usando aritmética de números inteiros primeiro dividindo o número inteiro do carimbo de data/hora pelo inteiro do intervalo e depois multiplicando o resultado pelo inteiro do intervalo. Lembrando que um carimbo de data/hora especifica um UTC ponto no tempo como o número de frações de segundo decorridas desde a POSIX época (1º de janeiro de 1970), o valor de retorno raramente se alinhará às unidades do calendário. Por exemplo, se você especific ar um intervalo de 30 dias, todos os dias desde a época serão divididos em increment os de 30 dias, e o início do incremento mais recente de 30 dias será retornado, o que não tem relação com os meses do calendário.

Função	Tipo de dados de saída	Descrição
		Veja alguns exemplos: `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`
atrás (intervalo)	timestamp	Retorna o valor correspondente a `interval` current_timestamp. `SELECT ago(1d)` Exemplo de resultado: `2022-07-06 21:08:53.245000000`

Função	Tipo de dados de saída	Descrição
literais de intervalo, como 1h, 1d e 30m	interval	Os literais de intervalo são convenientes para parse_duration (string). Por exemplo, `1d` é o mesmo que `parse_duration('1d')` . Isso permite o uso dos literais sempre que um intervalo é usado. Por exemplo, `ago(1d)` e `bin(<timestamp> , 1m)`.

Alguns literais de intervalo funcionam como uma abreviação de parse_duration. Por exemplo,`parse_duration('1day')`,`1day`,`parse_duration('1d')`, e `1d` cada retorno `1 00:00:00.000000000` onde o tipo está`interval day to second`. O espaço é permitido no formato fornecido para`parse_duration`. Por exemplo, `parse_duration('1day')` também retorna`00:00:00.000000000`. Mas não `1 day` é um intervalo literal.

As unidades relacionadas `interval day to second` são ns, nanosegundo, us, microssegundo, ms, milissegundo, s, segundo, m, minuto, h, hora, d e dia.

Também existe`interval year to month`. As unidades relacionadas ao intervalo ano a mês são y, ano e mês. Por exemplo, `SELECT 1year` devoluções`1-0`. `SELECT 12month`também retorna`1-0`. `SELECT 8month`devoluções`0-8`.

Embora a unidade de também `quarter` esteja disponível para algumas funções, como `date_trunc` e`date_add`, não `quarter` esteja disponível como parte de um intervalo literal.

Formatação e análise

O Timestream for LiveAnalytics suporta as seguintes funções de formatação e análise para data e hora.

Função	Tipo de dados de saída	Descrição
formato_data (timestamp, varchar (x))	varchar	Para obter mais informaçõ es sobre os especificadores

Função	Tipo de dados de saída	Descrição
		de formato usados por essa função, consulte # 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')` Exemplo de resultado: `2019-10-20 10:20:20`
date_parse (varchar (x), varchar (y))	timestamp	Para obter mais informações sobre os especificadores de formato usados por essa função, consulte # 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')` Exemplo de resultado: `2019-10-20 10:20:20.000000000`

Função	Tipo de dados de saída	Descrição
format_datetime (timestamp, varchar (x))	varchar	Para obter mais informaçõ es sobre a string de formato usada por essa função, consulte 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')` Exemplo de resultado: `1968-01-13 12`
parse_datetime (varchar (x), varchar (y))	timestamp	Para obter mais informaçõ es sobre a string de formato usada por essa função, consulte 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')` Exemplo de resultado: `2019-12-29 18:10:00.000000000`

Extração

O Timestream for LiveAnalytics suporta as seguintes funções de extração para data e hora. A função de extração é a base para as demais funções de conveniência.

Função	Tipo de dados de saída	Descrição
extract	bigint	Extrai um campo de um carimbo de data/hora, em que o campo é [YEAR,,QUARTER,MONTH, WEEKDAY, DAY _OF_,MONTH, DAY _OF_,, _OF_ WEEKDOW,,, DAY ou]. YEAR DOY YEAR WEEK YOW HOUR MINUTE SECOND `SELECT extract(YEAR FROM '2019-10-12 23:10:34.000000000')` Exemplo de resultado: 2019
dia (registro de data e hora), dia (data), dia (intervalo de dia a segundo)	bigint	`SELECT day('2019-10-12 23:10:34.000000000')` Exemplo de resultado: 12
day_of_month (timestamp), day_of_month (data), day_of_month (intervalo de dia a segundo)	bigint	`SELECT day_of_month('2019-10-12 23:10:34.000000000')` Exemplo de resultado: 12
day_of_week (timestamp), day_of_week (data)	bigint	`SELECT day_of_week('2019-10-12 23:10:34.000000000')`

Função	Tipo de dados de saída	Descrição
		Exemplo de resultado: 6
day_of_year (timestamp), day_of_year (data)	bigint	`SELECT day_of_year('2019-10-12 23:10:34.000000000')` Exemplo de resultado: 285
down (timestamp), down (data)	bigint	Alias para day_of_week
doy (timestamp), doy (data)	bigint	Alias para day_of_year
hora (registro de data e hora), hora (hora), hora (intervalo de dia a segundo)	bigint	`SELECT hour('2019-10-12 23:10:34.000000000')` Exemplo de resultado: 23
milissegundo (registro de data e hora), milissegundo (hora), milissegundo (intervalo de dia a segundo)	bigint	`SELECT millisecond('2019-10-12 23:10:34.000000000')` Exemplo de resultado: 0
minuto (timestamp), minuto (hora), minuto (intervalo de dia a segundo)	bigint	`SELECT minute('2019-10-12 23:10:34.000000000')` Exemplo de resultado: 10
mês (timestamp), mês (data), mês (intervalo ano a mês)	bigint	`SELECT month('2019-10-12 23:10:34.000000000')` Exemplo de resultado: 10

Função	Tipo de dados de saída	Descrição
nanosegundo (registro de data e hora), nanosegundo (tempo), nanosegundo (intervalo de dia a segundo)	bigint	`SELECT nanosecond(current_timestamp)` Exemplo de resultado: `162000000`
trimestre (registro de data e hora), trimestre (data)	bigint	`SELECT quarter('2019-10-12 23:10:34.000000000')` Exemplo de resultado: `4`
segundo (timestamp), segundo (hora), segundo (intervalo de dia a segundo)	bigint	`SELECT second('2019-10-12 23:10:34.000000000')` Exemplo de resultado: `34`
semana (data e hora), semana (data)	bigint	`SELECT week('2019-10-12 23:10:34.000000000')` Exemplo de resultado: `41`
semana_do_ano (timestamp), week_of_year (data)	bigint	Alias para a semana
ano (timestamp), ano (data), ano (intervalo ano a mês)	bigint	`SELECT year('2019-10-12 23:10:34.000000000')` Exemplo de resultado: `2019`

Função	Tipo de dados de saída	Descrição
ano_da_semana (registro de data e hora), ano_da_semana (data)	bigint	`SELECT year_of_week('2019-10-12 23:10:34.000000000')` Exemplo de resultado: 2019
como (carimbo de data/hora), como (data)	bigint	Alias para year_of_week

Funções agregadas

O Timestream for LiveAnalytics suporta as seguintes funções agregadas.

Função	Tipo de dados de saída	Descrição
arbitrário (x)	[o mesmo que a entrada]	Retorna um valor arbitrário não nulo de x, se existir. `SELECT arbitrary(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemplo de resultado: 1
array_agg (x)	array< [igual à entrada]	Retorna uma matriz criada a partir dos elementos x de entrada. `SELECT array_agg(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemplo de resultado: `[ 1,2,3,4 ]`

Função	Tipo de dados de saída	Descrição
média (x)	double	Retorna a média (média aritmética) de todos os valores de entrada. `SELECT avg(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemplo de resultado: `2.5`
bool_and (boolean) a cada (booleano)	boolean	Retorna TRUE se cada valor de entrada forTRUE, caso contrárioFALSE. `SELECT bool_and(t.c) FROM (VALUES true, true, false, true) AS t(c)` Exemplo de resultado: `false`
bool_or (booleano)	boolean	Retorna TRUE se algum valor de entrada forTRUE, caso contrárioFALSE. `SELECT bool_or(t.c) FROM (VALUES true, true, false, true) AS t(c)` Exemplo de resultado: `true`

Função	Tipo de dados de saída	Descrição
contagem (*) contagem (x)	bigint	count (*) retorna o número de linhas de entrada. count (x) retorna o número de valores de entrada não nulos. `SELECT count(t.c) FROM (VALUES true, true, false, true) AS t(c)` Exemplo de resultado: 4
contar_se (x)	bigint	Retorna o número de valores TRUE de entrada. `SELECT count_if(t.c) FROM (VALUES true, true, false, true) AS t(c)` Exemplo de resultado: 3
média_geométrica (x)	double	Retorna a média geométrica de todos os valores de entrada. `SELECT geometric_mean(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemplo de resultado: 2.213363839400643

Função	Tipo de dados de saída	Descrição
máximo por (x, y)	[o mesmo que x]	Retorna o valor de x associado ao valor máximo de y em todos os valores de entrada. `SELECT max_by(t.c1, t.c2) FROM (VALUES (('a', 1)), (('b', 2)), (('c', 3)), (('d', 4))) AS t(c1, c2)` Exemplo de resultado: d
máximo por (x, y, n)	matriz< [same as x] >	Retorna n valores de x associados ao n maior de todos os valores de entrada de y em ordem decrescente de y. `SELECT max_by(t.c1, t.c2, 2) FROM (VALUES (('a', 1)), (('b', 2)), (('c', 3)), (('d', 4))) AS t(c1, c2)` Exemplo de resultado: `[ d,c ]`

Função	Tipo de dados de saída	Descrição
min_by (x, y)	[o mesmo que x]	Retorna o valor de x associado ao valor mínimo de y em todos os valores de entrada. `SELECT min_by(t.c1, t.c2) FROM (VALUES (('a', 1)), (('b', 2)), (('c', 3)), (('d', 4))) AS t(c1, c2)` Exemplo de resultado: a
min_by (x, y, n)	matriz< [same as x] >	Retorna n valores de x associados ao menor de todos os valores de entrada de y em ordem crescente de y. `SELECT min_by(t.c1, t.c2, 2) FROM (VALUES (('a', 1)), (('b', 2)), (('c', 3)), (('d', 4))) AS t(c1, c2)` Exemplo de resultado: `[ a,b ]`
máximo (x)	[o mesmo que a entrada]	Retorna o valor máximo de todos os valores de entrada. `SELECT max(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemplo de resultado: 4

Função	Tipo de dados de saída	Descrição
máximo (x, n)	matriz< [same as x] >	Retorna n maiores valores de todos os valores de entrada de x. `SELECT max(t.c, 2) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemplo de resultado: `[ 4,3 ]`
minuto (x)	[o mesmo que a entrada]	Retorna o valor mínimo de todos os valores de entrada. `SELECT min(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemplo de resultado: 1
mínimo (x, n)	matriz< [same as x] >	Retorna n menores valores de todos os valores de entrada de x. `SELECT min(t.c, 2) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemplo de resultado: `[ 1,2 ]`

Função	Tipo de dados de saída	Descrição
soma (x)	[o mesmo que a entrada]	Retorna a soma de todos os valores de entrada. `SELECT sum(t.c) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemplo de resultado: `10`
bitwise_e_agg (x)	bigint	Retorna o bit a bit AND de todos os valores de entrada na representação do complemento 2s. `SELECT bitwise_and_agg(t.c) FROM (VALUES 1, -3) AS t(c)` Exemplo de resultado: 1
bitwise_ou_agg (x)	bigint	Retorna o OR bit a bit de todos os valores de entrada na representação do complemento 2s. `SELECT bitwise_or_agg(t.c) FROM (VALUES 1, -3) AS t(c)` Exemplo de resultado: `-3`

Função	Tipo de dados de saída	Descrição
aproximadamente_distinto (x)	bigint	Retorna o número aproximado de valores de entrada distintos. Essa função fornece uma aproximação de count (DISTINCTx). Zero será retornado se todos os valores de entrada forem nulos. Essa função deve produzir um erro padrão de 2,3%, que é o desvio padrão da distribuição de erro (aproximadamente normal) em todos os conjuntos possíveis. Isso não garante um limite superior para o erro para nenhum conjunto de entrada específico. `SELECT approx_distinct(t.c) FROM (VALUES 1, 2, 3, 4, 8) AS t(c)` Exemplo de resultado: 5

Função	Tipo de dados de saída	Descrição
aproximadamente_distinto (x, e)	bigint	Retorna o número aproximado de valores de entrada distintos. Essa função fornece uma aproximação de count (DISTINCTx). Zero será retornado se todos os valores de entrada forem nulos. Essa função deve produzir um erro padrão de não mais que e, que é o desvio padrão da distribuição de erro (aproximadamente normal) em todos os conjuntos possíveis. Isso não garante um limite superior para o erro para nenhum conjunto de entrada específico. A implementação atual dessa função exige que e esteja na faixa de [0,0040625, 0,26000]. `SELECT approx_distinct(t.c, 0.2) FROM (VALUES 1, 2, 3, 4, 8) AS t(c)` Exemplo de resultado: 5

Função	Tipo de dados de saída	Descrição
percentil aproximado (x, porcentagem)	[o mesmo que x]	Retorna o percentil aproximado para todos os valores de entrada de x na porcentagem fornecida. O valor da porcentagem deve estar entre zero e um e deve ser constante para todas as linhas de entrada. `SELECT approx_percentile(t.c, 0.4) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemplo de resultado: 2
percentil aproximado (x, porcentagens)	matriz< [same as x] >	Retorna o percentil aproximado para todos os valores de entrada de x em cada uma das porcentagens especificadas. Cada elemento da matriz de porcentagens deve estar entre zero e um, e a matriz deve ser constante para todas as linhas de entrada. `SELECT approx_percentile(t.c, ARRAY[0.1, 0.8, 0.8]) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemplo de resultado: `[ 1,4,4 ]`

Função	Tipo de dados de saída	Descrição
percentil aproximado (x, w, porcentagem)	[o mesmo que x]	Retorna o percentil ponderado aproximado para todos os valores de entrada de x usando o peso por item w na porcentagem p. O peso deve ser um valor inteiro de pelo menos um. É efetivamente uma contagem de replicação para o valor x no conjunto de percentis. O valor de p deve estar entre zero e um e deve ser constante para todas as linhas de entrada. `SELECT approx_pe rcentile(t.c, 1, 0.1) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemplo de resultado: 1

Função	Tipo de dados de saída	Descrição
percentil aproximado (x, w, porcentagens)	matriz< [same as x] >	Retorna o percentil ponderado aproximado para todos os valores de entrada de x usando o peso por item w em cada uma das porcentagens especificadas na matriz. O peso deve ser um valor inteiro de pelo menos um. É efetivamente uma contagem de replicação para o valor x no conjunto de percentis. Cada elemento da matriz deve estar entre zero e um, e a matriz deve ser constante para todas as linhas de entrada. `SELECT approx_percentile(t.c, 1, ARRAY[0.1, 0.8, 0.8]) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemplo de resultado: `[ 1,4,4 ]`

Função	Tipo de dados de saída	Descrição
percentil aproximado (x, w, porcentagem, precisão)	[o mesmo que x]	Retorna o percentil ponderado aproximado para todos os valores de entrada de x usando o peso por item w na porcentagem p, com um erro máximo de classificação de precisão. O peso deve ser um valor inteiro de pelo menos um. É efetivamente uma contagem de replicação para o valor x no conjunto de percentis. O valor de p deve estar entre zero e um e deve ser constante para todas as linhas de entrada. A precisão deve ser um valor maior que zero e menor que um, e deve ser constante para todas as linhas de entrada. `SELECT approx_percentile(t.c, 1, 0.1, 0.5) FROM (VALUES 1, 2, 3, 4) AS t(c)` Exemplo de resultado: 1

Função	Tipo de dados de saída	Descrição
corr (y, x)	double	Retorna o coeficiente de correlação dos valores de entrada. `SELECT corr(t.c1, t.c2) FROM (VALUES ((1, 1)), ((2, 2)), ((3, 3)), ((4, 4))) AS t(c1, c2)` Exemplo de resultado: `1.0`
covar_pop (y, x)	double	Retorna a covariância populacional dos valores de entrada. `SELECT covar_pop(t.c1, t.c2) FROM (VALUES ((1, 1)), ((2, 2)), ((3, 3)), ((4, 4))) AS t(c1, c2)` Exemplo de resultado: `1.25`
covar_samp (y, x)	double	Retorna a covariância da amostra dos valores de entrada. `SELECT covar_samp(t.c1, t.c2) FROM (VALUES ((1, 1)), ((2, 2)), ((3, 3)), ((4, 4))) AS t(c1, c2)` Exemplo de resultado: `1.6666666666666667`

Função	Tipo de dados de saída	Descrição
regr_interceptar (y, x)	double	Retorna a interceptação de regressão linear dos valores de entrada. y é o valor dependente. x é o valor independente. `SELECT regr_intercept(t.c1, t.c2) FROM (VALUES ((1, 1)), ((2, 2)), ((3, 3)), ((4, 4))) AS t(c1, c2)` Exemplo de resultado: `0.0`
rg_inclinação (y, x)	double	Retorna a inclinação da regressão linear dos valores de entrada. y é o valor dependente. x é o valor independente. `SELECT regr_slope(t.c1, t.c2) FROM (VALUES ((1, 1)), ((2, 2)), ((3, 3)), ((4, 4))) AS t(c1, c2)` Exemplo de resultado: `1.0`

Função	Tipo de dados de saída	Descrição
assimetria (x)	double	Retorna a distorção de todos os valores de entrada. `SELECT skewness(t.c1) FROM (VALUES 1, 2, 3, 4, 8) AS t(c1)` Exemplo de resultado: `0.8978957037987335`
stddev_pop (x)	double	Retorna o desvio padrão da população de todos os valores de entrada. `SELECT stddev_pop(t.c1) FROM (VALUES 1, 2, 3, 4, 8) AS t(c1)` Exemplo de resultado: `2.4166091947189146`
stddev_samp (x) stddev (x)	double	Retorna o desvio padrão da amostra de todos os valores de entrada. `SELECT stddev_sa mp(t.c1) FROM (VALUES 1, 2, 3, 4, 8) AS t(c1)` Exemplo de resultado: `2.701851217221259`

Função	Tipo de dados de saída	Descrição
var_pop (x)	double	Retorna a variância da população de todos os valores de entrada. `SELECT var_pop(t.c1) FROM (VALUES 1, 2, 3, 4, 8) AS t(c1)` Exemplo de resultado: `5.840000000000001`
var_samp (x) variância (x)	double	Retorna a variância da amostra de todos os valores de entrada. `SELECT var_samp(t.c1) FROM (VALUES 1, 2, 3, 4, 8) AS t(c1)` Exemplo de resultado: `7.300000000000001`

Funções de janela

As funções de janela realizam cálculos em todas as linhas do resultado da consulta. Eles são executados após a HAVING cláusula, mas antes da cláusula ORDER BY. A invocação de uma função de janela requer uma sintaxe especial usando a OVER cláusula para especificar a janela. Uma janela tem três componentes:

- A especificação da partição, que separa as linhas de entrada em partições diferentes. Isso é análogo à forma como a cláusula GROUP BY separa as linhas em grupos diferentes para funções agregadas.
- A especificação de ordenação, que determina a ordem na qual as linhas de entrada serão processadas pela função de janela.

- A moldura da janela, que especifica uma janela deslizante de linhas a serem processadas pela função para uma determinada linha. Se o quadro não for especificado, o padrão será RANGE UNBOUNDEDPRECEDING, que é o mesmo que. RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW Esse quadro contém todas as linhas do início da partição até o último par da linha atual.

Todas as funções agregadas podem ser usadas como funções de janela adicionando a OVER cláusula. A função agregada é calculada para cada linha sobre as linhas dentro da moldura da janela da linha atual. Além das funções agregadas, o Timestream for LiveAnalytics suporta as seguintes funções de classificação e valor.

Função	Tipo de dados de saída	Descrição
cume_dist ()	bigint	Retorna a distribuição cumulativa de um valor em um grupo de valores. O resultado é o número de linhas anteriores ou iguais à linha na ordem da janela da partição da janela dividido pelo número total de linhas na partição da janela. Assim, qualquer valor de empate na ordem será avaliado como o mesmo valor de distribuição.
classificação_densa ()	bigint	Retorna a classificação de um valor em um grupo de valores. Isso é semelhante a rank (), exceto que os valores de empate não produzem lacunas na sequência.
até (n)	bigint	Divide as linhas de cada partição de janela em n compartimentos que variam de 1 a no máximo n. Os

Função	Tipo de dados de saída	Descrição
		valores do bucket serão diferentes em no máximo 1. Se o número de linhas na partição não se dividir uniformemente no número de compartimentos, os valores restantes serão distribuí dos um por compartimento, começando pelo primeiro compartimento.
classificação_percentual ()	double	Retorna a classificação percentual de um valor em um grupo de valores. O resultado é (r - 1)/(n - 1) onde r é a classificação () da linha e n é o número total de linhas na partição da janela.
classificação ()	bigint	Retorna a classificação de um valor em um grupo de valores. A classificação é uma mais o número de linhas anteriores à linha que não são iguais à linha. Assim, valores de empate na ordenação produzirão lacunas na sequência. A classific ação é executada para cada partição da janela.

Função	Tipo de dados de saída	Descrição
número_linha ()	bigint	Retorna um número sequencial exclusivo para cada linha, começando com um, de acordo com a ordem das linhas na partição da janela.
primeiro_valor (x)	[o mesmo que a entrada]	Retorna o primeiro valor da janela. Essa função tem como escopo a moldura da janela. A função usa uma expressão ou destino como parâmetro.
último valor (x)	[o mesmo que a entrada]	Retorna o último valor da janela. Essa função tem como escopo a moldura da janela. A função usa uma expressão ou destino como parâmetro.
nth_value (x, deslocamento)	[o mesmo que a entrada]	Retorna o valor no deslocamento especificado desde o início da janela. As compensações começam em 1. O deslocamento pode ser qualquer expressão escalar. Se o deslocamento for nulo ou maior que o número de valores na janela, será retornado nulo. É um erro que o deslocamento seja zero ou negativo. A função usa uma expressão ou destino como seu primeiro parâmetro.

Função	Tipo de dados de saída	Descrição
lead (x [, offset [, valor_padrão]])	[o mesmo que a entrada]	Retorna o valor nas linhas de deslocamento após a linha atual na janela. Os deslocamentos começam em 0, que é a linha atual. O deslocamento pode ser qualquer expressão escalar. O deslocamento padrão é 1. Se o deslocamento for nulo ou maior que a janela, o valor_default_value será retornado ou, se não for especificado, null será retornado. A função usa uma expressão ou destino como seu primeiro parâmetro.
lag (x [, offset [, valor_padrão]])	[o mesmo que a entrada]	Retorna o valor nas linhas de deslocamento antes da linha atual na janela As compensações começam em 0, que é a linha atual. O deslocamento pode ser qualquer expressão escalar. O deslocamento padrão é 1. Se o deslocamento for nulo ou maior que a janela, o valor_default_value será retornado ou, se não for especificado, null será retornado. A função usa uma expressão ou destino como seu primeiro parâmetro.

Consultas de exemplo

Esta seção inclui exemplos de casos de uso da linguagem de consulta LiveAnalytics do Timestream for.

Tópicos

- Consultas simples
- Consultas com funções de série temporal
- Consultas com funções agregadas

Consultas simples

Veja a seguir os 10 pontos de dados adicionados mais recentemente a uma tabela.

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

Veja a seguir os 5 pontos de dados mais antigos de uma medida específica.

```
SELECT * FROM <database_name>.<table_name>
WHERE measure_name = '<measure_name>'
ORDER BY time ASC
LIMIT 5
```

O seguinte funciona com timestamps de granularidade de nanossegundos.

```
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
```

Os valores de medida para registros de várias medidas são identificados pelo nome da coluna. Os valores de medida para registros de medida única são identificados por`measure_value::<data_type>`, onde `<data_type>` está um dos `double`,`bigint`,`boolean`,

ou `varchar` conforme descrito emTipos de dados compatíveis. Para obter mais informações sobre como os valores de medida são modelados, consulte Tabela única versus várias tabelas.

O seguinte recupera valores de uma medida chamada `speed` de registros de várias medidas com um `measure_name` de. `IoTMulti-stats`

```
SELECT speed FROM <database_name>.<table_name> where measure_name = 'IoTMulti-stats'
```

O seguinte recupera `double` valores de registros de medida única com um de`measure_name`. `load`

```
SELECT measure_value::double FROM <database_name>.<table_name> WHERE measure_name =
 'load'
```

Consultas com funções de série temporal

Tópicos

- Exemplo de conjunto de dados e consultas

Exemplo de conjunto de dados e consultas

Você pode usar o Timestream LiveAnalytics para entender e melhorar o desempenho e a disponibilidade de seus serviços e aplicativos. Abaixo está um exemplo de tabela e exemplos de consultas executadas nessa tabela.

A tabela `ec2_metrics` armazena dados de telemetria, como CPU utilização e outras métricas das instâncias. EC2 Você pode ver a tabela abaixo.

Tempo	região	az	Hostname	nome_medida	valor_medida::duplo	valor_medida::bigint
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1a	front-end 01	utilização_da CPU	35,1	nulo
2019-12-04 19:00:00.	us-east-1	us-east-1a	front-end 01	memory_utilization	5.3	nulo

Tempo	região	az	Hostname	nome_medida	valor_medida::duplo	valor_medida::bigint
000 000000						
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1a	front-end 01	network_bytes_in	nulo	1.500
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1a	front-end 01	saída de bytes de rede	nulo	6.700
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1b	front-end 02	utilização_da CPU	38,5	nulo
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1b	front-end 02	memory_utilization	58,4	nulo
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1b	front-end 02	network_bytes_in	nulo	23.000
2019-12-04 19:00:00.000 000000	us-east-1	us-east-1b	front-end 02	saída de bytes de rede	nulo	12.000

Tempo	região	az	Hostname	nome_medida	valor_medida::duplo	valor_medida::bigint
2019-12-04 19:00:00.000000000	us-east-1	us-east-1c	front-end03	utilização_da CPU	45.0	nulo
2019-12-04 19:00:00.000000000	us-east-1	us-east-1c	front-end03	memory_utilization	65,8	nulo
2019-12-04 19:00:00.000000000	us-east-1	us-east-1c	front-end03	network_bytes_in	nulo	15.000
2019-12-04 19:00:00.000000000	us-east-1	us-east-1c	front-end03	saída de bytes de rede	nulo	836.000
2019-12-04 19:00:05.000000000	us-east-1	us-east-1a	front-end01	utilização_da CPU	5.2	nulo
2019-12-04 19:00:05.000000000	us-east-1	us-east-1a	front-end01	memory_utilization	75.0	nulo
2019-12-04 19:00:05.000000000	us-east-1	us-east-1a	front-end01	network_bytes_in	nulo	1.245

Tempo	região	az	Hostname	nome_medida	valor_medida::duplo	valor_medida::bigint
2019-12-04 19:00:05.000000000	us-east-1	us-east-1a	front-end 01	saída de bytes de rede	nulo	68.432
2019-12-04 19:00:08.000000000	us-east-1	us-east-1b	front-end 02	utilização_da CPU	65,6	nulo
2019-12-04 19:00:08.000000000	us-east-1	us-east-1b	front-end 02	memory_utilization	85,3	nulo
2019-12-04 19:00:08.000000000	us-east-1	us-east-1b	front-end 02	network_bytes_in	nulo	1.245
2019-12-04 19:00:08.000000000	us-east-1	us-east-1b	front-end 02	saída de bytes de rede	nulo	68.432
2019-12-04 19:00:20.000000000	us-east-1	us-east-1c	front-end 03	utilização_da CPU	12.1	nulo
2019-12-04 19:00:20.000000000	us-east-1	us-east-1c	front-end 03	memory_utilization	32,0	nulo

Tempo	região	az	Hostname	nome_medida	valor_medida::duplo	valor_medida::bigint
2019-12-04 19:00:20.000000000	us-east-1	us-east-1c	front-end03	network_bytes_in	nulo	1.400
2019-12-04 19:00:20.000000000	us-east-1	us-east-1c	front-end03	saída de bytes de rede	nulo	345
2019-12-04 19:00:10.000000000	us-east-1	us-east-1a	front-end01	utilização_da CPU	15.3	nulo
2019-12-04 19:00:10.000000000	us-east-1	us-east-1a	front-end01	memory_utilization	35,4	nulo
2019-12-04 19:00:10.000000000	us-east-1	us-east-1a	front-end01	network_bytes_in	nulo	23
2019-12-04 19:00:10.000000000	us-east-1	us-east-1a	front-end01	saída de bytes de rede	nulo	0
2019-12-04 19:00:16.000000000	us-east-1	us-east-1b	front-end02	utilização_da CPU	44.0	nulo

Tempo	região	az	Hostname	nome_medida	valor_medida::duplo	valor_medida::bigint
2019-12-04 19:00:16.000 000000	us-east-1	us-east-1b	front-end 02	memory_utilization	64.2	nulo
2019-12-04 19:00:16.000 000000	us-east-1	us-east-1b	front-end 02	network_bytes_in	nulo	1.450
2019-12-04 19:00:16.000 000000	us-east-1	us-east-1b	front-end 02	saída de bytes de rede	nulo	200
2019-12-04 19:00:40.000 000000	us-east-1	us-east-1c	front-end 03	utilização_da CPU	66.4	nulo
2019-12-04 19:00:40.000 000000	us-east-1	us-east-1c	front-end 03	memory_utilization	86,3	nulo
2019-12-04 19:00:40.000 000000	us-east-1	us-east-1c	front-end 03	network_bytes_in	nulo	300
2019-12-04 19:00:40.000 000000	us-east-1	us-east-1c	front-end 03	saída de bytes de rede	nulo	423

Encontre a CPU utilização média de p90, p95 e p99 para um EC2 host específico nas últimas 2 horas:

```
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
```

Identifique EC2 anfitriões com uma CPU utilização maior em 10% ou mais em comparação com a CPU utilização média de toda a frota nas últimas 2 horas:

```
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
```

Encontre a CPU utilização média armazenada em intervalos de 30 segundos para um EC2 host específico nas últimas 2 horas:

```
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
```

Encontre a CPU utilização média armazenada em intervalos de 30 segundos para um EC2 host específico nas últimas 2 horas, preenchendo os valores ausentes usando a interpolação linear:

```
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)
```

Encontre a CPU utilização média armazenada em intervalos de 30 segundos para um EC2 hospedeiro específico nas últimas 2 horas, preenchendo os valores ausentes usando a interpolação com base na última observação realizada:

```
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)
```

Consultas com funções agregadas

Abaixo está um exemplo de conjunto de dados de cenário de IoT para ilustrar consultas com funções agregadas.

Tópicos

- Exemplo de dados
- Consultas de exemplo

Exemplo de dados

O Timestream permite que você armazene e analise dados de sensores de IoT, como localização, consumo de combustível, velocidade e capacidade de carga de uma ou mais frotas de caminhões para permitir o gerenciamento eficaz da frota. Abaixo está o esquema e alguns dos dados de uma tabela iot_trucks que armazena telemetria, como localização, consumo de combustível, velocidade e capacidade de carga dos caminhões.

Tempo	ID do caminhã	Make	Modelo	Frota	capacida e_combu ível	capacida e_carga	nome_m da	valor_me ida::dupl o	valor_med ida::varc har
2019-12-4 19:00:00 000 000000	1234567	GMC	astro	Alpha	100	500	leitura de combusti el	65,2	nulo

Tempo	ID do caminhã	Make	Modelo	Frota	capacidade_combustível	capacidade_carga	nome_medida	valor_medida::duplo	valor_medida::varchar
2019-12-4 19:00:00.000000000	1234567	GMC	astro	Alpha	100	500	balanceamento	400,0	nulo
2019-12-4 19:00:00.000000000	1234567	GMC	astro	Alpha	100	500	velocidade	90,2	nulo
2019-12-4 19:00:00.000000000	1234567	GMC	astro	Alpha	100	500	local	nulo	47,6062 M, 122,3321 W
2019-12-4 19:00:00.000000000	1234567	Kenworth	W900	Alpha	150	1000	leitura de combustível	10.1	nulo
2019-12-4 19:00:00.000000000	1234567	Kenworth	W900	Alpha	150	1000	balanceamento	950,3	nulo

Tempo	ID do caminhã	Make	Modelo	Frota	capacidade_combustível	capacidade_carga	nome_medida	valor_medida::duplo	valor_medida::varchar
2019-12-4 19:00:00.000000000	1234567	Kenworth	W900	Alpha	150	1000	velocidade	50,8	nulo
2019-12-4 19:00:00.000000000	1234567	Kenworth	W900	Alpha	150	1000	local	nulo	40,7128 graus N, 74,0060 graus W

Consultas de exemplo

Obtenha uma lista de todos os atributos e valores dos sensores que estão sendo monitorados para cada caminhão da frota.

```
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
```

Obtenha a leitura de combustível mais recente de cada caminhão da frota nas últimas 24 horas.

```
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
```

Identifique caminhões que funcionaram com pouco combustível (menos de 10%) nas últimas 48 horas:

```
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
)
```

Encontre a carga média e a velocidade máxima de cada caminhão na última semana:

```
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
```

Obtenha a eficiência de carga de cada caminhão na última semana:

```
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
```

APIreferência

Esta seção contém a documentação API de referência do Amazon Timestream.

O Timestream tem doisAPIs: Query e Write.

- O Write API permite que você execute operações como criação de tabelas, marcação de recursos e gravação de registros no Timestream.
- A consulta API permite que você execute operações de consulta.

> **Note**
>
> Ambos APIs incluem a DescribeEndpoints ação. Tanto para Consulta quanto para DescribeEndpoints Gravação, as ações são idênticas.

Você pode ler mais sobre cada um API abaixo, juntamente com tipos de dados, erros e parâmetros comuns.

> **Note**
>
> Para códigos de erro comuns a todos os AWS serviços, consulte a seção AWS Support.

Tópicos

- Ações
- Tipos de dados
- Erros comuns
- Parâmetros gerais

Ações

As seguintes ações são suportadas pelo Amazon Timestream Write:

- CreateBatchLoadTask
- CreateDatabase
- CreateTable
- DeleteDatabase
- DeleteTable
- DescribeBatchLoadTask
- DescribeDatabase
- DescribeEndpoints
- DescribeTable
- ListBatchLoadTasks
- ListDatabases
- ListTables
- ListTagsForResource
- ResumeBatchLoadTask
- TagResource
- UntagResource
- UpdateDatabase

- UpdateTable
- WriteRecords

As seguintes ações são suportadas pelo Amazon Timestream Query:

- CancelQuery
- CreateScheduledQuery
- DeleteScheduledQuery
- DescribeAccountSettings
- DescribeEndpoints
- DescribeScheduledQuery
- ExecuteScheduledQuery
- ListScheduledQueries
- ListTagsForResource
- PrepareQuery
- Query
- TagResource
- UntagResource
- UpdateAccountSettings
- UpdateScheduledQuery

Gravação do Amazon Timestream

As seguintes ações são suportadas pelo Amazon Timestream Write:

- CreateBatchLoadTask
- CreateDatabase
- CreateTable
- DeleteDatabase
- DeleteTable
- DescribeBatchLoadTask
- DescribeDatabase

- DescribeEndpoints
- DescribeTable
- ListBatchLoadTasks
- ListDatabases
- ListTables
- ListTagsForResource
- ResumeBatchLoadTask
- TagResource
- UntagResource
- UpdateDatabase
- UpdateTable
- WriteRecords

CreateBatchLoadTask
Serviço: Amazon Timestream Write

Cria uma nova tarefa de carregamento em lote do Timestream. Uma tarefa de carregamento em lote processa dados de uma CSV fonte em um local do S3 e grava em uma tabela Timestream. Um mapeamento da origem para o destino é definido em uma tarefa de carregamento em lote. Erros e eventos são gravados em um relatório em um local do S3. Para o relatório, se a AWS KMS chave não for especificada, o relatório será criptografado com uma chave gerenciada do S3 quando SSE_S3 for a opção. Caso contrário, um erro será gerado. Para obter mais informações, consulte Chaves gerenciadas pela AWS. Cotas de serviço se aplicam. Para obter detalhes, consulte o exemplo de código.

Sintaxe da Solicitação

```
{
   "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"
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

ClientToken

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Obrigatório: Não

DataModelConfiguration

Tipo: objeto DataModelConfiguration

Obrigatório: Não

DataSourceConfiguration

Define detalhes de configuração sobre a fonte de dados para uma tarefa de carregamento em lote.

Tipo: objeto DataSourceConfiguration

Obrigatório: Sim

RecordVersion

Tipo: longo

Obrigatório: Não

ReportConfiguration

Configuração do relatório para uma tarefa de carregamento em lote. Ele contém detalhes sobre onde os relatórios de erros são armazenados.

Tipo: objeto ReportConfiguration

Obrigatório: Sim

TargetDatabaseName

Banco de dados Timestream de destino para uma tarefa de carregamento em lote.

Tipo: string

Padrão: `[a-zA-Z0-9_.-]+`

Exigido: Sim

TargetTableName

Tabela de Timestream de destino para uma tarefa de carregamento em lote.

Tipo: string

Padrão: `[a-zA-Z0-9_.-]+`

Exigido: Sim

Sintaxe da Resposta

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

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP de 200.

Os dados a seguir são retornados em JSON formato pelo serviço.

TaskId

O ID da tarefa de carregamento em lote.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Comprimento máximo de 32.

Padrão: `[A-Z0-9]+`

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

ConflictException

O Timestream não conseguiu processar essa solicitação porque ela contém um recurso que já existe.

HTTPCódigo de status: 400

InternalServerException

O Timestream não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 500

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ResourceNotFoundException

A operação tentou acessar um recurso inexistente. O recurso pode não estar especificado corretamente ou seu status pode não estarACTIVE.

HTTPCódigo de status: 400

ServiceQuotaExceededException

A cota de instância do recurso foi excedida para essa conta.

HTTPCódigo de status: 400

ThrottlingException

Muitas solicitações foram feitas por um usuário e elas excederam as cotas de serviço. A solicitação foi acelerada.

HTTPCódigo de status: 400

ValidationException

Uma solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

CreateDatabase
Serviço: Amazon Timestream Write

Cria um banco de dados do Timestream. Se a AWS KMS chave não for especificada, o banco de dados será criptografado com uma AWS KMS chave gerenciada do Timestream localizada em sua conta. Para obter mais informações, consulte Chaves gerenciadas pela AWS. Cotas de serviço se aplicam. Para obter detalhes, consulte o exemplo de código.

Sintaxe da Solicitação

```
{
   "DatabaseName": "string",
   "KmsKeyId": "string",
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

DatabaseName

O nome do banco de dados do Timestream.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Tamanho máximo de 256.

Padrão: `[a-zA-Z0-9_.-]+`

Exigido: Sim

KmsKeyId

A AWS KMS chave para o banco de dados. Se a AWS KMS chave não for especificada, o banco de dados será criptografado com uma AWS KMS chave gerenciada do Timestream localizada em sua conta. Para obter mais informações, consulte Chaves gerenciadas pela AWS.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Obrigatório: Não

Tags

Uma lista de pares de valores-chave para rotular a tabela.

Tipo: matriz de objetos Tag

Membros da Matriz: número mínimo de 0 itens. Número máximo de 200 itens.

Obrigatório: Não

Sintaxe da Resposta

```
{
   "Database": {
      "Arn": "string",
      "CreationTime": number,
      "DatabaseName": "string",
      "KmsKeyId": "string",
      "LastUpdatedTime": number,
      "TableCount": number
   }
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP de 200.

Os dados a seguir são retornados em JSON formato pelo serviço.

Database

O banco de dados Timestream recém-criado.

Tipo: objeto Database

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

ConflictException

O Timestream não conseguiu processar essa solicitação porque ela contém um recurso que já existe.

HTTPCódigo de status: 400

InternalServerException

O Timestream não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 500

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ServiceQuotaExceededException

A cota de instância do recurso foi excedida para essa conta.

HTTPCódigo de status: 400

ThrottlingException

Muitas solicitações foram feitas por um usuário e elas excederam as cotas de serviço. A solicitação foi acelerada.

HTTPCódigo de status: 400

ValidationException

Uma solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

CreateTable

Serviço: Amazon Timestream Write

Adiciona uma nova tabela a um banco de dados existente na sua conta. Em um Conta da AWS, os nomes das tabelas devem ser pelo menos exclusivos em cada região se estiverem no mesmo banco de dados. Você pode ter nomes de tabela idênticos na mesma região se as tabelas estiverem em bancos de dados separados. Ao criar a tabela, especifique o nome da tabela, o nome do banco de dados e as propriedades de retenção. Cotas de serviço se aplicam. Consulte a amostra de código para obter detalhes.

Sintaxe da Solicitação

```
{
   "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"
```

```
        }
    ]
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

DatabaseName

O nome do banco de dados do Timestream.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Tamanho máximo de 256.

Padrão: `[a-zA-Z0-9_.-]+`

Exigido: Sim

MagneticStoreWriteProperties

Contém propriedades a serem definidas na tabela ao habilitar gravações de armazenamento magnético.

Tipo: objeto MagneticStoreWriteProperties

Obrigatório: Não

RetentionProperties

A duração pela qual seus dados de séries temporais devem ser armazenados no armazenamento de memória e no armazenamento magnético.

Tipo: objeto RetentionProperties

Obrigatório: Não

Schema

O esquema da tabela.

Tipo: objeto Schema

Obrigatório: Não

TableName

O nome da tabela do Timestream.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Tamanho máximo de 256.

Padrão: `[a-zA-Z0-9_.-]+`

Exigido: Sim

Tags

Uma lista de pares de valores-chave para rotular a tabela.

Tipo: matriz de objetos Tag

Membros da Matriz: número mínimo de 0 itens. Número máximo de 200 itens.

Obrigatório: Não

Sintaxe da Resposta

```
{
   "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"
   }
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP de 200.

Os dados a seguir são retornados em JSON formato pelo serviço.

Table

A tabela Timestream recém-criada.

Tipo: objeto Table

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

ConflictException

O Timestream não conseguiu processar essa solicitação porque ela contém um recurso que já existe.

HTTPCódigo de status: 400

InternalServerException

O Timestream não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 500

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ResourceNotFoundException

A operação tentou acessar um recurso inexistente. O recurso pode não estar especificado corretamente ou seu status pode não estarACTIVE.

HTTPCódigo de status: 400

ServiceQuotaExceededException

A cota de instância do recurso foi excedida para essa conta.

HTTPCódigo de status: 400

ThrottlingException

Muitas solicitações foram feitas por um usuário e elas excederam as cotas de serviço. A solicitação foi acelerada.

HTTPCódigo de status: 400

ValidationException

Uma solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

DeleteDatabase
Serviço: Amazon Timestream Write

Exclui um determinado banco de dados Timestream. Essa é uma operação irreversível. Depois que um banco de dados é excluído, os dados de séries temporais de suas tabelas não podem ser recuperados.

> **Note**
>
> Todas as tabelas no banco de dados devem ser excluídas primeiro, ou um ValidationException erro será gerado.
> Devido à natureza das novas tentativas distribuídas, a operação pode retornar um sucesso ou um ResourceNotFoundException. Os clientes devem considerá-los equivalentes.

Consulte a amostra de código para obter detalhes.

Sintaxe da Solicitação

```
{
   "DatabaseName": "string"
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

DatabaseName

O nome do banco de dados Timestream a ser excluído.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Tamanho máximo de 256.

Obrigatório: Sim

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta de HTTP 200 com o HTTP corpo vazio.

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

InternalServerException

O Timestream não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 500

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ResourceNotFoundException

A operação tentou acessar um recurso inexistente. O recurso pode não estar especificado corretamente ou seu status pode não estarACTIVE.

HTTPCódigo de status: 400

ThrottlingException

Muitas solicitações foram feitas por um usuário e elas excederam as cotas de serviço. A solicitação foi acelerada.

HTTPCódigo de status: 400

ValidationException

Uma solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

DeleteTable
Serviço: Amazon Timestream Write

Exclui uma determinada tabela Timestream. Essa é uma operação irreversível. Depois que uma tabela do banco de dados Timestream é excluída, os dados da série temporal armazenados na tabela não podem ser recuperados.

> **Note**
>
> Devido à natureza das novas tentativas distribuídas, a operação pode retornar um sucesso ou um ResourceNotFoundException. Os clientes devem considerá-los equivalentes.

Consulte a amostra de código para obter detalhes.

Sintaxe da Solicitação

```
{
   "DatabaseName": "string",
   "TableName": "string"
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

DatabaseName

O nome do banco de dados em que o banco de dados Timestream deve ser excluído.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Tamanho máximo de 256.

Obrigatório: Sim

TableName

O nome da tabela Timestream a ser excluída.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Tamanho máximo de 256.

Obrigatório: Sim

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta de HTTP 200 com o HTTP corpo vazio.

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

InternalServerException

O Timestream não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 500

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ResourceNotFoundException

A operação tentou acessar um recurso inexistente. O recurso pode não estar especificado corretamente ou seu status pode não estarACTIVE.

HTTPCódigo de status: 400

ThrottlingException

Muitas solicitações foram feitas por um usuário e elas excederam as cotas de serviço. A solicitação foi acelerada.

HTTPCódigo de status: 400

ValidationException

Uma solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

DescribeBatchLoadTask
Serviço: Amazon Timestream Write

Retorna informações sobre a tarefa de carregamento em lote, incluindo configurações, mapeamentos, progresso e outros detalhes. Cotas de serviço se aplicam. Consulte a amostra de código para obter detalhes.

Sintaxe da Solicitação

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

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

TaskId

O ID da tarefa de carregamento em lote.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Comprimento máximo de 32.

Padrão: `[A-Z0-9]+`

Exigido: Sim

Sintaxe da Resposta

```
{
   "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"
   }
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP de 200.

Os dados a seguir são retornados em JSON formato pelo serviço.

BatchLoadTaskDescription

Descrição da tarefa de carregamento em lote.

Tipo: objeto BatchLoadTaskDescription

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

InternalServerException

O Timestream não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 500

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ResourceNotFoundException

A operação tentou acessar um recurso inexistente. O recurso pode não estar especificado corretamente ou seu status pode não estarACTIVE.

HTTPCódigo de status: 400

ThrottlingException

Muitas solicitações foram feitas por um usuário e elas excederam as cotas de serviço. A solicitação foi acelerada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET

- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

DescribeDatabase
Serviço: Amazon Timestream Write

Retorna informações sobre o banco de dados, incluindo o nome do banco de dados, a hora em que o banco de dados foi criado e o número total de tabelas encontradas no banco de dados. Cotas de serviço se aplicam. Consulte a amostra de código para obter detalhes.

Sintaxe da Solicitação

```
{
   "DatabaseName": "string"
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

DatabaseName

O nome do banco de dados do Timestream.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Tamanho máximo de 256.

Exigido: Sim

Sintaxe da Resposta

```
{
   "Database": {
      "Arn": "string",
      "CreationTime": number,
      "DatabaseName": "string",
      "KmsKeyId": "string",
      "LastUpdatedTime": number,
      "TableCount": number
   }
```

```
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP de 200.

Os dados a seguir são retornados em JSON formato pelo serviço.

Database

O nome da tabela do Timestream.

Tipo: objeto Database

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

InternalServerException

O Timestream não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 500

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ResourceNotFoundException

A operação tentou acessar um recurso inexistente. O recurso pode não estar especificado corretamente ou seu status pode não estarACTIVE.

HTTPCódigo de status: 400

ThrottlingException

Muitas solicitações foram feitas por um usuário e elas excederam as cotas de serviço. A solicitação foi acelerada.

HTTPCódigo de status: 400

ValidationException

Uma solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

DescribeEndpoints
Serviço: Amazon Timestream Write

Retorna uma lista de endpoints disponíveis para fazer chamadas do TimestreamAPI. Essa API operação está disponível por meio da Gravação e da ConsultaAPIs.

Como o SDKs Timestream foi projetado para trabalhar de forma transparente com a arquitetura do serviço, incluindo o gerenciamento e o mapeamento dos endpoints do serviço, não recomendamos que você use essa operação, a menos que: API

- Você está usando VPCendpoints (AWS PrivateLink) com Timestream
- Seu aplicativo usa uma linguagem de programação que ainda não tem SDK suporte
- Você precisa de um melhor controle sobre a implementação do lado do cliente

Para obter informações detalhadas sobre como e quando usar e implementar DescribeEndpoints, consulte The Endpoint Discovery Pattern.

Sintaxe da resposta

```
{
   "Endpoints": [
      {
         "Address": "string",
         "CachePeriodInMinutes": number
      }
   ]
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP de 200.

Os dados a seguir são retornados em JSON formato pelo serviço.

Endpoints

Um `Endpoints` objeto é retornado quando uma `DescribeEndpoints` solicitação é feita.

Tipo: matriz de objetos Endpoint

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

InternalServerException

O Timestream não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 500

ThrottlingException

Muitas solicitações foram feitas por um usuário e elas excederam as cotas de serviço. A solicitação foi acelerada.

HTTPCódigo de status: 400

ValidationException

Uma solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

DescribeTable
Serviço: Amazon Timestream Write

Retorna informações sobre a tabela, incluindo o nome da tabela, nome do banco de dados, duração da retenção do armazenamento de memória e do armazenamento magnético. Cotas de serviço se aplicam. Consulte a amostra de código para obter detalhes.

Sintaxe da Solicitação

```
{
   "DatabaseName": "string",
   "TableName": "string"
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

DatabaseName

O nome do banco de dados do Timestream.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Tamanho máximo de 256.

Obrigatório: Sim

TableName

O nome da tabela do Timestream.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Tamanho máximo de 256.

Exigido: Sim

Sintaxe da Resposta

```
{
```

```
   "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"
   }
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP de 200.

Os dados a seguir são retornados em JSON formato pelo serviço.

Table

A tabela Timestream.

Tipo: objeto Table

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

InternalServerException

O Timestream não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 500

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ResourceNotFoundException

A operação tentou acessar um recurso inexistente. O recurso pode não estar especificado corretamente ou seu status pode não estarACTIVE.

HTTPCódigo de status: 400

ThrottlingException

Muitas solicitações foram feitas por um usuário e elas excederam as cotas de serviço. A solicitação foi acelerada.

HTTPCódigo de status: 400

ValidationException

Uma solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

ListBatchLoadTasks
Serviço: Amazon Timestream Write

Fornece uma lista de tarefas de carregamento em lote, junto com o nome, status, até quando a tarefa pode ser retomada e outros detalhes. Consulte a amostra de código para obter detalhes.

Sintaxe da Solicitação

```
{
   "MaxResults": number,
   "NextToken": "string",
   "TaskStatus": "string"
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

MaxResults

O número total de itens a serem devolvidos na saída. Se o número total de itens disponíveis for maior que o valor especificado, um NextToken será fornecido na saída. Para retomar a paginação, forneça o NextToken valor como argumento de uma invocação subsequenteAPI.

Tipo: número inteiro

Faixa válida: valor mínimo de 1. Valor máximo de 100.

Obrigatório: Não

NextToken

Um token para especificar onde iniciar a paginação. Isso é NextToken de uma resposta previamente truncada.

Tipo: string

Obrigatório: Não

TaskStatus

Status da tarefa de carregamento em lote.

Tipo: string

Valores Válidos: `CREATED | IN_PROGRESS | FAILED | SUCCEEDED | PROGRESS_STOPPED | PENDING_RESUME`

Obrigatório: Não

Sintaxe da Resposta

```
{
   "BatchLoadTasks": [
      {
         "CreationTime": number,
         "DatabaseName": "string",
         "LastUpdatedTime": number,
         "ResumableUntil": number,
         "TableName": "string",
         "TaskId": "string",
         "TaskStatus": "string"
      }
   ],
   "NextToken": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP de 200.

Os dados a seguir são retornados em JSON formato pelo serviço.

BatchLoadTasks

Uma lista de detalhes da tarefa de carregamento em lote.

Tipo: matriz de objetos BatchLoadTask

NextToken

Um token para especificar onde iniciar a paginação. Forneça o próximo ListBatchLoadTasksRequest.

Tipo: string

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

InternalServerException

O Timestream não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 500

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ThrottlingException

Muitas solicitações foram feitas por um usuário e elas excederam as cotas de serviço. A solicitação foi acelerada.

HTTPCódigo de status: 400

ValidationException

Uma solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++

- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

ListDatabases
Serviço: Amazon Timestream Write

Retorna uma lista dos seus bancos de dados Timestream. Cotas de serviço se aplicam. Consulte a amostra de código para obter detalhes.

Sintaxe da Solicitação

```
{
   "MaxResults": number,
   "NextToken": "string"
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

MaxResults

O número total de itens a serem devolvidos na saída. Se o número total de itens disponíveis for maior que o valor especificado, um NextToken será fornecido na saída. Para retomar a paginação, forneça o NextToken valor como argumento de uma invocação subsequenteAPI.

Tipo: número inteiro

Faixa válida: valor mínimo de 1. Valor máximo de 20.

Obrigatório: Não

NextToken

O token de paginação. Para retomar a paginação, forneça o NextToken valor como argumento de uma invocação subsequenteAPI.

Tipo: string

Obrigatório: Não

Sintaxe da Resposta

```
{
   "Databases": [
      {
         "Arn": "string",
         "CreationTime": number,
         "DatabaseName": "string",
         "KmsKeyId": "string",
         "LastUpdatedTime": number,
         "TableCount": number
      }
   ],
   "NextToken": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP de 200.

Os dados a seguir são retornados em JSON formato pelo serviço.

Databases

Uma lista de nomes de bancos de dados.

Tipo: matriz de objetos Database

NextToken

O token de paginação. Esse parâmetro é retornado quando a resposta é truncada.

Tipo: string

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

InternalServerException

O Timestream não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 500

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ThrottlingException

Muitas solicitações foram feitas por um usuário e elas excederam as cotas de serviço. A solicitação foi acelerada.

HTTPCódigo de status: 400

ValidationException

Uma solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

ListTables
Serviço: Amazon Timestream Write

Fornece uma lista de tabelas, junto com o nome, o status e as propriedades de retenção de cada tabela. Consulte a amostra de código para obter detalhes.

Sintaxe da Solicitação

```
{
   "DatabaseName": "string",
   "MaxResults": number,
   "NextToken": "string"
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

DatabaseName

O nome do banco de dados do Timestream.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Tamanho máximo de 256.

Obrigatório: Não

MaxResults

O número total de itens a serem devolvidos na saída. Se o número total de itens disponíveis for maior que o valor especificado, um NextToken será fornecido na saída. Para retomar a paginação, forneça o NextToken valor como argumento de uma invocação subsequenteAPI.

Tipo: número inteiro

Faixa válida: valor mínimo de 1. Valor máximo de 20.

Obrigatório: Não

NextToken

O token de paginação. Para retomar a paginação, forneça o NextToken valor como argumento de uma invocação subsequenteAPI.

Tipo: string

Obrigatório: Não

Sintaxe da Resposta

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

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP de 200.

Os dados a seguir são retornados em JSON formato pelo serviço.

NextToken

Um token para especificar onde iniciar a paginação. Isso é NextToken de uma resposta previamente truncada.

Tipo: string

Tables

Uma lista de tabelas.

Tipo: matriz de objetos Table

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

InternalServerException

O Timestream não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 500

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ResourceNotFoundException

A operação tentou acessar um recurso inexistente. O recurso pode não estar especificado corretamente ou seu status pode não estarACTIVE.

HTTPCódigo de status: 400

ThrottlingException

Muitas solicitações foram feitas por um usuário e elas excederam as cotas de serviço. A solicitação foi acelerada.

HTTPCódigo de status: 400

ValidationException

Uma solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

ListTagsForResource
Serviço: Amazon Timestream Write

Lista todas as tags em um recurso Timestream.

Sintaxe da Solicitação

```
{
   "ResourceARN": "string"
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

ResourceARN

O recurso Timestream com tags a serem listadas. Esse valor é um nome de recurso da Amazon (ARN).

Tipo: string

Restrições de comprimento: tamanho mínimo de 1. Tamanho máximo de 1011.

Exigido: Sim

Sintaxe da Resposta

```
{
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP de 200.

Os dados a seguir são retornados em JSON formato pelo serviço.

Tags

As tags atualmente associadas ao recurso Timestream.

Tipo: matriz de objetos Tag

Membros da Matriz: número mínimo de 0 itens. Número máximo de 200 itens.

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ResourceNotFoundException

A operação tentou acessar um recurso inexistente. O recurso pode não estar especificado corretamente ou seu status pode não estarACTIVE.

HTTPCódigo de status: 400

ThrottlingException

Muitas solicitações foram feitas por um usuário e elas excederam as cotas de serviço. A solicitação foi acelerada.

HTTPCódigo de status: 400

ValidationException

Uma solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

ResumeBatchLoadTask
Serviço: Amazon Timestream Write

Sintaxe da Solicitação

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

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

TaskId

O ID da tarefa de carregamento em lote a ser retomada.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Comprimento máximo de 32.

Padrão: `[A-Z0-9]+`

Exigido: Sim

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta de HTTP 200 com o HTTP corpo vazio.

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

InternalServerException

O Timestream não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 500

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ResourceNotFoundException

A operação tentou acessar um recurso inexistente. O recurso pode não estar especificado corretamente ou seu status pode não estarACTIVE.

HTTPCódigo de status: 400

ThrottlingException

Muitas solicitações foram feitas por um usuário e elas excederam as cotas de serviço. A solicitação foi acelerada.

HTTPCódigo de status: 400

ValidationException

Uma solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2

- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

TagResource
Serviço: Amazon Timestream Write

Associa um conjunto de tags a um recurso Timestream. Em seguida, você pode ativar essas tags definidas pelo usuário para que elas apareçam no console do Billing and Cost Management para acompanhamento da alocação de custos.

Sintaxe da Solicitação

```
{
   "ResourceARN": "string",
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

ResourceARN

Identifica o recurso Timestream ao qual as tags devem ser adicionadas. Esse valor é um nome de recurso da Amazon (ARN).

Tipo: string

Restrições de comprimento: tamanho mínimo de 1. Tamanho máximo de 1011.

Obrigatório: Sim

Tags

As tags a serem atribuídas ao recurso Timestream.

Tipo: matriz de objetos Tag

Membros da Matriz: número mínimo de 0 itens. Número máximo de 200 itens.

Obrigatório: Sim

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta de HTTP 200 com o HTTP corpo vazio.

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ResourceNotFoundException

A operação tentou acessar um recurso inexistente. O recurso pode não estar especificado corretamente ou seu status pode não estarACTIVE.

HTTPCódigo de status: 400

ServiceQuotaExceededException

A cota de instância do recurso foi excedida para essa conta.

HTTPCódigo de status: 400

ThrottlingException

Muitas solicitações foram feitas por um usuário e elas excederam as cotas de serviço. A solicitação foi acelerada.

HTTPCódigo de status: 400

ValidationException

Uma solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

UntagResource
Serviço: Amazon Timestream Write

Remove a associação de tags de um recurso Timestream.

Sintaxe da Solicitação

```
{
   "ResourceARN": "string",
   "TagKeys": [ "string" ]
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

ResourceARN

O recurso Timestream do qual as tags serão removidas. Esse valor é um nome de recurso da Amazon (ARN).

Tipo: string

Restrições de comprimento: tamanho mínimo de 1. Tamanho máximo de 1011.

Obrigatório: Sim

TagKeys

Uma lista de chaves de tags. As tags existentes do recurso cujas chaves são membros dessa lista serão removidas do recurso Timestream.

Tipo: Matriz de strings

Membros da Matriz: número mínimo de 0 itens. Número máximo de 200 itens.

Restrições de tamanho: tamanho mínimo 1. O tamanho máximo é 128.

Obrigatório: Sim

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta de HTTP 200 com o HTTP corpo vazio.

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ResourceNotFoundException

A operação tentou acessar um recurso inexistente. O recurso pode não estar especificado corretamente ou seu status pode não estarACTIVE.

HTTPCódigo de status: 400

ServiceQuotaExceededException

A cota de instância do recurso foi excedida para essa conta.

HTTPCódigo de status: 400

ThrottlingException

Muitas solicitações foram feitas por um usuário e elas excederam as cotas de serviço. A solicitação foi acelerada.

HTTPCódigo de status: 400

ValidationException

Uma solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

UpdateDatabase
Serviço: Amazon Timestream Write

Modifica a AWS KMS chave de um banco de dados existente. Ao atualizar o banco de dados, você deve especificar o nome do banco de dados e o identificador da nova AWS KMS chave a ser usada (`KmsKeyId`). Se houver alguma `UpdateDatabase` solicitação simultânea, o primeiro escritor vence.

Consulte a amostra de código para obter detalhes.

Sintaxe da Solicitação

```
{
   "DatabaseName": "string",
   "KmsKeyId": "string"
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

DatabaseName

O nome do banco de dados.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Tamanho máximo de 256.

Obrigatório: Sim

KmsKeyId

O identificador da nova AWS KMS chave (`KmsKeyId`) a ser usada para criptografar os dados armazenados no banco de dados. Se o cadastrado `KmsKeyId` atualmente `KmsKeyId` no banco de dados for o mesmo da solicitação, não haverá nenhuma atualização.

Você pode especificar o `KmsKeyId` usando qualquer um dos seguintes:

- ID da chave: `1234abcd-12ab-34cd-56ef-1234567890ab`

- ChaveARN: `arn:aws:kms:us-east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`
- Nome do alias: `alias/ExampleAlias`
- PseudônimoARN: `arn:aws:kms:us-east-1:111122223333:alias/ExampleAlias`

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Exigido: Sim

Sintaxe da Resposta

```
{
   "Database": {
      "Arn": "string",
      "CreationTime": number,
      "DatabaseName": "string",
      "KmsKeyId": "string",
      "LastUpdatedTime": number,
      "TableCount": number
   }
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP de 200.

Os dados a seguir são retornados em JSON formato pelo serviço.

Database

Um contêiner de nível superior para uma mesa. Bancos de dados e tabelas são os conceitos fundamentais de gerenciamento no Amazon Timestream. Todas as tabelas em um banco de dados são criptografadas com a mesma AWS KMS chave.

Tipo: objeto Database

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

InternalServerException

O Timestream não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 500

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ResourceNotFoundException

A operação tentou acessar um recurso inexistente. O recurso pode não estar especificado corretamente ou seu status pode não estarACTIVE.

HTTPCódigo de status: 400

ServiceQuotaExceededException

A cota de instância do recurso foi excedida para essa conta.

HTTPCódigo de status: 400

ThrottlingException

Muitas solicitações foram feitas por um usuário e elas excederam as cotas de serviço. A solicitação foi acelerada.

HTTPCódigo de status: 400

ValidationException

Uma solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

UpdateTable

Serviço: Amazon Timestream Write

Modifica a duração da retenção do armazenamento de memória e do armazenamento magnético da sua tabela Timestream. Observe que a alteração na duração da retenção entra em vigor imediatamente. Por exemplo, se o período de retenção do armazenamento de memória foi inicialmente definido para 2 horas e depois alterado para 24 horas, o armazenamento de memória será capaz de armazenar 24 horas de dados, mas será preenchido com 24 horas de dados 22 horas após a alteração ter sido feita. O Timestream não recupera dados do armazenamento magnético para preencher o armazenamento de memória.

Consulte a amostra de código para obter detalhes.

Sintaxe da Solicitação

```
{
   "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"
```

```
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

DatabaseName

O nome do banco de dados do Timestream.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Tamanho máximo de 256.

Obrigatório: Sim

MagneticStoreWriteProperties

Contém propriedades a serem definidas na tabela ao habilitar gravações de armazenamento magnético.

Tipo: objeto MagneticStoreWriteProperties

Obrigatório: Não

RetentionProperties

A duração da retenção do armazenamento de memória e do armazenamento magnético.

Tipo: objeto RetentionProperties

Obrigatório: Não

Schema

O esquema da tabela.

Tipo: objeto Schema

Obrigatório: Não

TableName

O nome da tabela do Timestream.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Tamanho máximo de 256.

Exigido: Sim

Sintaxe da Resposta

```
{
   "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"
    }
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP de 200.

Os dados a seguir são retornados em JSON formato pelo serviço.

Table

A tabela Timestream atualizada.

Tipo: objeto Table

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

InternalServerException

O Timestream não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 500

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ResourceNotFoundException

A operação tentou acessar um recurso inexistente. O recurso pode não estar especificado corretamente ou seu status pode não estarACTIVE.

HTTPCódigo de status: 400

ThrottlingException

Muitas solicitações foram feitas por um usuário e elas excederam as cotas de serviço. A solicitação foi acelerada.

HTTPCódigo de status: 400

ValidationException

Uma solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

WriteRecords
Serviço: Amazon Timestream Write

Permite que você grave seus dados de séries temporais no Timestream. Você pode especificar um único ponto de dados ou um lote de pontos de dados a serem inseridos no sistema. O Timestream oferece um esquema flexível que detecta automaticamente os nomes das colunas e os tipos de dados das tabelas do Timestream com base nos nomes das dimensões e nos tipos de dados dos pontos de dados que você especifica ao invocar gravações no banco de dados.

O Timestream suporta uma eventual semântica de leitura consistente. Isso significa que, quando você consulta dados imediatamente após gravar um lote de dados no Timestream, os resultados da consulta podem não refletir os resultados de uma operação de gravação concluída recentemente. Os resultados também podem incluir alguns dados obsoletos. Se você repetir a solicitação de consulta após um curto período de tempo, os resultados deverão retornar os dados mais recentes. Cotas de serviço se aplicam.

Consulte a amostra de código para obter detalhes.

Surpresas

Você pode usar o `Version` parâmetro em uma `WriteRecords` solicitação para atualizar pontos de dados. O Timestream rastreia um número de versão com cada registro. `Version`o padrão é 1 quando não é especificado para o registro na solicitação. O Timestream atualiza o valor de medida de um registro existente junto com o valor `Version` quando recebe uma solicitação de gravação com um `Version` número maior para esse registro. Quando recebe uma solicitação de atualização em que o valor da medida é o mesmo do registro existente, o Timestream ainda é atualizado`Version`, se for maior que o valor existente de. `Version` Você pode atualizar um ponto de dados quantas vezes quiser, desde que o valor de aumente `Version` continuamente.

Por exemplo, suponha que você grave um novo registro sem indicar `Version` na solicitação. O Timestream armazena esse registro e o define como`Version`. 1 Agora, suponha que você tente atualizar esse registro com uma `WriteRecords` solicitação do mesmo registro com um valor de medida diferente, mas, como antes, não forneça`Version`. Nesse caso, o Timestream rejeitará essa atualização com a, `RejectedRecordsException` já que a versão do registro atualizado não é maior que o valor existente de Versão.

No entanto, se você reenviasse a solicitação de atualização com `Version` definido como2, o Timestream conseguiria atualizar o valor do registro e o `Version` seria definido como. 2 Em seguida, suponha que você tenha enviado uma `WriteRecords` solicitação com esse

mesmo registro e um valor de medida idêntico, mas com `Version` definido como`3`. Nesse caso, o Timestream só seria atualizado para o. `Version` `3` Quaisquer atualizações adicionais precisariam enviar um número de versão maior que`3`, ou as solicitações de atualização receberiam um`RejectedRecordsException`.

Sintaxe da Solicitação

```
{
   "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"
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

CommonAttributes

Um registro que contém os atributos comuns de medida, dimensão, tempo e versão compartilhados entre todos os registros na solicitação. Os atributos de medida e dimensão especificados serão mesclados com os atributos de medida e dimensão no objeto de registros quando os dados forem gravados no Timestream. As dimensões não podem se sobrepor ou podem ser `ValidationException` lançadas. Em outras palavras, um registro deve conter dimensões com nomes exclusivos.

Tipo: objeto Record

Obrigatório: Não

DatabaseName

O nome do banco de dados do Timestream.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Tamanho máximo de 256.

Obrigatório: Sim

Records

Uma matriz de registros que contém os atributos exclusivos de medida, dimensão, tempo e versão para cada ponto de dados de série temporal.

Tipo: Matriz de objetos Record

Membros da matriz: número mínimo de 1 item. Número máximo de 100 itens.

Obrigatório: Sim

TableName

O nome da tabela do Timestream.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Tamanho máximo de 256.

Exigido: Sim

Sintaxe da Resposta

```
{
   "RecordsIngested": {
      "MagneticStore": number,
      "MemoryStore": number,
      "Total": number
   }
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP de 200.

Os dados a seguir são retornados em JSON formato pelo serviço.

RecordsIngested

Informações sobre os registros ingeridos por essa solicitação.

Tipo: objeto RecordsIngested

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

InternalServerException

O Timestream não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 500

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

RejectedRecordsException

WriteRecords lançaria essa exceção nos seguintes casos:

- Registros com dados duplicados em que há vários registros com as mesmas dimensões, carimbos de data/hora e nomes de medidas, mas:
  - Os valores das medidas são diferentes
  - A versão não está presente na solicitação ou o valor da versão no novo registro é igual ou inferior ao valor existente

  Nesse caso, se o Timestream rejeitar dados, o `ExistingVersion` campo na `RejectedRecords` resposta indicará a versão atual do registro. Para forçar uma atualização, você pode reenviar a solicitação com uma versão do conjunto de registros com um valor maior que o. `ExistingVersion`
- Registros com carimbos de data/hora que estão fora da duração de retenção do armazenamento de memória.
- Registros com dimensões ou medidas que excedem os limites definidos pelo Timestream.

Para obter mais informações, consulte Cotas no Amazon Timestream Developer Guide.

HTTPCódigo de status: 400

ResourceNotFoundException

A operação tentou acessar um recurso inexistente. O recurso pode não estar especificado corretamente ou seu status pode não estarACTIVE.

HTTPCódigo de status: 400

ThrottlingException

Muitas solicitações foram feitas por um usuário e elas excederam as cotas de serviço. A solicitação foi acelerada.

HTTPCódigo de status: 400

ValidationException

Uma solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

Consulta do Amazon Timestream

As seguintes ações são suportadas pelo Amazon Timestream Query:

- CancelQuery
- CreateScheduledQuery
- DeleteScheduledQuery
- DescribeAccountSettings
- DescribeEndpoints
- DescribeScheduledQuery
- ExecuteScheduledQuery
- ListScheduledQueries
- ListTagsForResource
- PrepareQuery
- Query
- TagResource
- UntagResource
- UpdateAccountSettings
- UpdateScheduledQuery

CancelQuery
Serviço: Amazon Timestream Query

Cancela uma consulta que foi emitida. O cancelamento é fornecido somente se a consulta não tiver sido concluída antes da emissão da solicitação de cancelamento. Como o cancelamento é uma operação idempotente, as solicitações de cancelamento subsequentes retornarão a`CancellationMessage`, indicando que a consulta já foi cancelada. Consulte a amostra de código para obter detalhes.

Sintaxe da Solicitação

```
{
   "QueryId": "string"
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

QueryId

O ID da consulta que precisa ser cancelada. `QueryID`é retornado como parte do resultado da consulta.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `[a-zA-Z0-9]+`

Exigido: Sim

Sintaxe da Resposta

```
{
   "CancellationMessage": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP de 200.

Os dados a seguir são retornados em JSON formato pelo serviço.

CancellationMessage

A `CancellationMessage` é retornado quando uma `CancelQuery` solicitação para a consulta especificada por `QueryId` já foi emitida.

Tipo: string

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

InternalServerException

O serviço não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 400

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ThrottlingException

A solicitação foi negada devido à limitação da solicitação.

HTTPCódigo de status: 400

ValidationException

Solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

CreateScheduledQuery

Serviço: Amazon Timestream Query

Crie uma consulta programada que será executada em seu nome na programação configurada. O Timestream assume a função de execução fornecida como parte do parâmetro `ScheduledQueryExecutionRoleArn` para executar a consulta. É possível utilizar o parâmetro `NotificationConfiguration` para configurar a notificação das suas operações de consulta programadas.

Sintaxe da Solicitação

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

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

ClientToken

Usar a ClientToken torna a chamada para CreateScheduledQuery idempotente, em outras palavras, fazer a mesma solicitação repetidamente produzirá o mesmo resultado. Fazer várias CreateScheduledQuery solicitações idênticas tem o mesmo efeito que fazer uma única solicitação.

- Se CreateScheduledQuery for chamado sem um`ClientToken`, a consulta SDK gerará um `ClientToken` em seu nome.
- Após 8 horas, qualquer solicitação com o mesmo `ClientToken` será tratada como uma nova solicitação.

Tipo: string

Restrições de comprimento: comprimento mínimo de 32. O tamanho máximo é 128.

Obrigatório: Não

ErrorReportConfiguration

Configuração para relatórios de erros. Relatórios de erros serão gerados quando um problema for encontrado ao gravar resultados de consultas.

Tipo: objeto ErrorReportConfiguration

Obrigatório: Sim

KmsKeyId

A KMS chave da Amazon usada para criptografar o recurso de consulta agendada, em repouso. Se a KMS chave da Amazon não for especificada, o recurso de consulta agendada será criptografado com uma chave Amazon KMS de propriedade da Timestream. Para especificar uma KMS chave, use o ID da chave, a chaveARN, o nome do alias ou o ARN alias. Ao usar um nome de alias, use alias/ como prefixo.

Se for ErrorReportConfiguration usado `SSE_KMS` como tipo de criptografia, o mesmo KmsKeyId será usado para criptografar o relatório de erros em repouso.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Obrigatório: Não

Name

Nome da consulta agendada.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Exigido: Sim

NotificationConfiguration

Configuração de notificação para a consulta programada. Uma notificação é enviada pelo Timestream quando uma execução de consulta é finalizada, quando seu estado é atualizado ou quando ela é excluída.

Tipo: objeto NotificationConfiguration

Obrigatório: Sim

QueryString

A string de consulta a ser executada. Nomes de parâmetros podem ser especificados no caractere `@` da string de consulta, seguido por um identificador. O parâmetro nomeado `@scheduled_runtime` é reservado e pode ser utilizado na consulta para obter o horário em que ela está programada para ser executada.

O timestamp, calculado de acordo com o ScheduleConfiguration parâmetro, será o valor do `@scheduled_runtime` parâmetro para cada execução de consulta. Por exemplo, considere uma instância de uma consulta programada em execução em 2021-12-01 00:00:00. Para esse caso, o parâmetro `@scheduled_runtime` é inicializado no timestamp 2021-12-01 00:00:00 ao chamar a consulta.

Tipo: string

Restrições de comprimento: tamanho mínimo de 1. Tamanho máximo de 262144.

Obrigatório: Sim

ScheduleConfiguration

A configuração do cronograma para a consulta.

Tipo: objeto ScheduleConfiguration

Obrigatório: Sim

ScheduledQueryExecutionRoleArn

O ARN para a IAM função que o Timestream assumirá ao executar a consulta agendada.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Obrigatório: Sim

Tags

Uma lista de pares chave/valor para rotular a consulta programada.

Tipo: matriz de objetos Tag

Membros da Matriz: número mínimo de 0 itens. Número máximo de 200 itens.

Obrigatório: Não

TargetConfiguration

Configuração usada para gravar o resultado de uma consulta.

Tipo: objeto TargetConfiguration

Obrigatório: Não

Sintaxe da Resposta

```
{
   "Arn": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP de 200.

Os dados a seguir são retornados em JSON formato pelo serviço.

Arn

ARNpara a consulta agendada criada.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

ConflictException

Não é possível pesquisar os resultados de uma consulta cancelada.

HTTPCódigo de status: 400

InternalServerException

O serviço não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 400

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ServiceQuotaExceededException

Você excedeu a cota de serviço.

HTTPCódigo de status: 400

ThrottlingException

A solicitação foi negada devido à limitação da solicitação.

HTTPCódigo de status: 400

ValidationException

Solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

DeleteScheduledQuery
Serviço: Amazon Timestream Query

Exclui uma determinada consulta agendada. Essa é uma operação irreversível.

Sintaxe da Solicitação

```
{
   "ScheduledQueryArn": "string"
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

ScheduledQueryArn

O ARN da consulta programada.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Obrigatório: Sim

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta de HTTP 200 com o HTTP corpo vazio.

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

InternalServerException

O serviço não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 400

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ResourceNotFoundException

Não foi possível encontrar o recurso solicitado.

HTTPCódigo de status: 400

ThrottlingException

A solicitação foi negada devido à limitação da solicitação.

HTTPCódigo de status: 400

ValidationException

Solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3

- AWS SDKpara Python
- AWS SDKpara Ruby V3

DescribeAccountSettings
Serviço: Amazon Timestream Query

Descreve as configurações da sua conta que incluem o modelo de preços de consulta e o máximo configurado que TCUs o serviço pode usar para sua carga de trabalho de consulta.

Você é cobrado somente pela duração das unidades computacionais usadas para suas cargas de trabalho.

Sintaxe da resposta

```
{
   "MaxQueryTCU": number,
   "QueryPricingModel": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP de 200.

Os dados a seguir são retornados em JSON formato pelo serviço.

MaxQueryTCU

O número máximo de unidades computacionais do Timestream (TCUs) que o serviço usará a qualquer momento para atender às suas consultas.

Tipo: número inteiro

QueryPricingModel

O modelo de preços para consultas em sua conta.

Tipo: string

Valores Válidos: `BYTES_SCANNED | COMPUTE_UNITS`

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

InternalServerException

O serviço não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 400

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ThrottlingException

A solicitação foi negada devido à limitação da solicitação.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

DescribeEndpoints
Serviço: Amazon Timestream Query

DescribeEndpoints retorna uma lista de endpoints disponíveis para fazer chamadas do TimestreamAPI. Isso API está disponível por meio de Gravação e Consulta.

Como o SDKs Timestream foi projetado para trabalhar de forma transparente com a arquitetura do serviço, incluindo o gerenciamento e o mapeamento dos endpoints do serviço, não é recomendável que você o use, a menos que: API

- Você está usando VPCendpoints (AWS PrivateLink) com Timestream
- Seu aplicativo usa uma linguagem de programação que ainda não tem SDK suporte
- Você precisa de um melhor controle sobre a implementação do lado do cliente

Para obter informações detalhadas sobre como e quando usar e implementar DescribeEndpoints, consulte The Endpoint Discovery Pattern.

Sintaxe da resposta

```
{
   "Endpoints": [
      {
         "Address": "string",
         "CachePeriodInMinutes": number
      }
   ]
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP de 200.

Os dados a seguir são retornados em JSON formato pelo serviço.

Endpoints

Um `Endpoints` objeto é retornado quando uma `DescribeEndpoints` solicitação é feita.

Tipo: matriz de objetos Endpoint

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

InternalServerException

O serviço não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 400

ThrottlingException

A solicitação foi negada devido à limitação da solicitação.

HTTPCódigo de status: 400

ValidationException

Solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

DescribeScheduledQuery
Serviço: Amazon Timestream Query

Fornece informações detalhadas sobre uma consulta agendada.

Sintaxe da Solicitação

```
{
   "ScheduledQueryArn": "string"
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

ScheduledQueryArn

O ARN da consulta programada.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Exigido: Sim

Sintaxe da Resposta

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

```
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP de 200.

Os dados a seguir são retornados em JSON formato pelo serviço.

ScheduledQuery

A consulta agendada.

Tipo: objeto ScheduledQueryDescription

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

InternalServerException

O serviço não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 400

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ResourceNotFoundException

Não foi possível encontrar o recurso solicitado.

HTTPCódigo de status: 400

ThrottlingException

A solicitação foi negada devido à limitação da solicitação.

HTTPCódigo de status: 400

ValidationException

Solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

ExecuteScheduledQuery
Serviço: Amazon Timestream Query

Você pode usar isso API para executar uma consulta agendada manualmente.

Se você ativou`QueryInsights`, isso API também retornará informações e métricas relacionadas à consulta que você executou como parte de uma SNS notificação da Amazon. `QueryInsights`sajuda no ajuste do desempenho de sua consulta. Para obter mais informações sobre`QueryInsights`, consulte Uso de insights de consulta para otimizar consultas no Amazon Timestream.

Sintaxe da Solicitação

```
{
   "ClientToken": "string",
   "InvocationTime": number,
   "QueryInsights": {
      "Mode": "string"
   },
   "ScheduledQueryArn": "string"
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

ClientToken

Não usado.

Tipo: string

Restrições de comprimento: comprimento mínimo de 32. O tamanho máximo é 128.

Obrigatório: Não

InvocationTime

O registro de data e hora é. UTC A consulta será executada como se tivesse sido invocada nesse timestamp.

Tipo: carimbo de data/hora

Obrigatório: Sim

QueryInsights

Encapsula as configurações para ativação. `QueryInsights`

A ativação `QueryInsights` retorna insights e métricas como parte da SNS notificação da Amazon para a consulta que você executou. Você pode usar `QueryInsights` para ajustar o desempenho e o custo de sua consulta.

Tipo: objeto ScheduledQueryInsights

Obrigatório: Não

ScheduledQueryArn

ARNda consulta agendada.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Obrigatório: Sim

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta de HTTP 200 com o HTTP corpo vazio.

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

InternalServerException

O serviço não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 400

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ResourceNotFoundException

Não foi possível encontrar o recurso solicitado.

HTTPCódigo de status: 400

ThrottlingException

A solicitação foi negada devido à limitação da solicitação.

HTTPCódigo de status: 400

ValidationException

Solicitação inválida ou malformada.

HTTPCódigo de status: 400

Exemplos

Mensagem de notificação de consulta agendada para o CONTROL modo ENABLED WITH _ RATE _ _

O exemplo a seguir mostra uma mensagem de notificação de consulta agendada bem-sucedida para o `ENABLED_WITH_RATE_CONTROL` modo do `QueryInsights` parâmetro.

```
"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
        }
    }
}
```

Mensagem de notificação de consulta agendada para o DISABLED modo

O exemplo a seguir mostra uma mensagem de notificação de consulta agendada bem-sucedida para o `DISABLED` modo do `QueryInsights` parâmetro.

```
"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
        }
    }
}
```

Mensagem de notificação de falha para o CONTROL modo ENABLED WITH _ RATE _ _

O exemplo a seguir mostra uma mensagem de notificação de consulta agendada com falha para o `ENABLED_WITH_RATE_CONTROL` modo do `QueryInsights` parâmetro.

```
"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"
    }
}
```

Mensagem de notificação de falha para o DISABLED modo

O exemplo a seguir mostra uma mensagem de notificação de consulta agendada com falha para o `DISABLED` modo do `QueryInsights` parâmetro.

```
"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"
    }
}
```

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2

- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

ListScheduledQueries
Serviço: Amazon Timestream Query

Obtém uma lista de todas as consultas agendadas na conta e região da Amazon do chamador. `ListScheduledQueries`é eventualmente consistente.

Sintaxe da Solicitação

```
{
   "MaxResults": number,
   "NextToken": "string"
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

MaxResults

O número máximo de itens a serem retornados na saída. Se o número total de itens disponíveis for maior que o valor especificado, um `NextToken` será fornecido na saída. Para retomar a paginação, forneça o `NextToken` valor como argumento para a chamada subsequente para. `ListScheduledQueriesRequest`

Tipo: número inteiro

Intervalo válido: valor mínimo de 1. Valor máximo de 1.000.

Obrigatório: Não

NextToken

Um token de paginação para retomar a paginação.

Tipo: string

Obrigatório: Não

Sintaxe da Resposta

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

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP de 200.

Os dados a seguir são retornados em JSON formato pelo serviço.

NextToken

Um token para especificar onde iniciar a paginação. Isso é NextToken de uma resposta previamente truncada.

Tipo: string

ScheduledQueries

Uma lista de consultas agendadas.

Tipo: matriz de objetos ScheduledQuery

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

InternalServerException

O serviço não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 400

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ThrottlingException

A solicitação foi negada devido à limitação da solicitação.

HTTPCódigo de status: 400

ValidationException

Solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

ListTagsForResource
Serviço: Amazon Timestream Query

Liste todas as tags em um recurso de consulta Timestream.

Sintaxe da Solicitação

```
{
   "MaxResults": number,
   "NextToken": "string",
   "ResourceARN": "string"
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

MaxResults

O número máximo de tags a serem retornadas.

Tipo: número inteiro

Faixa válida: valor mínimo de 1. Valor máximo de 200.

Obrigatório: Não

NextToken

Um token de paginação para retomar a paginação.

Tipo: string

Obrigatório: Não

ResourceARN

O recurso Timestream com tags a serem listadas. Esse valor é um nome de recurso da Amazon (ARN).

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Exigido: Sim

Sintaxe da Resposta

```
{
   "NextToken": "string",
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP de 200.

Os dados a seguir são retornados em JSON formato pelo serviço.

NextToken

Um token de paginação para retomar a paginação com uma chamada subsequente para. `ListTagsForResourceResponse`

Tipo: string

Tags

As tags atualmente associadas ao recurso Timestream.

Tipo: matriz de objetos Tag

Membros da Matriz: número mínimo de 0 itens. Número máximo de 200 itens.

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ResourceNotFoundException

Não foi possível encontrar o recurso solicitado.

HTTPCódigo de status: 400

ThrottlingException

A solicitação foi negada devido à limitação da solicitação.

HTTPCódigo de status: 400

ValidationException

Solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

PrepareQuery
Serviço: Amazon Timestream Query

Uma operação síncrona que permite enviar uma consulta com parâmetros a serem armazenados pelo Timestream para execução posterior. O Timestream só suporta o uso desta operação com `ValidateOnly` definido como. `true`

Sintaxe da Solicitação

```
{
   "QueryString": "string",
   "ValidateOnly": boolean
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

QueryString

A string de consulta Timestream que você deseja usar como uma instrução preparada. Nomes de parâmetros podem ser especificados no caractere @ da string de consulta, seguido por um identificador.

Tipo: string

Restrições de comprimento: tamanho mínimo de 1. Tamanho máximo de 262144.

Obrigatório: Sim

ValidateOnly

Ao definir esse valor como`true`, o Timestream só validará se a string de consulta é uma consulta Timestream válida e não armazenará a consulta preparada para uso posterior.

Tipo: booliano

Obrigatório: não

Sintaxe da Resposta

```
{
   "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"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP de 200.

Os dados a seguir são retornados em JSON formato pelo serviço.

Columns

Uma lista de colunas de SELECT cláusulas da string de consulta enviada.

Tipo: matriz de objetos SelectColumn

Parameters

Uma lista de parâmetros usados na sequência de caracteres de consulta enviada.

Tipo: matriz de objetos ParameterMapping

QueryString

A sequência de caracteres de consulta que você deseja preparar.

Tipo: string

Restrições de comprimento: tamanho mínimo de 1. Tamanho máximo de 262144.

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

InternalServerException

O serviço não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 400

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ThrottlingException

A solicitação foi negada devido à limitação da solicitação.

HTTPCódigo de status: 400

ValidationException

Solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

Query
Serviço: Amazon Timestream Query

`Query`é uma operação síncrona que permite que você execute uma consulta em seus dados do Amazon Timestream.

Se você ativou`QueryInsights`, isso API também retornará informações e métricas relacionadas à consulta que você executou. `QueryInsights`ajuda no ajuste do desempenho de sua consulta. Para obter mais informações sobre`QueryInsights`, consulte Uso de insights de consulta para otimizar consultas no Amazon Timestream.

> **Note**
>
> O número máximo de `Query` API solicitações que você pode fazer com `QueryInsights` ativado é de 1 consulta por segundo (QPS). Se você exceder essa taxa de consulta, isso poderá resultar em limitação.

`Query`expirará após 60 segundos. Você deve atualizar o tempo limite padrão no SDK para suportar um tempo limite de 60 segundos. Consulte o exemplo de código para obter detalhes.

Sua solicitação de consulta falhará nos seguintes casos:

- Se você enviar uma `Query` solicitação com o mesmo token de cliente fora da janela de idempotência de 5 minutos.
- Se você enviar uma `Query` solicitação com o mesmo token de cliente, mas alterar outros parâmetros, dentro da janela de idempotência de 5 minutos.
- Se o tamanho da linha (incluindo os metadados da consulta) exceder 1 MB, a consulta falhará com a seguinte mensagem de erro:

  ```
  Query aborted as max page response size has been exceeded by the output result row
  ```

- Se o IAM principal do iniciador da consulta e do leitor de resultados não forem os mesmos e/ou o iniciador da consulta e o leitor de resultados não tiverem a mesma string de consulta nas solicitações de consulta, a consulta falhará com um `Invalid pagination token` erro.

Sintaxe da Solicitação

```
{
   "ClientToken": "string",
   "MaxRows": number,
   "NextToken": "string",
   "QueryInsights": {
      "Mode": "string"
   },
   "QueryString": "string"
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

ClientToken

Cadeia de caracteres exclusiva, com distinção entre maiúsculas e minúsculas, de até 64 ASCII caracteres, especificada quando uma `Query` solicitação é feita. Fornecer um `ClientToken` faz com que a chamada seja `Query` idempotente. Isso significa que executar a mesma consulta repetidamente produzirá o mesmo resultado. Em outras palavras, fazer várias `Query` solicitações idênticas tem o mesmo efeito que fazer uma única solicitação. Ao usar `ClientToken` em uma consulta, observe o seguinte:

- Se a consulta API for instanciada sem um`ClientToken`, a consulta SDK gerará um `ClientToken` em seu nome.
- Se a `Query` invocação contiver apenas o, `ClientToken` mas não incluir um`NextToken`, essa invocação de será `Query` considerada uma nova execução de consulta.
- Se a invocação contiver`NextToken`, essa invocação específica será considerada uma invocação subsequente de uma chamada anterior para a Consulta e um conjunto de resultados será API retornado.
- Depois de 4 horas, qualquer solicitação com a mesma `ClientToken` é tratada como uma nova solicitação.

Tipo: string

Restrições de comprimento: comprimento mínimo de 32. O tamanho máximo é 128.

Obrigatório: Não

MaxRows

O número total de linhas a serem retornadas na `Query` saída. A execução inicial de `Query` com um `MaxRows` valor especificado retornará o conjunto de resultados da consulta em dois casos:

- O tamanho do resultado é menor que`1MB`.
- O número de linhas no conjunto de resultados é menor que o valor de`maxRows`.

Caso contrário, a invocação inicial de retorna `Query` apenas a`NextToken`, que pode ser usada em chamadas subsequentes para buscar o conjunto de resultados. Para retomar a paginação, forneça o `NextToken` valor no comando subsequente.

Se o tamanho da linha for grande (por exemplo, uma linha tem muitas colunas), o Timestream pode retornar menos linhas para evitar que o tamanho da resposta exceda o limite de 1 MB. Se não `MaxRows` for fornecido, o Timestream enviará o número necessário de linhas para atingir o limite de 1 MB.

Tipo: número inteiro

Intervalo válido: valor mínimo de 1. Valor máximo de 1.000.

Obrigatório: Não

NextToken

Um token de paginação usado para retornar um conjunto de resultados. Quando o `Query` API é invocado usando`NextToken`, essa invocação específica é considerada uma invocação subsequente de uma chamada anterior para`Query`, e um conjunto de resultados é retornado. No entanto, se a `Query` invocação contiver apenas o`ClientToken`, essa invocação de será `Query` considerada uma nova consulta executada.

Observe o seguinte ao usar NextToken em uma consulta:

- Um token de paginação pode ser usado para até cinco `Query` invocações OU por uma duração de até 1 hora, o que ocorrer primeiro.
- Usar o mesmo `NextToken` retornará o mesmo conjunto de registros. Para continuar paginando o conjunto de resultados, você deve usar o mais recente. `nextToken`
- Suponha que uma `Query` invocação retorne dois `NextToken` valores, e. `TokenA` `TokenB` Se `TokenB` for usado em uma `Query` invocação subsequente, será invalidado e `TokenA` não poderá ser reutilizado.

- Para solicitar um conjunto de resultados anterior de uma consulta após o início da paginação, você deve invocar novamente a consulta. API
- O último `NextToken` deve ser usado para paginar até `null` ser retornado, momento em que um novo `NextToken` deve ser usado.
- Se o IAM principal do iniciador da consulta e do leitor de resultados não forem os mesmos e/ou o iniciador da consulta e o leitor de resultados não tiverem a mesma string de consulta nas solicitações de consulta, a consulta falhará com um `Invalid pagination token` erro.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Obrigatório: Não

QueryInsights

Encapsula as configurações para ativação. `QueryInsights`

A ativação `QueryInsights` retorna insights e métricas, além dos resultados da consulta que você executou. Você pode usar `QueryInsights` para ajustar o desempenho da consulta.

Tipo: objeto QueryInsights

Obrigatório: Não

QueryString

A consulta a ser executada pelo Timestream.

Tipo: string

Restrições de comprimento: tamanho mínimo de 1. Tamanho máximo de 262144.

Exigido: Sim

Sintaxe da Resposta

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

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP de 200.

Os dados a seguir são retornados em JSON formato pelo serviço.

ColumnInfo

Os tipos de dados da coluna do conjunto de resultados retornado.

Tipo: matriz de objetos ColumnInfo

NextToken

Um token de paginação que pode ser usado novamente em uma `Query` chamada para obter o próximo conjunto de resultados.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

QueryId

Um ID exclusivo para a consulta em questão.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `[a-zA-Z0-9]+`

QueryInsightsResponse

Encapsulamentos `QueryInsights` contendo insights e métricas relacionadas à consulta que você executou.

Tipo: objeto QueryInsightsResponse

QueryStatus

Informações sobre o status da consulta, incluindo progresso e bytes verificados.

Tipo: objeto QueryStatus

Rows

As linhas do conjunto de resultados retornadas pela consulta.

Tipo: matriz de objetos Row

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

ConflictException

Não é possível pesquisar os resultados de uma consulta cancelada.

HTTPCódigo de status: 400

InternalServerException

O serviço não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 400

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

QueryExecutionException

O Timestream não conseguiu executar a consulta com êxito.

HTTPCódigo de status: 400

ThrottlingException

A solicitação foi negada devido à limitação da solicitação.

HTTPCódigo de status: 400

ValidationException

Solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

TagResource
Serviço: Amazon Timestream Query

Associe um conjunto de tags a um recurso Timestream. Em seguida, você pode ativar essas tags definidas pelo usuário para que elas apareçam no console do Billing and Cost Management para acompanhamento da alocação de custos.

Sintaxe da Solicitação

```
{
   "ResourceARN": "string",
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

ResourceARN

Identifica o recurso Timestream ao qual as tags devem ser adicionadas. Esse valor é um nome de recurso da Amazon (ARN).

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Obrigatório: Sim

Tags

As tags a serem atribuídas ao recurso Timestream.

Tipo: matriz de objetos Tag

Membros da Matriz: número mínimo de 0 itens. Número máximo de 200 itens.

Obrigatório: Sim

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta de HTTP 200 com o HTTP corpo vazio.

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ResourceNotFoundException

Não foi possível encontrar o recurso solicitado.

HTTPCódigo de status: 400

ServiceQuotaExceededException

Você excedeu a cota de serviço.

HTTPCódigo de status: 400

ThrottlingException

A solicitação foi negada devido à limitação da solicitação.

HTTPCódigo de status: 400

ValidationException

Solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

UntagResource
Serviço: Amazon Timestream Query

Remove a associação de tags de um recurso de consulta Timestream.

Sintaxe da Solicitação

```
{
   "ResourceARN": "string",
   "TagKeys": [ "string" ]
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

ResourceARN

O recurso Timestream do qual as tags serão removidas. Esse valor é um nome de recurso da Amazon (ARN).

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Obrigatório: Sim

TagKeys

Uma lista de chaves de tags. As tags existentes do recurso cujas chaves são membros dessa lista serão removidas do recurso Timestream.

Tipo: Matriz de strings

Membros da Matriz: número mínimo de 0 itens. Número máximo de 200 itens.

Restrições de tamanho: tamanho mínimo 1. O tamanho máximo é 128.

Obrigatório: Sim

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta de HTTP 200 com o HTTP corpo vazio.

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ResourceNotFoundException

Não foi possível encontrar o recurso solicitado.

HTTPCódigo de status: 400

ThrottlingException

A solicitação foi negada devido à limitação da solicitação.

HTTPCódigo de status: 400

ValidationException

Solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2

- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

UpdateAccountSettings
Serviço: Amazon Timestream Query

Faz a transição da sua conta TCUs para uso nos preços de consulta e modifica o máximo de unidades computacionais de consulta que você configurou. Se você reduzir o valor de `MaxQueryTCU` para a configuração desejada, o novo valor poderá levar até 24 horas para entrar em vigor.

> **Note**
>
> Depois de fazer a transição de sua conta TCUs para uso nos preços de consulta, você não pode fazer a transição para o uso de bytes escaneados para preços de consulta.

Sintaxe da Solicitação

```
{
   "MaxQueryTCU": number,
   "QueryPricingModel": "string"
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

MaxQueryTCU

O número máximo de unidades de computação que o serviço usará em qualquer momento para atender às suas consultas. Para executar consultas, você deve definir uma capacidade mínima de 4TCU. Você pode definir o número máximo de TCU em múltiplos de 4, por exemplo, 4, 8, 16, 32 e assim por diante.

O valor máximo suportado `MaxQueryTCU` é 1000. Para solicitar um aumento desse limite flexível, entre em contato com o AWS Support. Para obter informações sobre a cota padrão para maxQueryTCU, consulte Cotas padrão.

Tipo: número inteiro

Obrigatório: não

QueryPricingModel

O modelo de preços para consultas em uma conta.

> **Note**
>
> O `QueryPricingModel` parâmetro é usado por várias operações de Timestream; no entanto, a `UpdateAccountSettings` API operação não reconhece nenhum valor além de. `COMPUTE_UNITS`

Tipo: string

Valores Válidos: `BYTES_SCANNED | COMPUTE_UNITS`

Obrigatório: Não

Sintaxe da Resposta

```
{
   "MaxQueryTCU": number,
   "QueryPricingModel": "string"
}
```

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta HTTP de 200.

Os dados a seguir são retornados em JSON formato pelo serviço.

MaxQueryTCU

O número máximo configurado de unidades computacionais que o serviço usará em qualquer momento para atender às suas consultas.

Tipo: número inteiro

QueryPricingModel

O modelo de preços de uma conta.

Tipo: string

Valores Válidos: `BYTES_SCANNED | COMPUTE_UNITS`

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

InternalServerException

O serviço não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 400

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ThrottlingException

A solicitação foi negada devido à limitação da solicitação.

HTTPCódigo de status: 400

ValidationException

Solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET

- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

UpdateScheduledQuery
Serviço: Amazon Timestream Query

Atualize uma consulta agendada.

Sintaxe da Solicitação

```
{
   "ScheduledQueryArn": "string",
   "State": "string"
}
```

Parâmetros da solicitação

Para obter informações sobre os parâmetros comuns a todas as ações, consulte Parâmetros Comuns.

A solicitação aceita os seguintes dados no JSON formato.

ScheduledQueryArn

ARNda consulta agendada.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Obrigatório: Sim

State

Estado da consulta agendada.

Tipo: string

Valores Válidos: `ENABLED | DISABLED`

Obrigatório: Sim

Elementos de Resposta

Se a ação for bem-sucedida, o serviço retornará uma resposta de HTTP 200 com o HTTP corpo vazio.

Erros

Para obter informações sobre os erros comuns retornados pelas ações, consulte Erros comuns.

AccessDeniedException

Você não está autorizado a realizar essa ação.

HTTPCódigo de status: 400

InternalServerException

O serviço não conseguiu processar totalmente essa solicitação devido a um erro interno do servidor.

HTTPCódigo de status: 400

InvalidEndpointException

O endpoint solicitado não era válido.

HTTPCódigo de status: 400

ResourceNotFoundException

Não foi possível encontrar o recurso solicitado.

HTTPCódigo de status: 400

ThrottlingException

A solicitação foi negada devido à limitação da solicitação.

HTTPCódigo de status: 400

ValidationException

Solicitação inválida ou malformada.

HTTPCódigo de status: 400

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS Command Line Interface
- AWS SDKpara. NET
- AWS SDKpara C++
- AWS SDKpara Go v2
- AWS SDKpara Java V2
- AWS SDKpara JavaScript V3
- AWS SDKpara PHP V3
- AWS SDKpara Python
- AWS SDKpara Ruby V3

Tipos de dados

Os seguintes tipos de dados são compatíveis com o Amazon Timestream Write:

- BatchLoadProgressReport
- BatchLoadTask
- BatchLoadTaskDescription
- CsvConfiguration
- Database
- DataModel
- DataModelConfiguration
- DataModelS3Configuration
- DataSourceConfiguration
- DataSourceS3Configuration
- Dimension
- DimensionMapping
- Endpoint
- MagneticStoreRejectedDataLocation
- MagneticStoreWriteProperties
- MeasureValue
- MixedMeasureMapping
- MultiMeasureAttributeMapping

- MultiMeasureMappings
- PartitionKey
- Record
- RecordsIngested
- RejectedRecord
- ReportConfiguration
- ReportS3Configuration
- RetentionProperties
- S3Configuration
- Schema
- Table
- Tag

Os seguintes tipos de dados são compatíveis com o Amazon Timestream Query:

- ColumnInfo
- Datum
- DimensionMapping
- Endpoint
- ErrorReportConfiguration
- ErrorReportLocation
- ExecutionStats
- MixedMeasureMapping
- MultiMeasureAttributeMapping
- MultiMeasureMappings
- NotificationConfiguration
- ParameterMapping
- QueryInsights
- QueryInsightsResponse
- QuerySpatialCoverage
- QuerySpatialCoverageMax

- QueryStatus
- QueryTemporalRange
- QueryTemporalRangeMax
- Row
- S3Configuration
- S3ReportLocation
- ScheduleConfiguration
- ScheduledQuery
- ScheduledQueryDescription
- ScheduledQueryInsights
- ScheduledQueryInsightsResponse
- ScheduledQueryRunSummary
- SelectColumn
- SnsConfiguration
- Tag
- TargetConfiguration
- TargetDestination
- TimeSeriesDataPoint
- TimestreamConfiguration
- TimestreamDestination
- Type

Gravação do Amazon Timestream

Os seguintes tipos de dados são compatíveis com o Amazon Timestream Write:

- BatchLoadProgressReport
- BatchLoadTask
- BatchLoadTaskDescription
- CsvConfiguration
- Database
- DataModel

- DataModelConfiguration
- DataModelS3Configuration
- DataSourceConfiguration
- DataSourceS3Configuration
- Dimension
- DimensionMapping
- Endpoint
- MagneticStoreRejectedDataLocation
- MagneticStoreWriteProperties
- MeasureValue
- MixedMeasureMapping
- MultiMeasureAttributeMapping
- MultiMeasureMappings
- PartitionKey
- Record
- RecordsIngested
- RejectedRecord
- ReportConfiguration
- ReportS3Configuration
- RetentionProperties
- S3Configuration
- Schema
- Table
- Tag

BatchLoadProgressReport
Serviço: Amazon Timestream Write

Detalhes sobre o progresso de uma tarefa de carregamento em lote.

Conteúdo

BytesMetered

Tipo: longo

Obrigatório: Não

FileFailures

Tipo: longo

Obrigatório: Não

ParseFailures

Tipo: longo

Obrigatório: Não

RecordIngestionFailures

Tipo: longo

Obrigatório: Não

RecordsIngested

Tipo: longo

Obrigatório: Não

RecordsProcessed

Tipo: longo

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

BatchLoadTask
Serviço: Amazon Timestream Write

Detalhes sobre uma tarefa de carregamento em lote.

Conteúdo

CreationTime

A hora em que a tarefa de carregamento em lote do Timestream foi criada.

Tipo: carimbo de data/hora

Obrigatório: Não

DatabaseName

Nome do banco de dados para o banco de dados no qual uma tarefa de carregamento em lote carrega dados.

Tipo: string

Obrigatório: Não

LastUpdatedTime

A hora em que a tarefa de carregamento em lote do Timestream foi atualizada pela última vez.

Tipo: carimbo de data/hora

Obrigatório: Não

ResumableUntil

Tipo: carimbo de data/hora

Obrigatório: Não

TableName

Nome da tabela na qual uma tarefa de carregamento em lote carrega dados.

Tipo: string

Obrigatório: Não

TaskId

O ID da tarefa de carregamento em lote.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Comprimento máximo de 32.

Padrão: `[A-Z0-9]+`

Obrigatório: Não

TaskStatus

Status da tarefa de carregamento em lote.

Tipo: string

Valores Válidos: `CREATED | IN_PROGRESS | FAILED | SUCCEEDED | PROGRESS_STOPPED | PENDING_RESUME`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

BatchLoadTaskDescription
Serviço: Amazon Timestream Write

Detalhes sobre uma tarefa de carregamento em lote.

Conteúdo

CreationTime

A hora em que a tarefa de carregamento em lote do Timestream foi criada.

Tipo: carimbo de data/hora

Obrigatório: Não

DataModelConfiguration

Configuração do modelo de dados para uma tarefa de carregamento em lote. Ele contém detalhes sobre onde um modelo de dados para uma tarefa de carregamento em lote é armazenado.

Tipo: objeto DataModelConfiguration

Obrigatório: Não

DataSourceConfiguration

Detalhes de configuração sobre a fonte de dados para uma tarefa de carregamento em lote.

Tipo: objeto DataSourceConfiguration

Obrigatório: não

ErrorMessage

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Obrigatório: Não

LastUpdatedTime

A hora em que a tarefa de carregamento em lote do Timestream foi atualizada pela última vez.

Tipo: carimbo de data/hora

Obrigatório: Não

ProgressReport

Tipo: objeto BatchLoadProgressReport

Obrigatório: Não

RecordVersion

Tipo: longo

Obrigatório: Não

ReportConfiguration

Configuração do relatório para uma tarefa de carregamento em lote. Ele contém detalhes sobre onde os relatórios de erros são armazenados.

Tipo: objeto ReportConfiguration

Obrigatório: Não

ResumableUntil

Tipo: carimbo de data/hora

Obrigatório: não

TargetDatabaseName

Tipo: string

Obrigatório: Não

TargetTableName

Tipo: string

Obrigatório: Não

TaskId

O ID da tarefa de carregamento em lote.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Comprimento máximo de 32.

Padrão: `[A-Z0-9]+`

Obrigatório: Não

TaskStatus

Status da tarefa de carregamento em lote.

Tipo: string

Valores Válidos: `CREATED | IN_PROGRESS | FAILED | SUCCEEDED | PROGRESS_STOPPED | PENDING_RESUME`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

CsvConfiguration
Serviço: Amazon Timestream Write

Um formato de dados delimitado em que o separador de coluna pode ser uma vírgula e o separador de registro é um caractere de nova linha.

Conteúdo

ColumnSeparator

O separador de colunas pode ser vírgula (','), barra vertical ('|), ponto e vírgula (';'), tabulação ('/t') ou espaço em branco (").

Tipo: string

Restrições de comprimento: comprimento fixo de 1.

Obrigatório: Não

EscapeChar

O personagem de fuga pode ser um dos

Tipo: string

Restrições de comprimento: comprimento fixo de 1.

Obrigatório: Não

NullValue

Pode ser um espaço em branco (").

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 256.

Obrigatório: Não

QuoteChar

Podem ser aspas simples (') ou aspas duplas (“).

Tipo: string

Restrições de comprimento: comprimento fixo de 1.

Obrigatório: Não

TrimWhiteSpace

Especifica a redução do espaço em branco à esquerda e à direita.

Tipo: booliano

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

Database
Serviço: Amazon Timestream Write

Um contêiner de nível superior para uma mesa. Bancos de dados e tabelas são os conceitos fundamentais de gerenciamento no Amazon Timestream. Todas as tabelas em um banco de dados são criptografadas com a mesma AWS KMS chave.

Conteúdo

Arn

O nome do recurso da Amazon que identifica exclusivamente esse banco de dados.

Tipo: string

Obrigatório: Não

CreationTime

A hora em que o banco de dados foi criado, calculada a partir da época do Unix.

Tipo: carimbo de data/hora

Obrigatório: Não

DatabaseName

O nome do banco de dados do Timestream.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Tamanho máximo de 256.

Obrigatório: Não

KmsKeyId

O identificador da AWS KMS chave usada para criptografar os dados armazenados no banco de dados.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Obrigatório: Não

LastUpdatedTime

A última vez que esse banco de dados foi atualizado.

Tipo: carimbo de data/hora

Obrigatório: Não

TableCount

O número total de tabelas encontradas em um banco de dados Timestream.

Tipo: longo

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

DataModel
Serviço: Amazon Timestream Write

Modelo de dados para uma tarefa de carregamento em lote.

Conteúdo

DimensionMappings

Mapeamentos de origem para destino para dimensões.

Tipo: Matriz de objetos DimensionMapping

Membros da matriz: número mínimo de 1 item.

Obrigatório: sim

MeasureNameColumn

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 256.

Obrigatório: Não

MixedMeasureMappings

Mapeamentos de origem para destino para medidas.

Tipo: Matriz de objetos MixedMeasureMapping

Membros da matriz: número mínimo de 1 item.

Obrigatório: Não

MultiMeasureMappings

Mapeamentos de origem para destino para registros de várias medidas.

Tipo: objeto MultiMeasureMappings

Obrigatório: Não

TimeColumn

Coluna de origem a ser mapeada para o tempo.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 256.

Obrigatório: Não

TimeUnit

A granularidade da unidade de registro de data e hora. Ele indica se o valor do tempo está em segundos, milissegundos, nanossegundos ou outros valores compatíveis. O padrão é `MILLISECONDS`.

Tipo: string

Valores Válidos: `MILLISECONDS | SECONDS | MICROSECONDS | NANOSECONDS`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

DataModelConfiguration
Serviço: Amazon Timestream Write

Conteúdo

DataModel

Tipo: objeto DataModel

Obrigatório: Não

DataModelS3Configuration

Tipo: objeto DataModelS3Configuration

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

DataModelS3Configuration
Serviço: Amazon Timestream Write

Conteúdo

BucketName

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Tamanho máximo de 63.

Padrão: `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Obrigatório: não

ObjectKey

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 1.024.

Padrão: `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

DataSourceConfiguration
Serviço: Amazon Timestream Write

Define detalhes de configuração sobre a fonte de dados.

Conteúdo

DataFormat

Isso é atualmenteCSV.

Tipo: string

Valores Válidos: CSV

Obrigatório: Sim

DataSourceS3Configuration

Configuração de um local do S3 para um arquivo que contém dados a serem carregados.

Tipo: objeto DataSourceS3Configuration

Obrigatório: Sim

CsvConfiguration

Um formato de dados delimitado em que o separador de coluna pode ser uma vírgula e o separador de registro é um caractere de nova linha.

Tipo: objeto CsvConfiguration

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

DataSourceS3Configuration
Serviço: Amazon Timestream Write

Conteúdo

BucketName

O nome de bucket do bucket do cliente do S3.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Tamanho máximo de 63.

Padrão: `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Exigido: Sim

ObjectKeyPrefix

Tipo: String

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 1.024.

Padrão: `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

Dimension
Serviço: Amazon Timestream Write

Representa os atributos de metadados da série temporal. Por exemplo, o nome e a zona de disponibilidade de uma EC2 instância ou o nome do fabricante de uma turbina eólica são dimensões.

Conteúdo

Name

A dimensão representa os atributos de metadados da série temporal. Por exemplo, o nome e a zona de disponibilidade de uma EC2 instância ou o nome do fabricante de uma turbina eólica são dimensões.

Para restrições sobre nomes de dimensões, consulte Restrições de nomenclatura.

Tipo: string

Restrições de comprimento: tamanho mínimo de 1. Comprimento máximo de 60.

Obrigatório: Sim

Value

O valor da dimensão.

Tipo: string

Obrigatório: Sim

DimensionValueType

O tipo de dados da dimensão para o ponto de dados da série temporal.

Tipo: string

Valores Válidos: `VARCHAR`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

DimensionMapping
Serviço: Amazon Timestream Write

Conteúdo

DestinationColumn

Tipo: string

Restrições de comprimento: tamanho mínimo de 1.

Obrigatório: não

SourceColumn

Tipo: string

Restrições de comprimento: tamanho mínimo de 1.

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

Endpoint
Serviço: Amazon Timestream Write

Representa um endpoint disponível contra o qual fazer API chamadas, bem como o TTL para esse endpoint.

Conteúdo

Address

Um endereço de endpoint.

Tipo: string

Obrigatório: Sim

CachePeriodInMinutes

O TTL para o endpoint, em minutos.

Tipo: longo

Obrigatório: Sim

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

MagneticStoreRejectedDataLocation
Serviço: Amazon Timestream Write

O local no qual gravar relatórios de erros para registros rejeitados, de forma assíncrona, durante gravações de armazenamento magnético.

Conteúdo

S3Configuration

A configuração de um local do S3 no qual gravar relatórios de erros para registros rejeitados, de forma assíncrona, durante gravações de armazenamento magnético.

Tipo: objeto S3Configuration

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

MagneticStoreWriteProperties
Serviço: Amazon Timestream Write

O conjunto de propriedades em uma tabela para configurar gravações em armazenamento magnético.

Conteúdo

EnableMagneticStoreWrites

Um sinalizador para habilitar gravações em armazenamento magnético.

Tipo: booliano

Obrigatório: Sim

MagneticStoreRejectedDataLocation

O local no qual gravar relatórios de erros para registros rejeitados, de forma assíncrona, durante gravações em armazenamento magnético.

Tipo: objeto MagneticStoreRejectedDataLocation

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

MeasureValue
Serviço: Amazon Timestream Write

Representa o atributo de dados da série temporal. Por exemplo, a CPU utilização de uma EC2 instância ou RPM de uma turbina eólica são medidas. MeasureValue tem nome e valor.

MeasureValue só é permitido para o tipo`MULTI`. Usando o `MULTI` tipo, você pode passar vários atributos de dados associados à mesma série temporal em um único registro

Conteúdo

Name

O nome do MeasureValue.

Para restrições de MeasureValue nomes, consulte Restrições de nomenclatura no Amazon Timestream Developer Guide.

Tipo: string

Restrições de comprimento: tamanho mínimo de 1.

Obrigatório: Sim

Type

Contém o tipo de dados do MeasureValue para o ponto de dados da série temporal.

Tipo: string

Valores Válidos: `DOUBLE | BIGINT | VARCHAR | BOOLEAN | TIMESTAMP | MULTI`

Obrigatório: Sim

Value

O valor do MeasureValue. Para obter informações, consulte Tipos de dados.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Obrigatório: Sim

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

MixedMeasureMapping
Serviço: Amazon Timestream Write

Conteúdo

MeasureValueType

Tipo: string

Valores Válidos: `DOUBLE | BIGINT | VARCHAR | BOOLEAN | TIMESTAMP | MULTI`

Obrigatório: sim

MeasureName

Tipo: String

Restrições de comprimento: tamanho mínimo de 1.

Obrigatório: Não

MultiMeasureAttributeMappings

Tipo: Matriz de objetos MultiMeasureAttributeMapping

Membros da matriz: número mínimo de 1 item.

Obrigatório: não

SourceColumn

Tipo: string

Restrições de comprimento: tamanho mínimo de 1.

Obrigatório: não

TargetMeasureName

Tipo: string

Restrições de comprimento: tamanho mínimo de 1.

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

MultiMeasureAttributeMapping
Serviço: Amazon Timestream Write

Conteúdo

SourceColumn

Tipo: string

Restrições de comprimento: tamanho mínimo de 1.

Obrigatório: sim

MeasureValueType

Tipo: String

Valores Válidos: `DOUBLE | BIGINT | BOOLEAN | VARCHAR | TIMESTAMP`

Obrigatório: não

TargetMultiMeasureAttributeName

Tipo: string

Restrições de comprimento: tamanho mínimo de 1.

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

MultiMeasureMappings
Serviço: Amazon Timestream Write

Conteúdo

MultiMeasureAttributeMappings

Tipo: Matriz de objetos MultiMeasureAttributeMapping

Membros da matriz: número mínimo de 1 item.

Obrigatório: sim

TargetMultiMeasureName

Tipo: String

Restrições de comprimento: tamanho mínimo de 1.

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

PartitionKey
Serviço: Amazon Timestream Write

Um atributo usado no particionamento de dados em uma tabela. Uma chave de dimensão particiona os dados usando os valores da dimensão especificada pelo nome da dimensão como chave de partição, enquanto uma chave de medida particiona os dados usando nomes de medida (valores da coluna “nome_medida”).

Conteúdo

Type

O tipo da chave de partição. As opções são DIMENSION (chave de dimensão) e MEASURE (chave de medida).

Tipo: string

Valores Válidos: `DIMENSION | MEASURE`

Obrigatório: Sim

EnforcementInRecord

O nível de fiscalização da especificação de uma chave de dimensão nos registros ingeridos. As opções são REQUIRED (a chave de dimensão deve ser especificada) e OPTIONAL (a chave de dimensão não precisa ser especificada).

Tipo: string

Valores Válidos: `REQUIRED | OPTIONAL`

Obrigatório: Não

Name

O nome do atributo usado para uma chave de dimensão.

Tipo: string

Restrições de comprimento: tamanho mínimo de 1.

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

Record
Serviço: Amazon Timestream Write

Representa um ponto de dados de série temporal sendo gravado no Timestream. Cada registro contém uma matriz de dimensões. As dimensões representam os atributos de metadados de um ponto de dados de série temporal, como o nome da instância ou a zona de disponibilidade de uma EC2 instância. Um registro também contém o nome da medida, que é o nome da medida que está sendo coletada (por exemplo, a CPU utilização de uma EC2 instância). Além disso, um registro contém o valor da medida e o tipo de valor, que é o tipo de dados do valor da medida. Além disso, o registro contém o timestamp de quando a medida foi coletada e a unidade de timestamp, que representa a granularidade do timestamp.

Os registros têm um `Version` campo de 64 bits `long` que você pode usar para atualizar pontos de dados. As gravações de um registro duplicado com a mesma dimensão, data e hora e nome de medida, mas com valores de medida diferentes, só serão bem-sucedidas se o `Version` atributo do registro na solicitação de gravação for maior que o do registro existente. O padrão do Timestream é `Version` de `1` para registros sem o campo. `Version`

Conteúdo

Dimensions

Contém a lista de dimensões para pontos de dados de séries temporais.

Tipo: matriz de objetos Dimension

Membros da matriz: número máximo de 128 itens.

Obrigatório: Não

MeasureName

A medida representa o atributo de dados da série temporal. Por exemplo, a CPU utilização de uma EC2 instância ou RPM de uma turbina eólica são medidas.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 256.

Obrigatório: Não

MeasureValue

Contém o valor da medida para o ponto de dados da série temporal.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Obrigatório: Não

MeasureValues

Contém a lista de quatro pontos MeasureValue de dados de séries temporais.

Isso só é permitido para tipos`MULTI`. Para valores escalares, use o `MeasureValue` atributo do registro diretamente.

Tipo: matriz de objetos MeasureValue

Obrigatório: Não

MeasureValueType

Contém o tipo de dados do valor da medida para o ponto de dados da série temporal. O tipo padrão é`DOUBLE`. Para obter mais informações, consulte Tipos de dados.

Tipo: string

Valores Válidos: `DOUBLE | BIGINT | VARCHAR | BOOLEAN | TIMESTAMP | MULTI`

Obrigatório: Não

Time

Contém a hora em que o valor da medida para o ponto de dados foi coletado. O valor do tempo mais a unidade fornece o tempo decorrido desde a época. Por exemplo, se o valor do tempo for `12345` e a unidade for`ms`, então já `12345 ms` passaram desde a época.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 256.

Obrigatório: Não

TimeUnit

A granularidade da unidade de registro de data e hora. Ele indica se o valor do tempo está em segundos, milissegundos, nanossegundos ou outros valores compatíveis. O padrão é `MILLISECONDS`.

Tipo: string

Valores Válidos: `MILLISECONDS | SECONDS | MICROSECONDS | NANOSECONDS`

Obrigatório: Não

Version

Atributo de 64 bits usado para atualizações de registros. As solicitações de gravação de dados duplicados com um número de versão maior atualizarão o valor e a versão existentes da medida. Nos casos em que o valor da medida for o mesmo, ainda `Version` será atualizado. O valor padrão é 1.

> **Note**
>
> `Version`deve ser 1 ou maior, ou você receberá um `ValidationException` erro.

Tipo: longo

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

RecordsIngested
Serviço: Amazon Timestream Write

Informações sobre os registros ingeridos por essa solicitação.

Conteúdo

MagneticStore

Contagem de registros ingeridos na loja magnética.

Tipo: número inteiro

Obrigatório: não

MemoryStore

Contagem de registros ingeridos no armazenamento de memória.

Tipo: número inteiro

Obrigatório: não

Total

Contagem total de registros ingeridos com sucesso.

Tipo: número inteiro

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

RejectedRecord
Serviço: Amazon Timestream Write

Representa registros que não foram inseridos com sucesso no Timestream devido a problemas de validação de dados que devem ser resolvidos antes da reinserção de dados de séries temporais no sistema.

Conteúdo

ExistingVersion

A versão existente do registro. Esse valor é preenchido em cenários em que existe um registro idêntico com uma versão superior à versão na solicitação de gravação.

Tipo: longo

Obrigatório: Não

Reason

O motivo pelo qual um registro não foi inserido com sucesso no Timestream. As possíveis causas de falha incluem:

- Registros com dados duplicados em que há vários registros com as mesmas dimensões, carimbos de data/hora e nomes de medidas, mas:
  - Os valores das medidas são diferentes
  - A versão não está presente na solicitação ou o valor da versão no novo registro é igual ou inferior ao valor existente

  Se o Timestream rejeitar os dados desse caso, o `ExistingVersion` campo na `RejectedRecords` resposta indicará a versão atual do registro. Para forçar uma atualização, você pode reenviar a solicitação com uma versão do conjunto de registros com um valor maior que o. `ExistingVersion`
- Registros com carimbos de data/hora que estão fora da duração de retenção do armazenamento de memória.

  > **Note**
  >
  > Quando a janela de retenção for atualizada, você receberá uma `RejectedRecords` exceção se tentar imediatamente ingerir dados dentro da nova janela. Para evitar uma `RejectedRecords` exceção, espere até a duração da nova janela para ingerir novos

dados. Para obter mais informações, consulte Melhores práticas para configurar o Timestream e a explicação de como o armazenamento funciona no Timestream.

- Registros com dimensões ou medidas que excedem os limites definidos pelo Timestream.

Para obter mais informações, consulte Gerenciamento de acesso no Guia do desenvolvedor do Timestream.

Tipo: string

Obrigatório: Não

RecordIndex

O índice do registro na solicitação de entrada para WriteRecords. Os índices começam com 0.

Tipo: número inteiro

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

ReportConfiguration
Serviço: Amazon Timestream Write

Configuração do relatório para uma tarefa de carregamento em lote. Ele contém detalhes sobre onde os relatórios de erros são armazenados.

Conteúdo

ReportS3Configuration

Configuração de um local do S3 para gravar relatórios de erros e eventos para um carregamento em lote.

Tipo: objeto ReportS3Configuration

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

ReportS3Configuration
Serviço: Amazon Timestream Write

Conteúdo

BucketName

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Tamanho máximo de 63.

Padrão: `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Exigido: Sim

EncryptionOption

Tipo: String

Valores Válidos: `SSE_S3 | SSE_KMS`

Obrigatório: não

KmsKeyId

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Obrigatório: não

ObjectKeyPrefix

Tipo: string

Restrições de comprimento: tamanho mínimo de 1. Comprimento máximo de 928.

Padrão: `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

RetentionProperties
Serviço: Amazon Timestream Write

As propriedades de retenção contêm o período de tempo durante o qual dados de série temporal devem ser armazenados no armazenamento magnético e no armazenamento de memória.

Conteúdo

MagneticStoreRetentionPeriodInDays

O período de tempo durante o qual os dados devem ser armazenados no armazenamento magnético.

Tipo: longo

Faixa válida: valor mínimo de 1. Valor máximo de 73000.

Obrigatório: Sim

MemoryStoreRetentionPeriodInHours

o período de tempo durante o qual os dados devem ser armazenados de memória magnético.

Tipo: longo

Faixa válida: valor mínimo de 1. Valor máximo de 8766.

Obrigatório: Sim

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

S3Configuration
Serviço: Amazon Timestream Write

A configuração que especifica um local do S3.

Conteúdo

BucketName

O nome de bucket do bucket do cliente do S3.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Tamanho máximo de 63.

Padrão: `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Obrigatório: Não

EncryptionOption

A opção de criptografia para o local do cliente do S3. As opções são criptografia do lado do servidor S3 com uma chave gerenciada S3 ou chave gerenciada. AWS

Tipo: string

Valores Válidos: `SSE_S3 | SSE_KMS`

Obrigatório: Não

KmsKeyId

O ID da AWS KMS chave do local S3 do cliente ao criptografar com uma chave AWS gerenciada.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Obrigatório: Não

ObjectKeyPrefix

A pré-visualização da chave do objeto para um local do cliente do S3.

Tipo: string

Restrições de comprimento: tamanho mínimo de 1. Comprimento máximo de 928.

Padrão: `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

Schema
Serviço: Amazon Timestream Write

Um esquema especifica o modelo de dados esperado da tabela.

Conteúdo

CompositePartitionKey

Uma lista não vazia de chaves de partição que define os atributos usados para particionar os dados da tabela. A ordem da lista determina a hierarquia da partição. O nome e o tipo de cada chave de partição, bem como a ordem da chave de partição, não podem ser alterados após a criação da tabela. No entanto, o nível de imposição de cada chave de partição pode ser alterado.

Tipo: Matriz de objetos PartitionKey

Membros da matriz: número mínimo de 1 item.

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

Table
Serviço: Amazon Timestream Write

Representa uma tabela de banco de dados no Timestream. As tabelas contêm uma ou mais séries temporais relacionadas. Você pode modificar a duração da retenção do armazenamento de memória e do armazenamento magnético de uma tabela.

Conteúdo

Arn

O nome do recurso da Amazon que identifica exclusivamente essa tabela.

Tipo: string

Obrigatório: Não

CreationTime

A hora em que a tabela Timestream foi criada.

Tipo: carimbo de data/hora

Obrigatório: Não

DatabaseName

O nome do banco de dados do Timestream que contém essa tabela.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Tamanho máximo de 256.

Obrigatório: Não

LastUpdatedTime

A hora em que a tabela Timestream foi atualizada pela última vez.

Tipo: carimbo de data/hora

Obrigatório: Não

MagneticStoreWriteProperties

Contém propriedades a serem definidas na tabela ao habilitar gravações de armazenamento magnético.

Tipo: objeto MagneticStoreWriteProperties

Obrigatório: Não

RetentionProperties

A duração da retenção para o armazenamento de memória e o armazenamento magnético.

Tipo: objeto RetentionProperties

Obrigatório: Não

Schema

O esquema da tabela.

Tipo: objeto Schema

Obrigatório: Não

TableName

O nome da tabela do Timestream.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Tamanho máximo de 256.

Obrigatório: Não

TableStatus

O estado atual da tabela:

- `DELETING`- A tabela está sendo excluída.
- `ACTIVE`- A mesa está pronta para uso.

Tipo: string

Valores Válidos: `ACTIVE | DELETING | RESTORING`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

Tag
Serviço: Amazon Timestream Write

Uma tag é um rótulo que você atribui às 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 tabelas do banco de dados Timestream, por exemplo, por finalidade, proprietário ou ambiente.

Conteúdo

Key

A chave da tag. Chaves de tag fazem distinção entre maiúsculas e minúsculas.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. O tamanho máximo é 128.

Obrigatório: Sim

Value

O valor da tag. Os valores das tags diferenciam maiúsculas de minúsculas e podem ser nulos.

Tipo: string

Restrições de tamanho: o tamanho mínimo é 0. O tamanho máximo é 256.

Obrigatório: Sim

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

Consulta do Amazon Timestream

Os seguintes tipos de dados são compatíveis com o Amazon Timestream Query:

- ColumnInfo
- Datum
- DimensionMapping
- Endpoint
- ErrorReportConfiguration
- ErrorReportLocation
- ExecutionStats
- MixedMeasureMapping
- MultiMeasureAttributeMapping
- MultiMeasureMappings
- NotificationConfiguration
- ParameterMapping
- QueryInsights
- QueryInsightsResponse
- QuerySpatialCoverage
- QuerySpatialCoverageMax
- QueryStatus
- QueryTemporalRange
- QueryTemporalRangeMax
- Row
- S3Configuration
- S3ReportLocation
- ScheduleConfiguration
- ScheduledQuery
- ScheduledQueryDescription
- ScheduledQueryInsights
- ScheduledQueryInsightsResponse
- ScheduledQueryRunSummary
- SelectColumn
- SnsConfiguration

- Tag
- TargetConfiguration
- TargetDestination
- TimeSeriesDataPoint
- TimestreamConfiguration
- TimestreamDestination
- Type

ColumnInfo
Serviço: Amazon Timestream Query

Contém os metadados dos resultados da consulta, como nomes de colunas, tipos de dados e outros atributos.

Conteúdo

Type

O tipo de dados da coluna do conjunto de resultados. O tipo de dados pode ser escalar ou complexo. Os tipos de dados escalares são números inteiros, cadeias de caracteres, duplos, booleanos e outros. Tipos de dados complexos são tipos como matrizes, linhas e outros.

Tipo: objeto Type

Obrigatório: Sim

Name

O nome da coluna do conjunto de resultados. O nome do conjunto de resultados está disponível para colunas de todos os tipos de dados, exceto matrizes.

Tipo: string

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

Datum

Serviço: Amazon Timestream Query

O datum representa um único ponto de dados em um resultado de consulta.

Conteúdo

ArrayValue

Indica se o ponto de dados é uma matriz.

Tipo: matriz de objetos Datum

Obrigatório: Não

NullValue

Indica se o ponto de dados é nulo.

Tipo: booliano

Obrigatório: não

RowValue

Indica se o ponto de dados é uma linha.

Tipo: objeto Row

Obrigatório: Não

ScalarValue

Indica se o ponto de dados é um valor escalar, como inteiro, string, duplo ou booleano.

Tipo: string

Obrigatório: Não

TimeSeriesValue

Indica se o ponto de dados é um tipo de dados de série temporal.

Tipo: matriz de objetos TimeSeriesDataPoint

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

DimensionMapping
Serviço: Amazon Timestream Query

Esse tipo é utilizado para mapear coluna(s) do resultado da consulta para uma dimensão na tabela de destino.

Conteúdo

DimensionValueType

Tipo da dimensão.

Tipo: string

Valores Válidos: `VARCHAR`

Obrigatório: Sim

Name

Nome da coluna do resultado da consulta.

Tipo: string

Obrigatório: Sim

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

Endpoint
Serviço: Amazon Timestream Query

Representa um endpoint disponível contra o qual fazer API chamadas, bem como o TTL para esse endpoint.

Conteúdo

Address

Um endereço de endpoint.

Tipo: string

Obrigatório: Sim

CachePeriodInMinutes

O TTL para o endpoint, em minutos.

Tipo: longo

Obrigatório: Sim

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

ErrorReportConfiguration
Serviço: Amazon Timestream Query

Configuração necessária para relatórios de erros.

Conteúdo

S3Configuration

A configuração do S3 para os relatórios de erros.

Tipo: objeto S3Configuration

Obrigatório: Sim

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

ErrorReportLocation
Serviço: Amazon Timestream Query

Ele contém a localização do relatório de erros para uma única chamada de consulta agendada.

Conteúdo

S3ReportLocation

O local do S3 onde os relatórios de erro são gravados.

Tipo: objeto S3ReportLocation

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

ExecutionStats
Serviço: Amazon Timestream Query

Estatísticas para uma única execução de consulta agendada.

Conteúdo

BytesMetered

Bytes medidos para uma única execução de consulta agendada.

Tipo: longo

Obrigatório: Não

CumulativeBytesScanned

Bytes escaneados para uma única execução de consulta agendada.

Tipo: longo

Obrigatório: Não

DataWrites

As gravações de dados foram medidas para registros ingeridos em uma única execução de consulta agendada.

Tipo: longo

Obrigatório: Não

ExecutionTimeInMillis

Tempo total, medido em milissegundos, necessário para a conclusão da execução da consulta agendada.

Tipo: longo

Obrigatório: Não

QueryResultRows

Número de linhas presentes na saída da execução de uma consulta antes da ingestão na fonte de dados de destino.

Tipo: longo

Obrigatório: Não

RecordsIngested

O número de registros ingeridos para uma única execução de consulta agendada.

Tipo: longo

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

MixedMeasureMapping
Serviço: Amazon Timestream Query

MixedMeasureMappings são mapeamentos que podem ser usados para ingerir dados em uma mistura de medidas estreitas e múltiplas na tabela derivada.

Conteúdo

MeasureValueType

Tipo do valor que deve ser lidosourceColumn. Se o mapeamento for paraMULTI, use MeasureValueType. MULTI.

Tipo: string

Valores Válidos: `BIGINT | BOOLEAN | DOUBLE | VARCHAR | MULTI`

Obrigatório: Sim

MeasureName

Refere-se ao valor de measure_name em uma linha de resultado. Esse campo é obrigatório se MeasureNameColumn for fornecido.

Tipo: string

Obrigatório: Não

MultiMeasureAttributeMappings

Exigido quando measureValueType éMULTI. Mapeamentos de atributos para medidas de MULTI valor.

Tipo: Matriz de objetos MultiMeasureAttributeMapping

Membros da matriz: número mínimo de 1 item.

Obrigatório: Não

SourceColumn

Esse campo refere-se à coluna de origem da qual o valor de medida deve ser lido para a materialização do resultado.

Tipo: string

Obrigatório: Não

TargetMeasureName

Nome da medida de destino a ser utilizada. Se não for fornecido, o nome da medida de destino, por padrão, seria nome da medida, se fornecido, ou sourceColumn de outra forma.

Tipo: string

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

MultiMeasureAttributeMapping
Serviço: Amazon Timestream Query

Mapeamento de atributos para medidas de MULTI valor.

Conteúdo

MeasureValueType

Tipo do atributo a ser lido na coluna de origem.

Tipo: string

Valores Válidos: `BIGINT | BOOLEAN | DOUBLE | VARCHAR | TIMESTAMP`

Obrigatório: Sim

SourceColumn

Coluna de origem de onde o valor do atributo deve ser lido.

Tipo: string

Obrigatório: Sim

TargetMultiMeasureAttributeName

Nome personalizado a ser utilizado para o nome do atributo na tabela derivada. Se não for fornecido, o nome da coluna de origem será usado.

Tipo: string

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

MultiMeasureMappings
Serviço: Amazon Timestream Query

Apenas um dos MixedMeasureMappings ou MultiMeasureMappings deve ser fornecido. MultiMeasureMappings pode ser usado para ingerir dados como medidas múltiplas na tabela derivada.

Conteúdo

MultiMeasureAttributeMappings

Obrigatório. Mapeamentos de atributos a serem utilizados para mapear resultados de consulta para ingerir dados para atributos de várias medidas.

Tipo: Matriz de objetos MultiMeasureAttributeMapping

Membros da matriz: número mínimo de 1 item.

Obrigatório: Sim

TargetMultiMeasureName

O nome do nome de várias medidas de destino na tabela derivada. Essa entrada é necessária quando não measureNameColumn é fornecida. Se MeasureNameColumn for fornecido, o valor dessa coluna será usado como nome de várias medidas.

Tipo: string

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

NotificationConfiguration
Serviço: Amazon Timestream Query

Configuração de notificação para uma consulta programada. Uma notificação é enviada pelo Timestream quando uma consulta programada é criada, seu estado é atualizado ou quando é excluído.

Conteúdo

SnsConfiguration

Detalhes sobre a SNS configuração.

Tipo: objeto SnsConfiguration

Obrigatório: Sim

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

ParameterMapping
Serviço: Amazon Timestream Query

Mapeamento para parâmetros nomeados.

Conteúdo

Name

Nome do parâmetro.

Tipo: string

Obrigatório: Sim

Type

Contém o tipo de dados de uma coluna em um conjunto de resultados de consulta. O tipo de dados pode ser escalar ou complexo. Os tipos de dados escalares suportados são inteiros, booleanos, string, double, timestamp, data, hora e intervalos. Os tipos de dados complexos compatíveis são matrizes, linhas e séries temporais.

Tipo: objeto Type

Obrigatório: Sim

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

QueryInsights
Serviço: Amazon Timestream Query

`QueryInsights`é um recurso de ajuste de desempenho que ajuda você a otimizar suas consultas, reduzir custos e melhorar o desempenho. Com`QueryInsights`, você pode avaliar a eficiência da redução de suas consultas e identificar áreas de melhoria para aprimorar o desempenho das consultas. Com`QueryInsights`, você também pode analisar a eficácia de suas consultas em termos de redução temporal e espacial e identificar oportunidades para melhorar o desempenho. Especificamente, você pode avaliar o quão bem suas consultas usam estratégias de indexação baseadas em tempo e chave de partição para otimizar a recuperação de dados. Para otimizar o desempenho da consulta, é essencial que você ajuste os parâmetros temporais e espaciais que governam a execução da consulta.

As principais métricas fornecidas por `QueryInsights` são `QuerySpatialCoverage` `QueryTemporalRange` e. `QuerySpatialCoverage`indica quanto do eixo espacial a consulta digitaliza, com valores mais baixos sendo mais eficientes. `QueryTemporalRange`mostra o intervalo de tempo escaneado, com intervalos mais estreitos tendo melhor desempenho.

Benefícios do QueryInsights

A seguir estão os principais benefícios do uso`QueryInsights`:

- Identificação de consultas ineficientes — `QueryInsights` fornece informações sobre a remoção baseada em tempo e em atributos das tabelas acessadas pela consulta. Essas informações ajudam a identificar as tabelas que não são acessadas de forma ideal.
- Otimizando seu modelo de dados e particionamento — Você pode usar as `QueryInsights` informações para acessar e ajustar seu modelo de dados e sua estratégia de particionamento.
- Ajuste de consultas — `QueryInsights` destaca as oportunidades de usar índices com mais eficiência.

> **Note**
>
> O número máximo de `Query` API solicitações que você pode fazer com `QueryInsights` ativado é de 1 consulta por segundo (QPS). Se você exceder essa taxa de consulta, isso poderá resultar em limitação.

Conteúdo

Mode

Fornece os seguintes modos para ativar`QueryInsights`:

- `ENABLED_WITH_RATE_CONTROL`— Habilita `QueryInsights` as consultas que estão sendo processadas. Esse modo também inclui um mecanismo de controle de taxa, que limita o `QueryInsights` recurso a 1 consulta por segundo (QPS).
- `DISABLED`— Desativa`QueryInsights`.

Tipo: string

Valores Válidos: `ENABLED_WITH_RATE_CONTROL | DISABLED`

Exigido: Sim

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

QueryInsightsResponse
Serviço: Amazon Timestream Query

Fornece vários insights e métricas relacionados à consulta que você executou.

Conteúdo

OutputBytes

Indica o tamanho do resultado da consulta definido em bytes. Você pode usar esses dados para validar se o conjunto de resultados foi alterado como parte do exercício de ajuste da consulta.

Tipo: longo

Obrigatório: Não

OutputRows

Indica o número total de linhas retornadas como parte do conjunto de resultados da consulta. Você pode usar esses dados para validar se o número de linhas no conjunto de resultados foi alterado como parte do exercício de ajuste da consulta.

Tipo: longo

Obrigatório: Não

QuerySpatialCoverage

Fornece informações sobre a cobertura espacial da consulta, incluindo a tabela com redução espacial abaixo do ideal (máximo). Essas informações podem ajudá-lo a identificar áreas de melhoria em sua estratégia de particionamento para aprimorar a poda espacial.

Tipo: objeto QuerySpatialCoverage

Obrigatório: Não

QueryTableCount

Indica o número de tabelas na consulta.

Tipo: longo

Obrigatório: Não

QueryTemporalRange

Fornece informações sobre o intervalo temporal da consulta, incluindo a tabela com o maior intervalo de tempo (máximo). A seguir estão algumas das opções possíveis para otimizar a poda com base no tempo:

- Adicione predicados de tempo ausentes.
- Remova funções de acordo com os predicados de tempo.
- Adicione predicados de tempo a todas as subconsultas.

Tipo: objeto QueryTemporalRange

Obrigatório: Não

UnloadPartitionCount

Indica as partições criadas pela `Unload` operação.

Tipo: longo

Obrigatório: Não

UnloadWrittenBytes

Indica o tamanho, em bytes, gravado pela `Unload` operação.

Tipo: longo

Obrigatório: Não

UnloadWrittenRows

Indica as linhas gravadas pela `Unload` consulta.

Tipo: longo

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++

- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

QuerySpatialCoverage
Serviço: Amazon Timestream Query

Fornece informações sobre a cobertura espacial da consulta, incluindo a tabela com redução espacial abaixo do ideal (máximo). Essas informações podem ajudá-lo a identificar áreas de melhoria em sua estratégia de particionamento para aprimorar a poda espacial.

Por exemplo, você pode fazer o seguinte com as `QuerySpatialCoverage` informações:

- Adicione measure_name ou use predicados de chave de partição () definidos pelo cliente. CDPK
- Se você já executou a ação anterior, remova as funções em torno delas ou cláusulas, como. `LIKE`

Conteúdo

Max

Fornece informações sobre a cobertura espacial da consulta executada e da tabela com a redução espacial mais ineficiente.

- `Value`— A proporção máxima de cobertura espacial.
- `TableArn`— O Amazon Resource Name (ARN) da tabela com poda espacial abaixo do ideal.
- `PartitionKey`— A chave de partição usada para particionamento, que pode ser padrão `measure_name` ou a. CDPK

Tipo: objeto QuerySpatialCoverageMax

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

QuerySpatialCoverageMax
Serviço: Amazon Timestream Query

Fornece informações sobre a tabela com a faixa espacial mais abaixo do ideal escaneada pela sua consulta.

Conteúdo

PartitionKey

A chave de partição usada para particionamento, que pode ser uma chave de partição padrão `measure_name` ou definida pelo cliente.

Tipo: Matriz de strings

Obrigatório: Não

TableArn

O Amazon Resource Name (ARN) da tabela com a poda espacial mais abaixo do ideal.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Obrigatório: Não

Value

A proporção máxima de cobertura espacial.

Tipo: duplo

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

QueryStatus
Serviço: Amazon Timestream Query

Informações sobre o status da consulta, incluindo progresso e bytes verificados.

Conteúdo

CumulativeBytesMetered

A quantidade de dados digitalizados pela consulta em bytes pela qual você será cobrado. Essa é uma soma cumulativa e representa a quantidade total de dados pela qual você será cobrado desde o início da consulta. A cobrança é aplicada somente uma vez e é aplicada quando a consulta é concluída ou quando a consulta é cancelada.

Tipo: longo

Obrigatório: Não

CumulativeBytesScanned

A quantidade de dados digitalizados pela consulta em bytes. Essa é uma soma cumulativa e representa a quantidade total de bytes verificados desde que a consulta foi iniciada.

Tipo: longo

Obrigatório: Não

ProgressPercentage

O progresso da consulta, expresso como uma porcentagem.

Tipo: duplo

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

QueryTemporalRange
Serviço: Amazon Timestream Query

Fornece informações sobre o intervalo temporal da consulta, incluindo a tabela com o maior intervalo de tempo (máximo).

Conteúdo

Max

Encapsula as seguintes propriedades que fornecem informações sobre a tabela de desempenho mais abaixo do ideal no eixo temporal:

- `Value`— A duração máxima em nanossegundos entre o início e o fim da consulta.
- `TableArn`— O Amazon Resource Name (ARN) da tabela que é consultada com o maior intervalo de tempo.

Tipo: objeto QueryTemporalRangeMax

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

QueryTemporalRangeMax
Serviço: Amazon Timestream Query

Fornece informações sobre a tabela com a poda temporal mais abaixo do ideal verificada pela sua consulta.

Conteúdo

TableArn

O Amazon Resource Name (ARN) da tabela que é consultada com o maior intervalo de tempo.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Obrigatório: Não

Value

A duração máxima em nanossegundos entre o início e o fim da consulta.

Tipo: longo

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

Row
Serviço: Amazon Timestream Query

Representa uma única linha nos resultados da consulta.

Conteúdo

Data

Lista de pontos de dados em uma única linha do conjunto de resultados.

Tipo: matriz de objetos Datum

Obrigatório: Sim

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

S3Configuration
Serviço: Amazon Timestream Query

Detalhes sobre a localização do S3 para relatórios de erros resultantes da execução de uma consulta.

Conteúdo

BucketName

Nome do bucket do S3 sob o qual os relatórios de erros serão criados.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Tamanho máximo de 63.

Padrão: `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Exigido: Sim

EncryptionOption

Opções de criptografia em repouso para os relatórios de erros. Se nenhuma opção de criptografia for especificada, o Timestream escolherá SSE _S3 como padrão.

Tipo: string

Valores Válidos: `SSE_S3 | SSE_KMS`

Obrigatório: Não

ObjectKeyPrefix

Prefixo para a chave do relatório de erros. Por padrão, o fluxo de tempo adiciona o seguinte prefixo ao caminho do relatório de erros.

Tipo: string

Restrições de comprimento: tamanho mínimo de 1. Comprimento máximo de 896.

Padrão: `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

S3ReportLocation
Serviço: Amazon Timestream Query

Local do relatório S3 para a execução da consulta agendada.

Conteúdo

BucketName

O nome do bucket do S3.

Tipo: string

Restrições de tamanho: comprimento mínimo de 3. Tamanho máximo de 63.

Padrão: `[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]`

Obrigatório: Não

ObjectKey

Chave S3.

Tipo: string

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

ScheduleConfiguration
Serviço: Amazon Timestream Query

Configuração da programação da consulta.

Conteúdo

ScheduleExpression

Uma expressão que indica quando acionar a execução da consulta programada. Pode ser uma expressão cron ou uma expressão de taxa.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 256.

Obrigatório: Sim

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

ScheduledQuery
Serviço: Amazon Timestream Query

Consulta agendada

Conteúdo

Arn

O nome do recurso da Amazon.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Obrigatório: Sim

Name

O nome da consulta agendada.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Exigido: Sim

State

Estado da consulta agendada.

Tipo: string

Valores Válidos: `ENABLED | DISABLED`

Obrigatório: Sim

CreationTime

O horário de criação da consulta agendada.

Tipo: carimbo de data/hora

Obrigatório: Não

ErrorReportConfiguration

Configuração para relatório de erros de consulta agendada.

Tipo: objeto ErrorReportConfiguration

Obrigatório: Não

LastRunStatus

Status da última execução de consulta agendada.

Tipo: string

Valores Válidos: `AUTO_TRIGGER_SUCCESS | AUTO_TRIGGER_FAILURE | MANUAL_TRIGGER_SUCCESS | MANUAL_TRIGGER_FAILURE`

Obrigatório: Não

NextInvocationTime

Na próxima vez em que a consulta agendada for executada.

Tipo: carimbo de data/hora

Obrigatório: Não

PreviousInvocationTime

A última vez em que a consulta agendada foi executada.

Tipo: carimbo de data/hora

Obrigatório: Não

TargetDestination

Fonte de dados de destino na qual o resultado final da consulta agendada será gravado.

Tipo: objeto TargetDestination

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

ScheduledQueryDescription
Serviço: Amazon Timestream Query

Estrutura que descreve a consulta agendada.

Conteúdo

Arn

Consulta agendadaARN.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Obrigatório: Sim

Name

Nome da consulta agendada.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Comprimento máximo de 64.

Padrão: `[a-zA-Z0-9|!\-_*'\(\)]([a-zA-Z0-9]|[!\-_*'\(\)\/.])+`

Exigido: Sim

NotificationConfiguration

Configuração de notificação.

Tipo: objeto NotificationConfiguration

Obrigatório: Sim

QueryString

A consulta a ser executada.

Tipo: string

Restrições de comprimento: tamanho mínimo de 1. Tamanho máximo de 262144.

Obrigatório: Sim

ScheduleConfiguration

Configuração de programação.

Tipo: objeto ScheduleConfiguration

Obrigatório: Sim

State

Estado da consulta agendada.

Tipo: string

Valores Válidos: `ENABLED | DISABLED`

Obrigatório: Sim

CreationTime

Horário de criação da consulta agendada.

Tipo: carimbo de data/hora

Obrigatório: Não

ErrorReportConfiguration

Configuração de relatório de erros para a consulta agendada.

Tipo: objeto ErrorReportConfiguration

Obrigatório: Não

KmsKeyId

Uma KMS chave fornecida pelo cliente usada para criptografar o recurso de consulta agendada.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Obrigatório: Não

LastRunSummary

Resumo do tempo de execução da última consulta agendada.

Tipo: objeto ScheduledQueryRunSummary

Obrigatório: Não

NextInvocationTime

Na próxima vez em que a consulta agendada for programada para ser executada.

Tipo: carimbo de data/hora

Obrigatório: Não

PreviousInvocationTime

Última vez que a consulta foi executada.

Tipo: carimbo de data/hora

Obrigatório: Não

RecentlyFailedRuns

Resumo do tempo de execução das últimas cinco execuções de consulta agendadas com falha.

Tipo: matriz de objetos ScheduledQueryRunSummary

Obrigatório: Não

ScheduledQueryExecutionRoleArn

IAMfunção que o Timestream usa para executar a consulta de agendamento.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Obrigatório: Não

TargetConfiguration

Configuração programada do armazenamento de destinos da consulta.

Tipo: objeto TargetConfiguration

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

ScheduledQueryInsights
Serviço: Amazon Timestream Query

Encapsula as configurações para ativação `QueryInsights` em um. `ExecuteScheduledQueryRequest`

Conteúdo

Mode

Fornece os seguintes modos para ativar`ScheduledQueryInsights`:

- `ENABLED_WITH_RATE_CONTROL`— Habilita `ScheduledQueryInsights` as consultas que estão sendo processadas. Esse modo também inclui um mecanismo de controle de taxa, que limita o `QueryInsights` recurso a 1 consulta por segundo (QPS).
- `DISABLED`— Desativa`ScheduledQueryInsights`.

Tipo: string

Valores Válidos: `ENABLED_WITH_RATE_CONTROL | DISABLED`

Exigido: Sim

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

ScheduledQueryInsightsResponse
Serviço: Amazon Timestream Query

Fornece vários insights e métricas relacionados ao `ExecuteScheduledQueryRequest` que foi executado.

Conteúdo

OutputBytes

Indica o tamanho do resultado da consulta definido em bytes. Você pode usar esses dados para validar se o conjunto de resultados foi alterado como parte do exercício de ajuste da consulta.

Tipo: longo

Obrigatório: Não

OutputRows

Indica o número total de linhas retornadas como parte do conjunto de resultados da consulta. Você pode usar esses dados para validar se o número de linhas no conjunto de resultados foi alterado como parte do exercício de ajuste da consulta.

Tipo: longo

Obrigatório: Não

QuerySpatialCoverage

Fornece informações sobre a cobertura espacial da consulta, incluindo a tabela com redução espacial abaixo do ideal (máximo). Essas informações podem ajudá-lo a identificar áreas de melhoria em sua estratégia de particionamento para aprimorar a poda espacial.

Tipo: objeto QuerySpatialCoverage

Obrigatório: Não

QueryTableCount

Indica o número de tabelas na consulta.

Tipo: longo

Obrigatório: Não

QueryTemporalRange

Fornece informações sobre o intervalo temporal da consulta, incluindo a tabela com o maior intervalo de tempo (máximo). A seguir estão algumas das opções possíveis para otimizar a poda com base no tempo:

- Adicione predicados de tempo ausentes.
- Remova funções de acordo com os predicados de tempo.
- Adicione predicados de tempo a todas as subconsultas.

Tipo: objeto QueryTemporalRange

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

ScheduledQueryRunSummary
Serviço: Amazon Timestream Query

Execute o resumo da consulta agendada

Conteúdo

ErrorReportLocation

Local do S3 para relatório de erros.

Tipo: objeto ErrorReportLocation

Obrigatório: Não

ExecutionStats

Estatísticas de tempo de execução para uma execução programada.

Tipo: objeto ExecutionStats

Obrigatório: Não

FailureReason

Mensagem de erro para a consulta agendada em caso de falha. Talvez seja necessário consultar o relatório de erros para obter mais detalhes sobre os motivos dos erros.

Tipo: string

Obrigatório: Não

InvocationTime

InvocationTime para esta corrida. Esse é o horário em que a consulta está programada para ser executada. O parâmetro `@scheduled_runtime` pode ser usado na consulta para obter o valor.

Tipo: carimbo de data/hora

Obrigatório: Não

QueryInsightsResponse

Fornece vários insights e métricas relacionados ao resumo da execução da consulta agendada.

Tipo: objeto ScheduledQueryInsightsResponse

Obrigatório: Não

RunStatus

O status de uma execução de consulta agendada.

Tipo: string

Valores Válidos: `AUTO_TRIGGER_SUCCESS | AUTO_TRIGGER_FAILURE | MANUAL_TRIGGER_SUCCESS | MANUAL_TRIGGER_FAILURE`

Obrigatório: Não

TriggerTime

A hora real em que a consulta foi executada.

Tipo: carimbo de data/hora

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

SelectColumn
Serviço: Amazon Timestream Query

Detalhes da coluna que é retornada pela consulta.

Conteúdo

Aliased

Verdadeiro, se o nome da coluna tiver o alias da consulta. Caso contrário, falso.

Tipo: booliano

Obrigatório: não

DatabaseName

Banco de dados que tem essa coluna.

Tipo: string

Obrigatório: Não

Name

O nome da coluna.

Tipo: string

Obrigatório: Não

TableName

Tabela dentro do banco de dados que tem essa coluna.

Tipo: string

Obrigatório: Não

Type

Contém o tipo de dados de uma coluna em um conjunto de resultados de consulta. O tipo de dados pode ser escalar ou complexo. Os tipos de dados escalares suportados são inteiros, booleanos, string, double, timestamp, data, hora e intervalos. Os tipos de dados complexos compatíveis são matrizes, linhas e séries temporais.

Tipo: objeto Type

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

SnsConfiguration
Serviço: Amazon Timestream Query

Detalhes sobre SNS isso são necessários para enviar a notificação.

Conteúdo

TopicArn

SNStópico para ARN o qual as notificações de status da consulta agendada serão enviadas.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. Tamanho máximo de 2.048.

Obrigatório: Sim

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

Tag
Serviço: Amazon Timestream Query

Uma tag é um rótulo que você atribui às 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 tabelas do banco de dados Timestream, por exemplo, por finalidade, proprietário ou ambiente.

Conteúdo

Key

A chave da tag. Chaves de tag fazem distinção entre maiúsculas e minúsculas.

Tipo: string

Restrições de tamanho: tamanho mínimo 1. O tamanho máximo é 128.

Obrigatório: Sim

Value

O valor da tag. Os valores de tag diferenciam maiúsculas de minúsculas e podem ser nulos.

Tipo: string

Restrições de tamanho: o tamanho mínimo é 0. O tamanho máximo é 256.

Obrigatório: Sim

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

TargetConfiguration
Serviço: Amazon Timestream Query

Configuração utilizada para gravar a saída de uma consulta.

Conteúdo

TimestreamConfiguration

Configuração necessária para gravar dados no banco de dados e na tabela Timestream.

Tipo: objeto TimestreamConfiguration

Obrigatório: Sim

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

TargetDestination
Serviço: Amazon Timestream Query

Detalhes do destino para gravar dados para uma fonte de dados de destino. A fonte de dados atualmente suportada é Timestream.

Conteúdo

TimestreamDestination

Detalhes do destino do resultado da consulta para a fonte de dados Timestream.

Tipo: objeto TimestreamDestination

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

TimeSeriesDataPoint
Serviço: Amazon Timestream Query

O tipo de dados de série temporal representa os valores de uma medida ao longo do tempo. Uma série temporal é uma matriz de linhas de registros de data e hora e valores de medida, com linhas classificadas em ordem crescente de tempo. A TimeSeriesDataPoint é um único ponto de dados na série temporal. Ele representa uma tupla de (tempo, valor da medida) em uma série temporal.

Conteúdo

Time

A data e hora em que o valor da medida foi coletado.

Tipo: string

Obrigatório: Sim

Value

O valor da medida para o ponto de dados.

Tipo: objeto Datum

Obrigatório: Sim

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

TimestreamConfiguration
Serviço: Amazon Timestream Query

Configuração para gravar dados no banco de dados e na tabela Timestream. Essa configuração permite que o usuário mapeie as colunas de seleção de resultados da consulta nas colunas da tabela de destino.

Conteúdo

DatabaseName

Nome do banco de dados do Timestream no qual o resultado da consulta será gravado.

Tipo: string

Obrigatório: Sim

DimensionMappings

Isso permite o mapeamento de coluna(s) do resultado da consulta para uma dimensão na tabela de destino.

Tipo: matriz de objetos DimensionMapping

Obrigatório: Sim

TableName

Nome da tabela Timestream na qual o resultado da consulta será gravado. A tabela deve estar dentro do mesmo banco de dados fornecido na configuração de Timestream.

Tipo: string

Obrigatório: Sim

TimeColumn

Coluna do resultado da consulta que deve ser utilizada como coluna de tempo na tabela de destino. O tipo de coluna para isso deve serTIMESTAMP.

Tipo: string

Obrigatório: Sim

MeasureNameColumn

O nome da coluna de medida.

Tipo: string

Obrigatório: Não

MixedMeasureMappings

Especifica como mapear medidas para registros de várias medidas.

Tipo: Matriz de objetos MixedMeasureMapping

Membros da matriz: número mínimo de 1 item.

Obrigatório: Não

MultiMeasureMappings

Mapeamentos de várias medidas.

Tipo: objeto MultiMeasureMappings

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

TimestreamDestination
Serviço: Amazon Timestream Query

Destino para consulta agendada.

Conteúdo

DatabaseName

Nome do banco de dados Timestream.

Tipo: string

Obrigatório: Não

TableName

Nome da tabela Timestream.

Tipo: string

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

Type
Serviço: Amazon Timestream Query

Contém o tipo de dados de uma coluna em um conjunto de resultados de consulta. O tipo de dados pode ser escalar ou complexo. Os tipos de dados escalares suportados são inteiros, booleanos, string, double, timestamp, data, hora e intervalos. Os tipos de dados complexos compatíveis são matrizes, linhas e séries temporais.

Conteúdo

ArrayColumnInfo

Indica se a coluna é uma matriz.

Tipo: objeto ColumnInfo

Obrigatório: Não

RowColumnInfo

Indica se a coluna é uma linha.

Tipo: matriz de objetos ColumnInfo

Obrigatório: Não

ScalarType

Indica se a coluna é do tipo string, inteiro, booleano, duplo, timestamp, data, hora. Para obter mais informações, consulte Tipos de dados compatíveis.

Tipo: string

Valores Válidos: `VARCHAR | BOOLEAN | BIGINT | DOUBLE | TIMESTAMP | DATE | TIME | INTERVAL_DAY_TO_SECOND | INTERVAL_YEAR_TO_MONTH | UNKNOWN | INTEGER`

Obrigatório: Não

TimeSeriesMeasureValueColumnInfo

Indica se a coluna é do tipo de dados de série temporal.

Tipo: objeto ColumnInfo

Obrigatório: Não

Consulte também

Para obter mais informações sobre como usar isso API em um idioma específico AWS SDKs, consulte o seguinte:

- AWS SDKpara C++
- AWS SDKpara Java V2
- AWS SDKpara Ruby V3

Erros comuns

Esta seção lista os erros comuns às API ações de todos os AWS serviços. Para erros específicos de uma API ação desse serviço, consulte o tópico dessa API ação.

AccessDeniedException

Você não tem acesso suficiente para executar essa ação.

HTTPCódigo de status: 400

IncompleteSignature

A assinatura da solicitação não está em conformidade com os AWS padrões.

HTTPCódigo de status: 400

InternalFailure

O processamento da solicitação falhou por causa de um erro, uma exceção ou uma falha desconhecida.

HTTPCódigo de status: 500

InvalidAction

A ação ou operação solicitada é inválida. Verifique se a ação foi digitada corretamente.

HTTPCódigo de status: 400

InvalidClientTokenId

O certificado X.509 ou ID da chave de AWS acesso fornecido não existe em nossos registros.

HTTPCódigo de status: 403

NotAuthorized

Você não tem permissão para realizar esta ação.

HTTPCódigo de status: 400

OptInRequired

O ID da chave de AWS acesso precisa de uma assinatura para o serviço.

HTTPCódigo de status: 403

RequestExpired

A solicitação chegou ao serviço mais de 15 minutos após o carimbo de data na solicitação ou mais de 15 minutos após a data de expiração da solicitação (como para pré-assinadaURLs), ou o carimbo de data na solicitação é mais de 15 minutos no futuro.

HTTPCódigo de status: 400

ServiceUnavailable

Falha na solicitação devido a um erro temporário do servidor.

HTTPCódigo de status: 503

ThrottlingException

A solicitação foi negada devido à limitação da solicitação.

HTTPCódigo de status: 400

ValidationError

A entrada não satisfaz as restrições especificadas por um AWS serviço.

HTTPCódigo de status: 400

Parâmetros gerais

A lista a seguir contém os parâmetros que todas as ações usam para assinar solicitações do Signature versão 4 com uma string de consulta. Todos os parâmetros específicos de uma ação são

listados no tópico para a ação. Para obter mais informações sobre o Signature versão 4, consulte AWS APISolicitações de assinatura no Guia IAM do usuário.

Action

A ação a ser executada.

Tipo: string

Obrigatório: Sim

Version

A API versão para a qual a solicitação foi escrita, expressa no formato YYYY-MM-DD.

Tipo: string

Obrigatório: Sim

X-Amz-Algorithm

O algoritmo de hash que foi usado para criar a assinatura da solicitação.

Condição: especifique esse parâmetro ao incluir informações de autenticação em uma string de consulta em vez de no cabeçalho de HTTP autorização.

Tipo: string

Valores Válidos: `AWS4-HMAC-SHA256`

Obrigatório: Condicional

X-Amz-Credential

O valor de escopo da credencial, uma string que inclui a sua chave de acesso, a data, a região visada, o serviço que está sendo solicitado e uma sequência de encerramento ("aws4_request"). O valor é expresso no seguinte formato: access_key//region YYYYMMDD/service /aws4_request.

Para obter mais informações, consulte Criar uma AWS API solicitação assinada no Guia IAM do usuário.

Condição: especifique esse parâmetro ao incluir informações de autenticação em uma string de consulta em vez de no cabeçalho de HTTP autorização.

Tipo: string

Obrigatório: Condicional

X-Amz-Date

A data usada para criar a assinatura. O formato deve ser o formato básico ISO 8601 (YYYYMMDD'T' 'ZHHMMSS'). Por exemplo, a data e hora a seguir é um X-Amz-Date valor válido:`20120325T120000Z`.

Condição: X-Amz-Date é opcional para todas as solicitações; ela pode ser usada para substituir a data usada para assinar solicitações. Se o cabeçalho da data for especificado no formato básico ISO 8601, não X-Amz-Date é necessário. Quando X-Amz-Date usado, ele sempre substitui o valor do cabeçalho da data. Para obter mais informações, consulte Elementos de uma assinatura de AWS API solicitação no Guia IAM do usuário.

Tipo: string

Obrigatório: Condicional

X-Amz-Security-Token

O token de segurança temporário obtido por meio de uma chamada para AWS Security Token Service (AWS STS). Para obter uma lista de serviços que oferecem suporte a credenciais de segurança temporárias de AWS STS, consulte Serviços da AWS “Trabalhe com” IAM no Guia do IAM usuário.

Condição: se você estiver usando credenciais de segurança temporárias do AWS STS, deverá incluir o token de segurança.

Tipo: string

Obrigatório: Condicional

X-Amz-Signature

Especifica a assinatura com codificação hexadecimal que foi calculada com base na string a ser assinada e na chave de assinatura derivada.

Condição: especifique esse parâmetro ao incluir informações de autenticação em uma string de consulta em vez de no cabeçalho de HTTP autorização.

Tipo: string

Obrigatório: Condicional

X-Amz-SignedHeaders

Especifica todos os HTTP cabeçalhos que foram incluídos como parte da solicitação canônica. Para obter mais informações sobre a especificação de cabeçalhos assinados, consulte Criar uma AWS API solicitação assinada no Guia do IAMusuário.

Condição: especifique esse parâmetro ao incluir informações de autenticação em uma string de consulta em vez de no cabeçalho de HTTP autorização.

Tipo: string

Obrigatório: Condicional

Histórico do documento

Alteração	Descrição	Data
Atualização somente com documentação	O tópico Cotas foi atualizado para separar as cotas padrão e os limites do sistema.	22 de outubro de 2024
O Amazon Timestream agora oferece suporte a insights de consulta	O Timestream agora inclui suporte para o recurso de insights de consulta que ajuda você a otimizar suas consultas , melhorar seu desempenho e reduzir custos.	22 de outubro de 2024
Atualização do Amazon Timestream para InfluxDB para uma política existente.	O Amazon Timestream for InfluxDB adicionou a ação à política `ec2:DescribeRouteTables` gerenciada `AmazonTimestreamInfluxDBFullAccess` existente para descrever suas tabelas de rotas	8 de outubro de 2024

AmazonTimestreamReadOnlyAccess — Atualização de uma política existente	O Timestream for LiveAnalytics adicionou a DescribeAccountSettings permissão à política AmazonTimestreamReadOnlyAccess gerenciada para descrever Conta da AWS as configurações.	3 de junho de 2024
LiveAnalytics Por enquanto, o Amazon Timestream oferece suporte às unidades de computação do Timestream () TCUs	LiveAnalytics Por enquanto, o Amazon Timestream inclui suporte para Timestream Compute Units TCUs () para medir a capacidade computacional alocada para suas necessidades de consulta.	29 de abril de 2024
Novas políticas adicionadas	O Amazon Timestream for InfluxDB adicionou duas novas políticas: uma que permite que o serviço gerencie interfaces de rede e grupos de segurança em sua conta. Para obter mais informações, consulte AmazonTimestreamInfluxDBServiceRolePolicy. Outro que fornece acesso administrativo total para criar, atualizar, excluir e listar instâncias do Amazon Timestream InfluxDB e criar e listar grupos de parâmetros. Para obter mais informações, consulte AmazonTimestreamInfluxDBFullAccess.	14 de março de 2024

O Amazon Timestream para InfluxDB agora está disponível ao público em geral.	Esta documentação abrange o lançamento inicial do Amazon Timestream para InfluxDB.	14 de março de 2024
Os eventos do Amazon Timestream LiveAnalytics for Query estão disponíveis em AWS CloudTrail	LiveAnalytics Por enquanto, o Amazon Timestream publica API eventos de dados de consulta no. AWS CloudTrai l Os clientes podem auditar todas as API solicitações de consulta feitas em suas AWS contas e ver informações como qual IAM usuário/função fez a solicitação, quando a solicitação foi feita, quais bancos de dados e tabelas foram consultados e o ID da consulta da solicitação.	12 de setembro de 2023
Amazon Timestream para LiveAnalytics UNLOAD	LiveAnalytics Por enquanto, o Amazon Timestream UNLOAD suporta a exportação dos resultados da consulta para o S3.	12 de maio de 2023
Amazon Timestream LiveAnalytics para atualização de uma política existente.	Permissões de carregamento em lote adicionadas a uma política gerenciada.	24 de fevereiro de 2023
Amazon Timestream LiveAnalytics para carregamento em lote.	LiveAnalytics Por enquanto, o Amazon Timestream oferece suporte à funcionalidade de carregamento em lote.	24 de fevereiro de 2023
Por enquanto, o Amazon Timestream oferece suporte LiveAnalytics . AWS Backup	Por enquanto, o Amazon Timestream oferece suporte LiveAnalytics . AWS Backup	14 de dezembro de 2022

Amazon Timestream LiveAnalytics para atualizações de políticas gerenciadas AWS	Novas informações sobre políticas AWS gerenciadas e Amazon Timestream LiveAnalytics for, incluindo atualizações de políticas gerenciadas existentes.	29 de novembro de 2021
O Amazon Timestream LiveAnalytics for oferece suporte a consultas programadas	LiveAnalytics Por enquanto, o Amazon Timestream suporta a execução de uma consulta em seu nome, com base em uma programação.	29 de novembro de 2021
Amazon Timestream LiveAnalytics para suporte à loja magnética.	LiveAnalytics Por enquanto, o Amazon Timestream suporta o uso de armazenamento magnético para suas gravações em tabelas.	29 de novembro de 2021
Amazon Timestream LiveAnalytics para registros de várias medidas.	LiveAnalytics Por enquanto, o Amazon Timestream oferece suporte a um formato mais compacto para armazenar seus dados de séries temporais.	29 de novembro de 2021
Amazon Timestream LiveAnalytics para atualizações de políticas gerenciadas AWS	Novas informações sobre políticas AWS gerenciadas e Amazon Timestream LiveAnalytics for, incluindo atualizações de políticas gerenciadas existentes.	24 de maio de 2021

O Amazon Timestream LiveAnalytics for já está disponível na região da Europa (Frankfurt).	O Amazon Timestream LiveAnalytics for agora está disponível ao público em geral na região da Europa (Frankfurt) (). `eu-central-1`	23 de abril de 2021
O Amazon Timestream LiveAnalytics for is agora VPC oferece suporte a endpoints ().AWS PrivateLink	LiveAnalytics Por enquanto, o Amazon Timestream suporta o uso VPC de endpoints ().AWS PrivateLink	23 de março de 2021
O Amazon Timestream agora oferece suporte a consultas entre tabelas.	Você pode usar o Amazon Timestream LiveAnalytics para executar consultas entre tabelas.	10 de fevereiro de 2021
LiveAnalytics Por enquanto, o Amazon Timestream oferece suporte a estatísticas aprimoradas de execução de consultas.	LiveAnalytics Por enquanto, o Amazon Timestream oferece suporte a estatísticas aprimoradas de execução de consultas, como a quantidade de dados escaneados.	10 de fevereiro de 2021
LiveAnalytics Por enquanto, o Amazon Timestream oferece suporte a funções avançadas de séries temporais.	Você pode usar o Amazon Timestream LiveAnalytics para SQL executar consultas com funções avançadas de séries temporais, como derivadas, integrais e correlações.	10 de fevereiro de 2021
O Amazon Timestream LiveAnalytics for HIPAA já ISO está em conformidade com o Amazon Timestream. PCI	Agora você pode usar o Amazon Timestream LiveAnalytics para cargas de trabalho que HIPAA exigem infraestrutura compatível com ISO e. PCI	27 de janeiro de 2021

LiveAnalytics Por enquanto, o Amazon Timestream oferece suporte a telegraf e grafana de código aberto.	Agora você pode usar o Telegraf, o agente de servidor de código aberto e orientado por plug-ins para coletar e reportar métricas, e o Grafana, a plataforma de análise e monitoramento de código aberto para bancos de dados, com o Amazon Timestream for. LiveAnalytics	25 de novembro de 2020
O Amazon Timestream LiveAnalytics for já está disponível ao público em geral.	Esta documentação abrange a versão inicial do Amazon LiveAnalytics Timestream for.	30 de setembro de 2020

O que é Timestream for InfluxDB?

O Amazon Timestream for InfluxDB é um mecanismo gerenciado de banco de dados de séries temporais que facilita que desenvolvedores DevOps e equipes de aplicativos executem bancos de dados do InfluxDB para aplicativos de séries temporais em tempo real usando código aberto. AWS APIs Com o Amazon Timestream para InfluxDB, é fácil configurar, operar e escalar cargas de trabalho de séries temporais que podem responder a consultas com tempo de resposta de consulta de um dígito em milissegundos.

O Amazon Timestream para InfluxDB oferece acesso aos recursos da conhecida versão de código aberto do InfluxDB em sua ramificação 2.x. Isso significa que o código, os aplicativos e as ferramentas que você já usa hoje com seus bancos de dados de código aberto do InfluxDB existentes devem funcionar perfeitamente com o Amazon Timestream for InfluxDB. O Amazon Timestream for InfluxDB pode fazer backup automático do seu banco de dados e manter seu software de banco de dados atualizado com a versão mais recente. Além disso, o Amazon Timestream para InfluxDB facilita o uso da replicação para aprimorar a disponibilidade do banco de dados e melhorar a durabilidade dos dados. Como acontece com todos os AWS serviços, não são necessários investimentos iniciais e você paga apenas pelos recursos que usa.

Instâncias de banco de dados

Uma instância de banco de dados é um ambiente de banco de dados isolado em execução na nuvem. É o alicerce básico do Amazon Timestream para InfluxDB. Uma instância de banco de dados pode conter vários bancos de dados criados pelo usuário (ou organizações e buckets, no caso de bancos de dados InfluxDb 2.x) e pode ser acessada usando as mesmas ferramentas e aplicativos de cliente que você pode usar para acessar uma instância autônoma autogerenciada do InfluxDB. As instâncias de banco de dados são simples de criar e modificar com as ferramentas de linha de AWS comando, as operações do Amazon Timestream API InfluxDB ou o. AWS Management Console

> **Note**
>
> O Amazon Timestream para InfluxDB oferece suporte ao acesso a bancos de dados usando as operações do Influx e a interface do usuário do API Influx. O Amazon Timestream para InfluxDB não permite acesso direto ao host.

Você pode ter até 40 Amazon Timestream para instâncias do InfluxDB.

Cada instância de banco de dados tem um nome de instância de banco de dados. Esse nome fornecido pelo cliente identifica exclusivamente a instância de banco de dados ao interagir com o Amazon Timestream para InfluxDB e comandos. API AWS CLI O nome da instância de banco de dados deve ser exclusivo para esse cliente em uma AWS região.

O nome da instância de banco de dados faz parte do DNS nome do host alocado à sua instância pelo Timestream para o InfluxDB. Por exemplo, se você especificar influxdb1 como o nome da instância de banco de dados, o Timestream alocará automaticamente um endpoint para sua instância. DNS Um exemplo de endpoint é`influxdb1-3ksj4dla5nfjhi.us-east-1.timestream-influxdb.amazonaws.com`, onde `influxdb1` está o nome da sua instância.

No endpoint de exemplo`influxdb1-3ksj4dla5nfjhi.us-east-1.timestream-influxdb.amazonaws.com`, a string `3ksj4dla5nfjhi` é um identificador de conta exclusivo gerado por AWS. O identificador `3ksj4dla5nfjhi` no exemplo não muda para a conta especificada em uma determinada região. Portanto, todas as suas instâncias de banco de dados criadas por essa conta compartilham o mesmo identificador fixo. Considere as seguintes características do identificador fixo:

- Atualmente, o Timestream for InfluxDB não oferece suporte à renomeação de instâncias de banco de dados.
- Se você excluir e recriar uma instância de banco de dados com o mesmo identificador de instância de banco de dados, o endpoint será o mesmo.
- Se você usar a mesma conta para criar uma instância de banco de dados em uma região diferente, o identificador gerado internamente será diferente porque a região é diferente, como em `influxdb2.4a3j5du5ks7md2.us-west-1.timestream-influxdb.amazonaws.com`.

Cada instância de banco de dados suporta apenas um Timestream para o mecanismo de banco de dados InfluxDB.

Ao criar uma instância de banco de dados, o InfluxDB exige que um nome de organização seja especificado. Uma instância de banco de dados pode hospedar várias organizações e vários buckets associados a cada organização.

O Amazon Timestream para InfluxDB permite que você crie uma conta de usuário principal e uma senha para sua instância de banco de dados como parte do processo de criação. Esse usuário principal tem permissões para criar organizações, buckets e realizar operações de leitura, gravação, exclusão e upsert em seus dados. Você também poderá acessar o InfluxUI e recuperar seu token de

operador em seu primeiro login. A partir daí, você também poderá gerenciar todos os seus tokens de acesso. Você deve definir a senha do usuário principal ao criar uma instância de banco de dados, mas pode alterá-la a qualquer momento usando o InfluxAPI, o Influx ou o CLI InfluxUI.

Classes da instância de banco de dados

A classe de instância de banco de dados determina a capacidade computacional e de memória de uma instância de banco de dados `fi UbfkyxDB` Amazon Timestream. A classe de instância de banco de dados da qual você precisa depende dos requisitos de memória e potência de processamento.

Uma classe de instância de banco de dados consiste no tipo de classe e no tamanho de instância de banco de dados. Por exemplo, `db.influx` é um tipo de classe de instância de banco de dados otimizado para memória, adequado aos requisitos de memória de alto desempenho relacionados à execução InfluxDb de cargas de trabalho. Dentro do tipo de classe de `db.influx` instância, `db.influx.2xlarge` está uma classe de instância de banco de dados. O tamanho dessa classe é 2xlarge.

Para obter mais informações sobre preços de classes de instância, consulte os preços do Amazon Timestream para InfluxDB.

Tipos de classe de instância de banco de dados

O Amazon Timestream para InfluxDB oferece suporte a classes de instância de banco de dados para o seguinte caso de uso otimizado para casos de uso do InfluxDB.

- **`db.influx`**—Essas classes de instância são ideais para executar cargas de trabalho com uso intenso de memória em bancos de dados InfluxDB de código aberto

Especificações de hardware para classes de instância de banco de dados

A terminologia a seguir descreve as especificações de hardware para classes de instâncias de banco de dados:

- v CPU

O número de unidades de processamento central virtual (CPUs). Um virtual CPU é uma unidade de capacidade que você pode usar para comparar classes de instâncias de banco de dados.

- Memória (GiB)

  ORAM, em gibibytes, alocado para a instância de banco de dados. Freqüentemente, há uma proporção consistente entre memória e CPU v. Como exemplo, veja a classe de instância db.influx, que tem uma CPU relação memória/v semelhante à classe de instância EC2 r7g.

- Otimizado para influxo

  A instância de banco de dados usa uma pilha de configuração otimizada e fornece capacidade adicional dedicada para E/S do Amazon EBS. Essa otimização proporciona a melhor performance para seus volumes do EBS ao minimizar a contenção entre a E/S e outro tráfego de sua instância.

- Largura de banda da rede

  A velocidade da rede em relação a outras classes de instância de banco de dados. Na tabela a seguir, você pode encontrar detalhes de hardware sobre as classes de instância Amazon Timestream para InfluxDB.

Classe de instâncias	v CPU	Memória (GiB)	Tipo de armazenamento	Largura de banda da rede (Gbps)
db.influx.medium	1	8	Influxo incluído IOPS	10
db.influx.large	2	16	Influxo incluído IOPS	10
db.influx.xlarge	4	32	Influxo incluído IOPS	10
db.influx.2xlarge	8	64	Influxo incluído IOPS	10
db.influx.4xlarge	16	128	Influxo incluído IOPS	10

Classe de instâncias	v CPU	Memória (GiB)	Tipo de armazenamento	Largura de banda da rede (Gbps)
db.influx.8xlarge	32	256	Influxo incluído IOPS	12
db.influx .12xlarge	48	384	Influxo incluído IOPS	20
db.influx .16xlarge	64	512	Influxo incluído IOPS	25

Armazenamento de instâncias do InfluxDB

As instâncias de banco de dados do Amazon Timestream for InfluxDB usam volumes incluídos do IOPS Influx para bancos de dados e armazenamento de logs.

Em alguns casos, a carga de trabalho do banco de dados pode não conseguir atingir 100% do IOPS que você provisionou. Para ter mais informações, consulte Fatores que afetam a performance do armazenamento. Para obter mais informações sobre os preços de armazenamento do Timestream for InfluxDB, consulte os preços do Amazon Timestream.

Amazon Timestream para tipos de armazenamento do InfluxDB

O Amazon Timestream para InfluxDB fornece suporte para um tipo de armazenamento, o Influx Included. IOPS Você pode criar Timestream para instâncias do InfluxDB com até 16 tebibytes (TiB) de armazenamento.

Aqui está uma breve descrição do tipo de armazenamento disponível:

- Armazenamento incluído no Influx IO: o desempenho do armazenamento é a combinação de operações de E/S por segundo (IOPS) e a rapidez com que o volume de armazenamento pode realizar leituras e gravações (taxa de transferência de armazenamento). Nos volumes de armazenamento IOPS incluídos no Influx, o Amazon Timestream for InfluxDB fornece 3 níveis de armazenamento que vêm pré-configurados IOPS com a taxa de transferência ideal necessária para diferentes tipos de cargas de trabalho.

Dimensionamento de instâncias do InfluxDB

A configuração ideal de uma instância Timestream for InfluxDB depende de muitos fatores que incluem taxa de ingestão, tamanhos de lotes, cardinalidade de séries temporais, consultas simultâneas e tipos de consulta. Em um esforço para fornecer recomendações de dimensionamento, estamos nos concentrando em uma carga de trabalho exemplar com as seguintes características:

- Os dados são coletados e gravados por uma frota de agentes do Telegraf que coletam sistemaCPU, memória, disco, E/S etc. de um data center.

  Cada solicitação de gravação contém 5000 linhas.
- Os tipos de consultas executadas no sistema são categorizados como consultas de “complexidade moderada”. Essa categoria de consultas apresenta as seguintes características:
  - Tenha várias funções e uma ou duas expressões regulares
  - Também pode agrupar por cláusulas ou experimentar um intervalo de tempo de várias semanas.
  - Normalmente, leva de algumas centenas de milissegundos a alguns milhares de milissegundos para ser executado.
  - CPUfavorece principalmente o desempenho da consulta.

Classe de instância	Tipo de armazenamento	Escreve (linhas por segundo)	Leituras (consultas por segundo)
db.influx.large	Influx IO incluído: 3K	~50.000	<10
db.influx.2xlarge	Influx IO incluído: 3K	~150.000	<25
db.influx.4xlarge	Influx IO incluído: 3K	~200.000	~25
db.influx.4xlarge	Influx IO incluído: 12K	~250.000	~35
db.influx.8xlarge	Influx IO incluído: 12K	~500.000	~50
db.influx.12xlarge	Influx IO incluído: 12K	<750.000	<100

AWS Regiões e zonas de disponibilidade

Os recursos de computação em nuvem da Amazon são hospedados em vários locais no mundo todo. Esses locais são compostos por AWS regiões e zonas de disponibilidade. Cada AWS região é uma área geográfica separada. Cada AWS região tem vários locais isolados, conhecidos como zonas de disponibilidade.

> **Note**
>
> Para obter informações sobre como encontrar as zonas de disponibilidade para uma AWS região, consulte Regiões e zonas no Guia EC2 do usuário da Amazon.

O Amazon Timestream for InfluxDB permite que você coloque recursos, como instâncias de banco de dados e dados em vários locais.

A Amazon opera state-of-the-art centros de dados de alta disponibilidade. Embora sejam raras, podem ocorrer falhas que afetam a disponibilidade das instâncias de banco de dados que estiverem no mesmo local. Se você hospeda todas as instâncias de banco de dados em um único local afetado por essa falha, nenhuma delas estará disponível.

É importante lembrar que cada AWS região é completamente independente. Qualquer atividade do Amazon Timestream para InfluxDB que você iniciar (por exemplo, criar instâncias de banco de dados ou listar instâncias de banco de dados disponíveis) é executada somente em sua região padrão atual. AWS A região da AWS padrão pode ser alterada no console ou definindo a variável de ambiente `AWS_DEFAULT_REGION`. Ou ele pode ser substituído usando o `--region` parâmetro com o AWS Command Line Interface ()AWS CLI. Para obter mais informações, consulte Configurando o AWS Command Line Interface, especificamente as seções sobre variáveis de ambiente e opções de linha de comando.

Para criar ou trabalhar com uma instância de banco de dados Amazon Timestream para InfluxDB em uma região AWS específica, use o endpoint de serviço regional correspondente.

AWS Disponibilidade da região

Para obter mais informações sobre AWS as regiões em que o Amazon Timestream para InfluxDB está disponível atualmente e o endpoint para cada região, consulte Pontos de extremidade e cotas do Amazon Timestream.

AWS Design de regiões

Cada AWS região foi projetada para ser isolada das outras AWS regiões. Esse design proporciona o máximo de tolerância a falhas e estabilidade possível.

Ao visualizar seus recursos, você vê somente os recursos vinculados à AWS região que você especificou. Isso ocorre porque AWS as regiões estão isoladas umas das outras e não replicamos automaticamente os recursos entre AWS regiões.

AWS Zonas de disponibilidade

Quando você cria uma instância de banco de dados, o Amazon Timestream for InfluxDB escolhe uma para você aleatoriamente com base na configuração da sua sub-rede. Uma zona de disponibilidade é representada por um código de AWS região seguido por um identificador de letra (por exemplo,`us-east-1a`).

Use o EC2 comando `describe-availability` -zones Amazon da seguinte forma para descrever as zonas de disponibilidade dentro da região especificada que estão habilitadas para sua conta.

```
aws ec2 describe-availability-zones --region region-name
```

Por exemplo, para descrever as zonas de disponibilidade na região Leste dos EUA (Norte da Virgínia) (us-east-1) que estão habilitadas para sua conta, execute o seguinte comando:

```
aws ec2 describe-availability-zones --region us-east-1
```

Você não pode escolher as zonas de disponibilidade para as instâncias de banco de dados primária e secundária em uma implantação de banco de dados Multi-AZ. O Amazon Timesteram para InfluxDB os escolhe aleatoriamente para você. Para obter mais informações sobre implantações Multi-AZ, consulte.. Configurando e gerenciando uma implantação Multi-AZ

Cobrança de instância de banco de dados para Amazon Timestream para InfluxDB

As instâncias do Amazon Timestream para InfluxDB são cobradas com base nos seguintes componentes:

- Horas da instância de banco de dados (por hora) — Com base na classe de instância de banco de dados da instância de banco de dados, por exemplo, db.influx.large. A definição de preço

está listada em uma base por hora, mas é calculada em segundos e mostra o tempo no formato decimal. O uso do Amazon Timestream para InfluxDB é cobrado em incrementos de 1 segundo, com um mínimo de 10 minutos. Para obter mais informações, consulte Classes de Classes da instância de banco de dados instância de banco de dados.

- Armazenamento (por GiB por mês) — Capacidade de armazenamento que você provisionou para sua instância de banco de dados. Para obter mais informações, consulte Armazenamento de instâncias do InfluxDB.
- Transferência de dados (por GB) — Transferência de dados para dentro e para fora da sua instância de banco de dados de ou para a Internet e outras AWS regiões.

Para obter informações sobre preços do Amazon Timestream para InfluxDB, consulte a página de preços do Amazon Timestream para InfluxDB.

Configurando o Amazon Timestream para InfluxDB

Antes de usar o Amazon Timestream para InfluxDB pela primeira vez, conclua as seguintes tarefas:

Se você já tem uma AWS conta, conheça seus requisitos do Amazon Timestream para InfluxDB e prefira usar os padrões e IAM VPC Introdução ao Timestream for InfluxDB começar a usar o Amazon Timestream para InfluxDB.

Cadastre-se para uma AWS conta

Se você não tiver uma AWS conta, conclua as etapas a seguir para criar uma.

Para se inscrever em uma AWS conta

- Acesse a página de AWS login.
- Escolha Criar uma nova conta e siga as instruções.

> **Note**
>
> Parte do procedimento de inscrição envolve receber uma chamada telefônica e inserir um código de verificação no teclado do telefone.

Quando você se inscreve AWS em uma conta, um usuário raiz da AWS conta é criado. O usuário root tem acesso a todos os AWS serviços e recursos da conta. Como prática recomendada de

segurança, atribua acesso administrativo a um usuário administrativo e use somente o usuário raiz para realizar as tarefas que exigem acesso do usuário raiz.

AWS envia um e-mail de confirmação após a conclusão do processo de inscrição. A qualquer momento, você pode visualizar a atividade atual da sua conta e gerenciar sua conta acessando https://aws.amazon.com/e escolhendo Minha conta.

Gerenciamento de usuários

Criar um usuário administrativo

Criar um usuário administrador

Depois de se inscrever AWS em uma conta, crie um usuário administrativo para não usar o usuário root nas tarefas diárias.

Proteja o usuário root da sua AWS conta

Faça login no AWS Management Console como proprietário da conta, escolhendo Usuário raiz e inserindo o endereço de e-mail AWS da sua conta. Na próxima página, insira sua senha. Para obter ajuda para fazer login usando o usuário root, consulte Como fazer login como usuário root no Guia do usuário AWS de login

Ative a autenticação multifator (MFA) para seu usuário root. Para obter instruções, consulte Habilitar um MFA dispositivo virtual para o usuário raiz da sua AWS conta (console) no Guia IAM do usuário.

Conceder acesso programático

Os usuários precisam de acesso programático se quiserem interagir com pessoas AWS fora do AWS Management Console. A forma de conceder acesso programático depende do tipo de usuário que está acessando a AWS.

Para conceder acesso programático aos usuários, escolha uma das seguintes opções:

Qual usuário precisa de acesso programático?	Para	Por
Identidade da força de trabalho (usuários gerenciados no IAM Identity Center)	Use credenciais temporári as para assinar solicitações programáticas para o AWS	Seguindo as instruções da interface que você deseja

Qual usuário precisa de acesso programático?	Para	Por
	CLI AWS SDKs, ou. AWS APIs	usar.* Para o AWS CLI, consulte Configurando o AWS IAM Identity AWS CLI Center para usar no Guia do Usuário da Interface de Linha de ComandoAWS .* Para AWS SDKs, ferrament as e AWS APIs, consulte IAMautenticação do Identity Center no AWSSDKsGuia de referência de ferramentas e ferramentas.
IAM	Use credenciais temporári as para assinar solicitações programáticas para o AWS CLISDKs, e. APIs	Siga as instruções em Uso de credenciais temporárias com AWS recursos no Guia do IAM usuário.

Qual usuário precisa de acesso programático?	Para	Por
IAM	(Não recomendado) Use credenciais de longo prazo para assinar solicitações programáticas para AWS CLISDKs, e. APIs	Seguindo as instruções da interface que você deseja usar. Para o AWS CLI, consulte Autenticação usando credenciais IAM de usuário no AWS Guia do usuário da interface de linha de comando .Para ferramentas AWS SDKs e ferramentas, consulte Autenticar usando credenciais de longo prazo no AWS SDKsGuia de referência de ferramentas e ferramentas .Para AWS APIs, consulte Gerenciando chaves de acesso para IAM usuários no IAMGuia do usuário.

Determinar requisitos

O componente básico do Amazon Timestream for Influx é a instância de banco de dados. Em uma instância de banco de dados, você cria seus buckets. Uma instância de banco de dados fornece um endereço de rede chamado de endpoint. Seus aplicativos usam o endpoint para se conectar à instância de banco de dados. Você também acessará seu InfluxUI usando esse mesmo endpoint do seu navegador. Ao criar uma instância de banco de dados, você especifica detalhes

como armazenamento, memória, mecanismo e versão do banco de dados, configuração de rede e segurança. Você controla o acesso de rede a uma instância de banco de dados por meio de um grupo de segurança.

Antes de criar uma instância de banco de dados e um grupo de segurança, você precisa conhecer as necessidades de sua instância de banco de dados e de sua rede. Veja aqui alguns fatores importantes a considerar:

- Requisitos de recursos — Quais são os requisitos de memória e processador para seu aplicativo ou serviço? Você usa essas configurações para ajudá-lo a determinar que classe de instância de banco de dados deve usar. Para especificações sobre classes de instância de banco de dados, consulte Classes de instância de banco de dados.
- VPCe grupo de segurança — Sua instância de banco de dados provavelmente estará em uma nuvem privada virtual (VPC). Para se conectar à sua instância de banco de dados, você precisa definir regras de grupo de segurança. Essas regras são configuradas de forma diferente, dependendo do tipo de VPC uso e da forma como você o usa. Por exemplo, você pode usar: um padrão VPC ou definido pelo usuárioVPC.

  A lista a seguir descreve as regras para cada VPC opção:

  - Padrão VPC — Se sua AWS conta tiver um padrão VPC na AWS região atual, ela VPC está configurada para oferecer suporte a instâncias de banco de dados. Se você especificar o padrão VPC ao criar a instância de banco de dados, certifique-se de criar um grupo de VPC segurança que autorize conexões do aplicativo ou serviço com a instância de banco de dados Amazon Timestream for InfluxDB. Use a opção Grupo de Segurança no VPC console ou no AWS CLI para criar grupos VPC de segurança. Para obter mais informações, consulte Etapa 3: Criar um grupo VPC de segurança.
- Definido pelo usuário VPC — Se você quiser especificar um definido pelo usuário VPC ao criar uma instância de banco de dados, esteja ciente do seguinte:
  - Certifique-se de criar um grupo de VPC segurança que autorize conexões do aplicativo ou serviço com a instância de banco de dados Amazon Timestream for InfluxDB. Use a opção Grupo de Segurança no VPC console ou no AWS CLI para criar grupos VPC de segurança. Para obter informações, consulte Etapa 3: Criar um grupo VPC de segurança.
  - Eles VPC devem atender a certos requisitos para hospedar instâncias de banco de dados, como ter pelo menos duas sub-redes, cada uma em uma zona de disponibilidade separada. Para obter informações, consulte Amazon VPC VPCs e Amazon Timestream para InfluxDB.

- Alta disponibilidade — Você precisa de suporte de failover? No Amazon Timestream para InfluxDB, uma implantação Multi-AZ cria uma instância de banco de dados primária e uma instância de banco de dados secundária em espera em outra zona de disponibilidade para suporte de failover. Para manter a alta disponibilidade, recomendamos as implantações multi-AZ para cargas de trabalho de produção. Para fins de desenvolvimento e teste, você pode usar uma implantação que não seja multi-AZ. Para obter mais informações, consulte Implantações de instâncias de banco de dados multi-AZ.
- IAMpolíticas — Sua AWS conta tem políticas que concedem as permissões necessárias para realizar operações do Amazon Timestream para InfluxDB? Se você estiver se conectando AWS usando IAM credenciais, sua IAM conta deve ter IAM políticas que concedam as permissões necessárias para realizar as operações do plano de controle do Amazon Timestream para o InfluxDB. Para obter mais informações, consulte Identity and Access Management para Amazon Timestream para InfluxDB.
- Portas abertas — Qual porta TCP /IP seu banco de dados escuta? O firewall de algumas empresas pode bloquear conexões com a porta padrão para o seu mecanismo de banco de dados. O padrão para Timestream for InfluxDB é 8086.
- AWS Região — Em qual AWS região você deseja seu banco de dados? Ter o banco de dados próximo do aplicativo ou do serviço Web pode reduzir a latência da rede. Para obter mais informações, consulte AWS Regiões e zonas de disponibilidade.
- Subsistema de disco de banco de dados — Quais são seus requisitos de armazenamento? O Amazon Timestream for InfluxDB fornece três configurações para o tipo de armazenamento incluído no Influx: IOPS
  - Influx Io incluído: 3k () IOPS SSD
  - Influx Io incluído: 12k () IOPS SSD
  - Influx Io incluído: 25k () IOPS SSD

  Para obter mais informações sobre o Amazon Timestream para armazenamento do InfluxDB, consulte Amazon Timestream para armazenamento de instâncias de banco de dados do InfluxDB. Quando você tiver as informações necessárias para criar o grupo de segurança e a instância de banco de dados, continue na próxima etapa.

Forneça acesso à sua instância de banco de dados no seu VPC criando um grupo de segurança

VPCgrupos de segurança fornecem acesso às instâncias de banco de dados em VPC a. Eles atuam como um firewall para a instância de banco de dados associada, controlando o tráfego de entrada e de saída no nível da instância de banco de dados. As instâncias de bancos de dados são criadas por padrão com um firewall e um grupo de segurança padrão que protege a instância de banco de dados.

Para conseguir se conectar à sua instância de banco de dados, você deve adicionar regras a um grupo de segurança que permita que você se conecte. Use suas informações de rede e configuração para criar regras e permitir acesso à sua instância de banco de dados.

Por exemplo, suponha que você tenha um aplicativo que acessa um banco de dados na sua instância de banco de dados em umVPC. Nesse caso, você deve adicionar uma TCP regra personalizada que especifique o intervalo de portas e os endereços IP que seu aplicativo usa para acessar o banco de dados. Se você tiver um aplicativo em uma EC2 instância da Amazon, poderá usar o grupo de segurança que você configurou para a EC2 instância da Amazon.

Criando um grupo de segurança para VPC acesso

Para criar um grupo VPC de segurança, faça login no AWS Management Console e escolha VPC.

> **Note**
>
> Verifique se você está no VPC console, não no console Amazon Timesteam para InfluxDB.

- No canto superior direito do AWS Management Console, escolha a AWSregião em que você deseja criar seu grupo de VPC segurança e instância de banco de dados. Na lista de VPC recursos da Amazon para essa AWS região, você deve ver pelo menos uma VPC e várias sub-redes. Caso contrário, você não tem um padrão VPC nessa AWS região. .
- No painel de navegação, escolha Grupos de segurança.
- Escolha Create grupo de segurança (Criar grupo de segurança).
- Na seção Detalhes básicos da página do grupo de segurança, insira o nome e a Descrição do grupo de segurança. Para VPC, escolha aquela em VPC que você deseja criar sua instância de banco de dados.

- Em Inbound rules (Regras de entrada), escolha Add rule (Adicionar regra).
  - Em Tipo, escolha Personalizado TCP.
  - Em Source, escolha um nome de grupo de segurança ou insira o intervalo de endereços IP (CIDRvalor) de onde você acessa a instância de banco de dados. Se você selecionar My IP (Meu IP), isso concederá acesso à instância de banco de dados do endereço IP detectado no navegador.

  Em Source, escolha um nome de grupo de segurança ou digite o intervalo de endereços IP (CIDRvalor) de onde você acessa a instância de banco de dados. Se você selecionar My IP (Meu IP), isso concederá acesso à instância de banco de dados do endereço IP detectado no navegador.
- (Opcional) Em Outbound rules (Regras de saída), adicione regras para o tráfego de saída. Por padrão, todo tráfego de saída é permitido.
- Escolha Create grupo de segurança (Criar grupo de segurança).

Você pode usar esse grupo VPC de segurança como o grupo de segurança da sua instância de banco de dados ao criá-la.

> **Note**
>
> Se você usar um padrão VPC, um grupo de sub-redes padrão abrangendo todas as sub-redes será criado para você. VPC Ao criar uma instância de banco de dados, você pode escolher o eiifccntf padrão VPC e escolher o padrão para o grupo de sub-redes de banco de dados.

Assim que completar os requisitos de configuração, você poderá criar uma instância de banco de dados usando seus requisitos e grupo de segurança. Para isso, siga as declarações em Criar uma instância de banco de dados.

Introdução ao Timestream for InfluxDB

Nos exemplos a seguir, você pode descobrir como criar e se conectar a uma instância de banco de dados usando o Amazon Timestream for InfluxDB Service.

> **Note**
>
> É necessário concluir as tarefas em Configurando o Amazon Timestream para InfluxDB antes de criar ou se conectar a uma instância de banco de dados.

Tópicos

- Criando e conectando-se a uma instância Timestream for InfluxDB
- Criando um novo token de operador para sua instância do InfluxDB

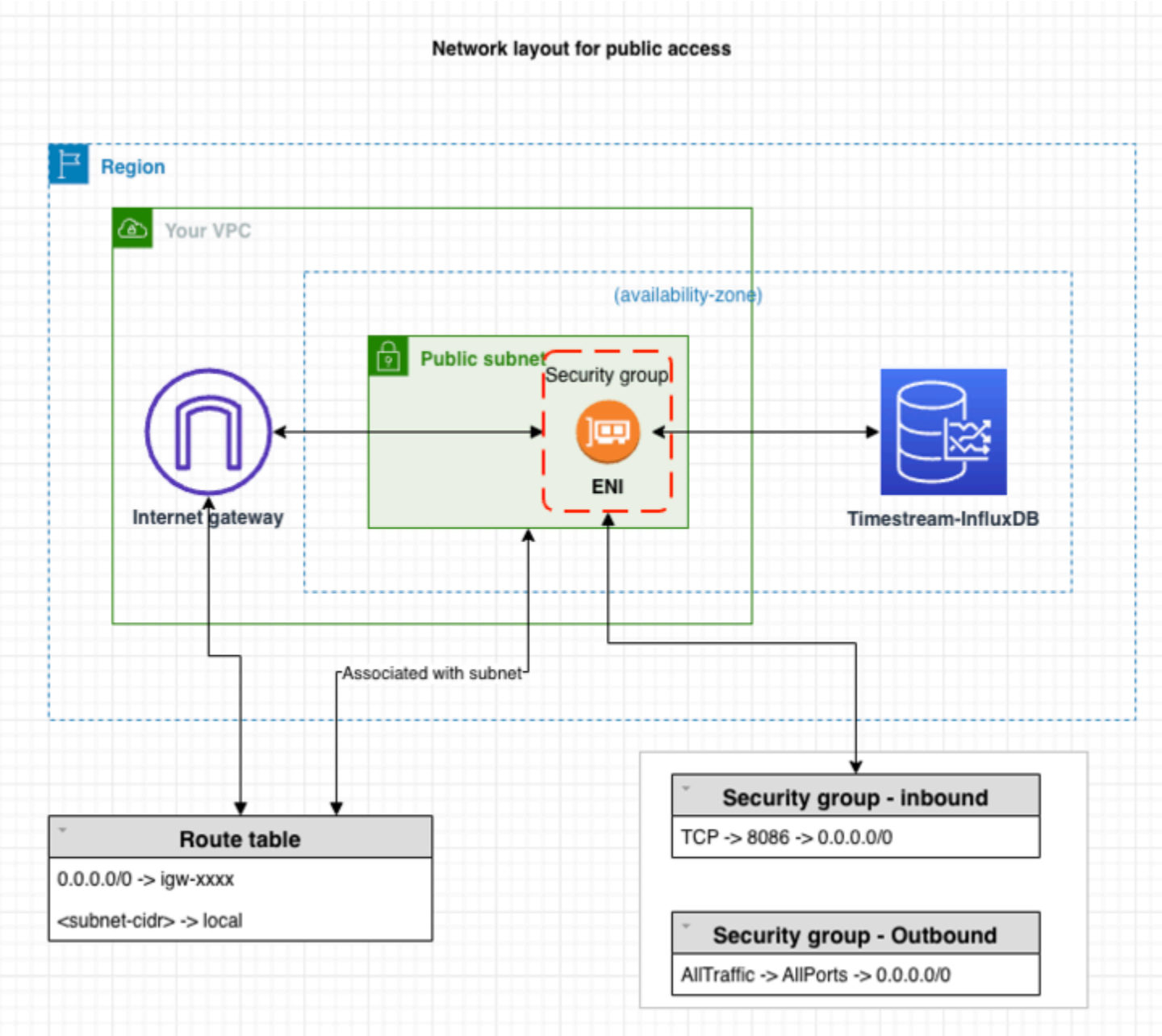
Criando e conectando-se a uma instância Timestream for InfluxDB

Este tutorial cria uma EC2 instância Amazon e uma instância de banco de dados Amazon Timestream para InfluxDB. O tutorial mostra como gravar dados na instância de banco de dados a partir da EC2 instância usando o cliente Telegraf. Como prática recomendada, este tutorial cria uma instância de banco de dados privada em uma nuvem privada virtual (VPC). Na maioria dos casos, outros recursos na mesma instânciaVPC, como EC2 instâncias, podem acessar a instância de banco de dados, mas recursos externos não VPC podem acessá-la.

Depois de concluir o tutorial, há uma sub-rede pública e privada em cada zona de disponibilidade da suaVPC. Em uma zona de disponibilidade, a EC2 instância está na sub-rede pública e a instância de banco de dados está na sub-rede privada.

> **Note**
>
> Não há cobrança pela criação de uma AWS conta. No entanto, ao concluir este tutorial, você poderá incorrer em custos com os AWS recursos que usa. Se esses recursos não forem mais necessários após a conclusão do tutorial, você poderá excluí-los.

O diagrama a seguir mostra a configuração quando a acessibilidade é pública.



Warning

Não recomendamos usar 0.0.0.0/0 para HTTP acesso, pois você possibilita que todos os endereços IP acessem sua instância pública do InfluxDB via. HTTP Essa abordagem nem mesmo é aceitável por um curto período em um ambiente de teste. Autorize apenas um endereço IP específico ou um intervalo de endereços para acessar suas instâncias do InfluxDB usando being HTTP for WebUI ou access. API

Este tutorial cria uma instância de banco de dados executando o InfluxDB com o. AWS Management Console Vamos nos concentrar apenas no tamanho da instância de banco de dados e no identificador da instância de banco de dados. Usaremos as configurações padrão para as outras opções de configuração. A instância de banco de dados criada por esse exemplo será privada.

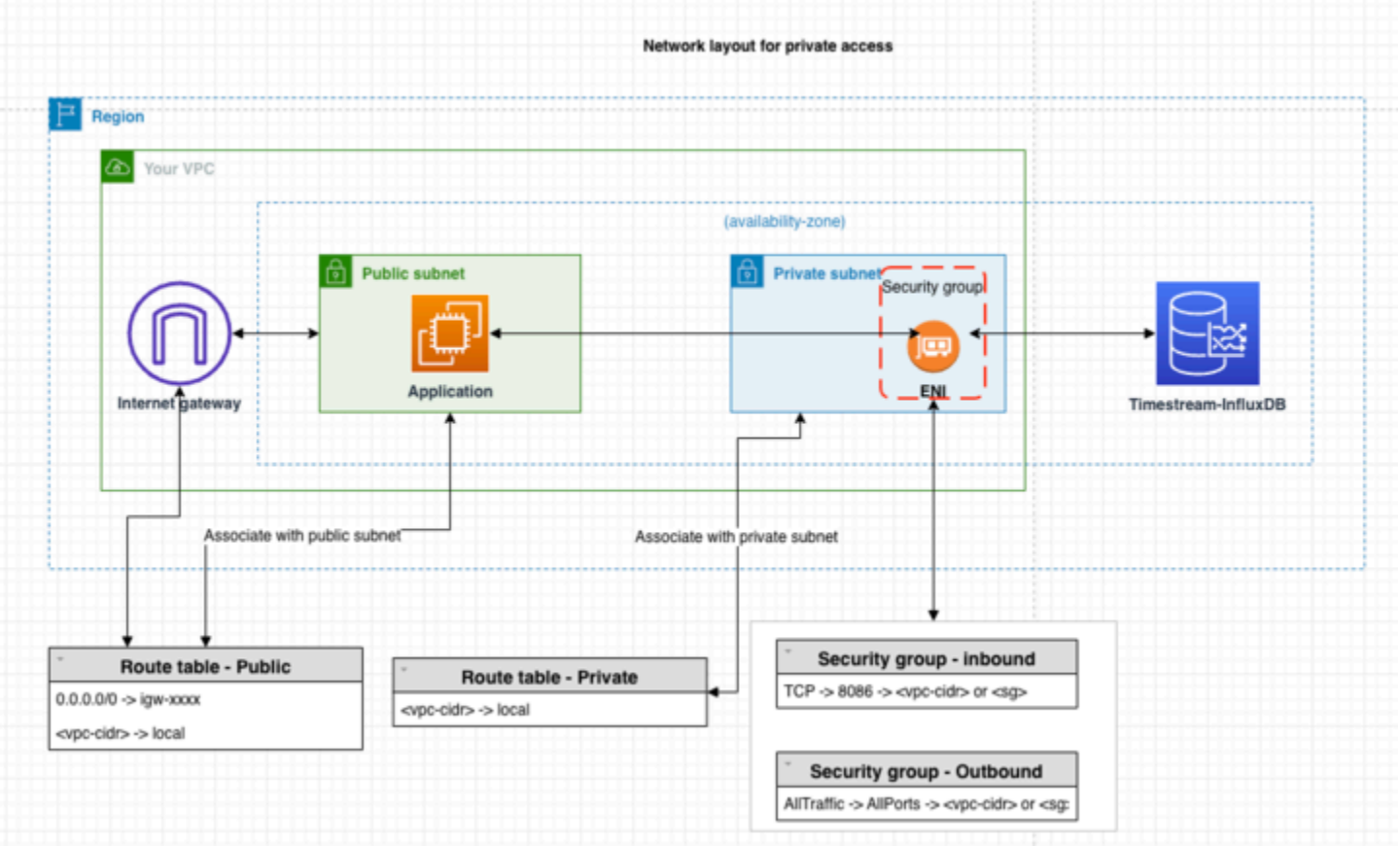
Outras configurações que você pode definir incluem disponibilidade, segurança e registro. Para criar uma instância de banco de dados pública, você deve escolher tornar sua instância “acessível publicamente” na seção Configuração de conectividade. Para obter informações sobre a criação de instâncias de banco de dados, consulteCriar uma instância de banco de dados..

Se sua instância não estiver acessível publicamente, faça o seguinte:

- Crie um host na instância por meio VPC do qual você pode encapsular o tráfego.
- Configure o tunelamento ssh para a instância. Para obter mais informações, consulte Encaminhamento de portas de EC2 instâncias da Amazon com AWS Systems Manager
- Para que o certificado funcione, adicione a seguinte linha ao `/etc/hosts` arquivo da sua máquina cliente:`127.0.0.1`. Esse é o DNS endereço da sua instância.
- Conecte-se à sua instância usando o nome de domínio totalmente qualificado, por exemplo, https: //< DNS >:8086.

> **Note**
>
> O localhost não consegue validar o certificado porque o localhost não faz parte do certificado. SAN

O diagrama a seguir mostra a configuração quando a acessibilidade é privada:



Pré-requisitos

Antes de começar, conclua as etapas nas seguintes seções:

- Cadastre-se em uma AWS conta.
- Crie um usuário administrativo.

Etapa 1: criar uma EC2 instância da Amazon

Crie uma EC2 instância da Amazon que você usará para se conectar ao seu banco de dados.

1. Faça login no AWS Management Console e abra o EC2 console da Amazon em https://console.aws.amazon.com/ec2/.
2. No canto superior direito do AWS Management Console, escolha a AWS região na qual você deseja criar a EC2 instância.
3. Escolha EC2Dashboard e, em seguida, escolha Launch instance.

4. Quando a página Iniciar uma instância for aberta, escolha as seguintes configurações na página Iniciar uma instância.

   a. Em Nome e tags, em Nome, insira ec2-database-connect.

   b. Em Imagens do aplicativo e do sistema operacional (Amazon Machine Image), escolha Amazon Linux e, em seguida, escolha Amazon Linux 2023AMI. Mantenha as seleções padrão nas outras opções.

   c. Em Instance type (Tipo de instância), escolha t2.micro.

   d. Em Key pair (login) (Par de chaves (login)), escolha um Key pair name (Nome do par de chaves) para usar um par de chaves existente. Para criar um novo par de chaves para a EC2 instância da Amazon, escolha Create new key pair e use a janela Create key pair para criá-lo. Para obter mais informações sobre a criação de um novo par de chaves, consulte Criar um par de chaves no Guia EC2 do usuário da Amazon para instâncias Linux.

   e. Em Permitir SSH tráfego nas configurações de rede, escolha a origem das SSH conexões com a EC2 instância. Você pode escolher Meu IP se o endereço IP exibido estiver correto para SSH conexões. Caso contrário, você pode determinar o endereço IP a ser usado para se conectar às EC2 instâncias VPC usando o Secure Shell (SSH). Para determinar seu endereço IP público, em uma janela ou guia diferente do navegador, você pode usar o serviço em https://checkip.amazonaws.com. Um exemplo de endereço IP é 192.0.2.1/32. Em muitos casos, você pode se conectar por meio de um provedor de serviços de Internet (ISP) ou por trás do firewall sem um endereço IP estático. Em caso afirmativo, determine o intervalo de endereços IP utilizado por computadores cliente.

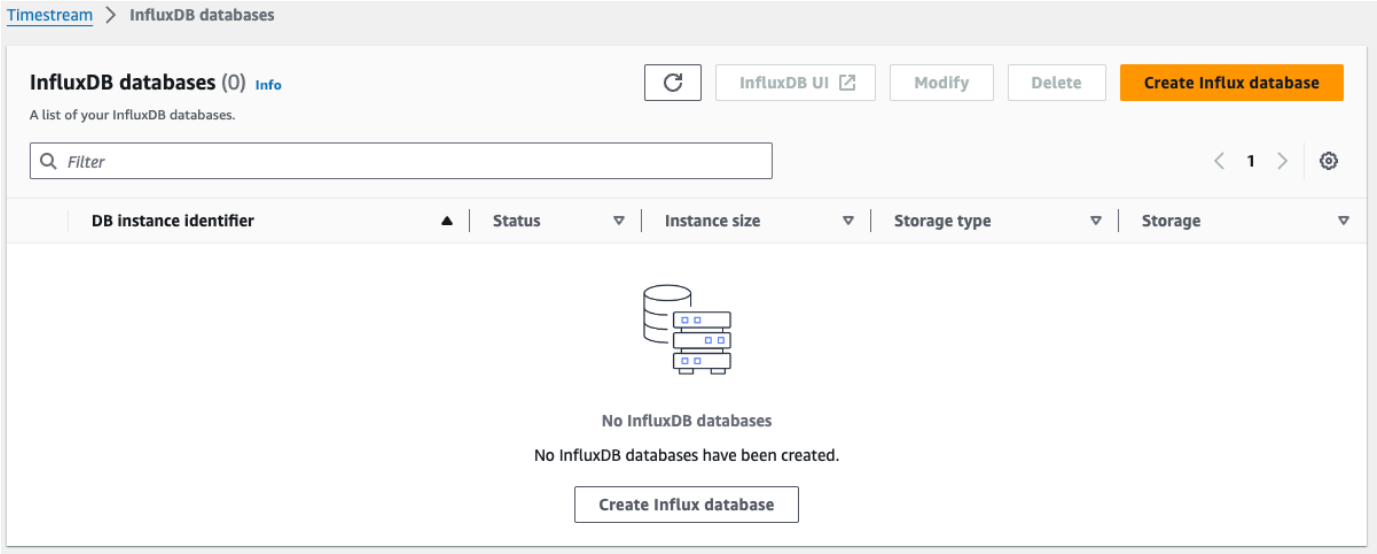
   > ⚠ Warning
   >
   > Não recomendamos usar 0.0.0.0/0 para SSH acesso, pois você possibilita que todos os endereços IP acessem suas instâncias públicas usando. EC2 SSH Essa abordagem nem mesmo é aceitável por um curto período em um ambiente de teste. Autorize somente um endereço IP específico ou um intervalo de endereços para acessar suas EC2 instâncias usandoSSH.

Etapa 2: criar uma instância de banco de dados InfluxDB

O alicerce básico do Amazon Timestream para InfluxDB é a instância de banco de dados. Esse ambiente é onde você executa seus bancos de dados do InfluxDB.

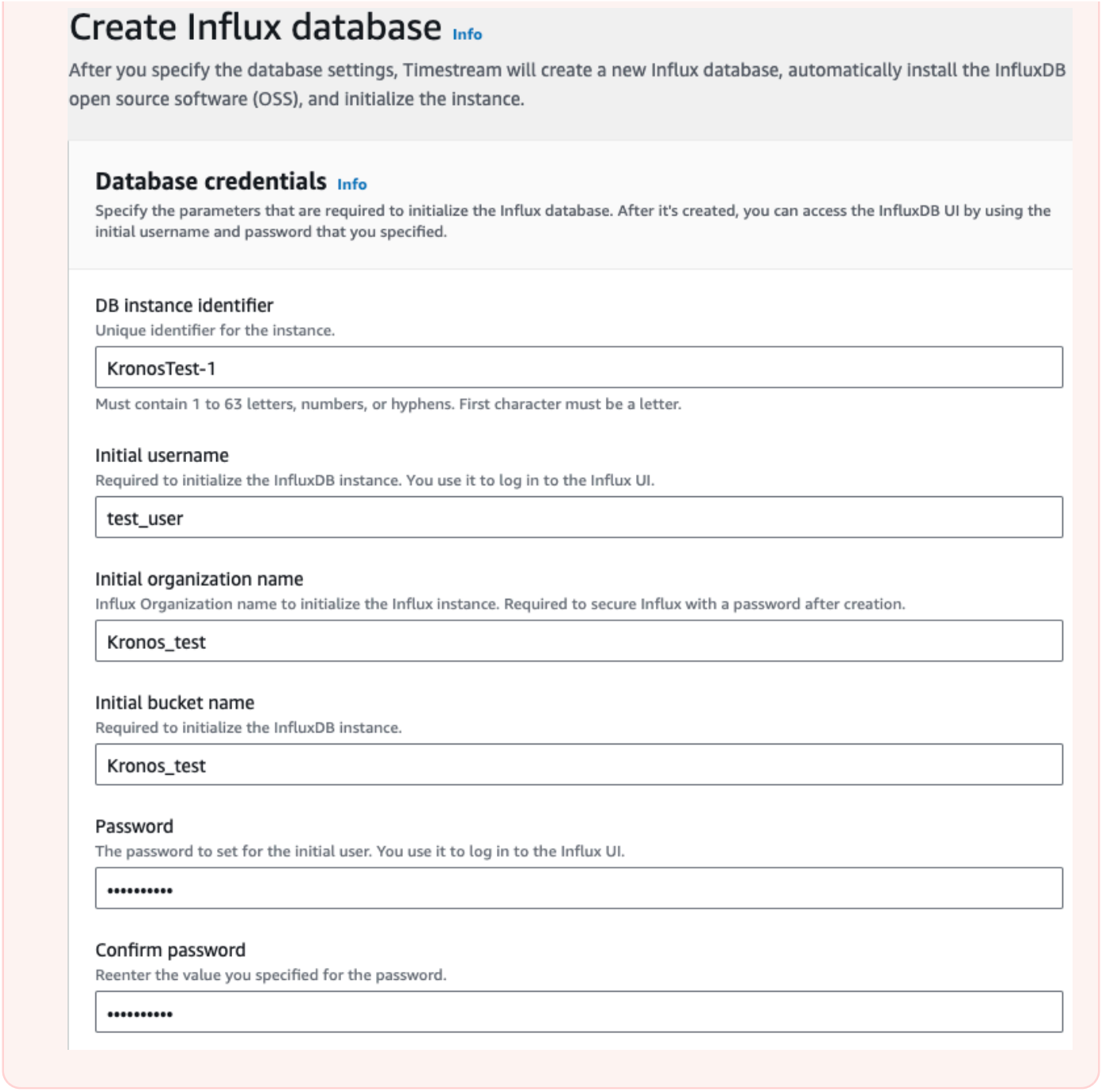
Neste exemplo, você criará uma instância de banco de dados executando o mecanismo de banco de dados InfluxDB com uma classe de instância de banco de dados db.influx.large.

1. Faça login AWS Management Console e abra o console Amazon Timestream for InfluxDB em. https://console.aws.amazon.com/timestream/
2. No canto superior direito do console Amazon Timestream for InfluxDB, escolha a região na qual você deseja criar AWS a instância de banco de dados.
3. No painel de navegação, escolha Bancos de dados InfluxDB.
4. Escolha Criar banco de dados Influx.

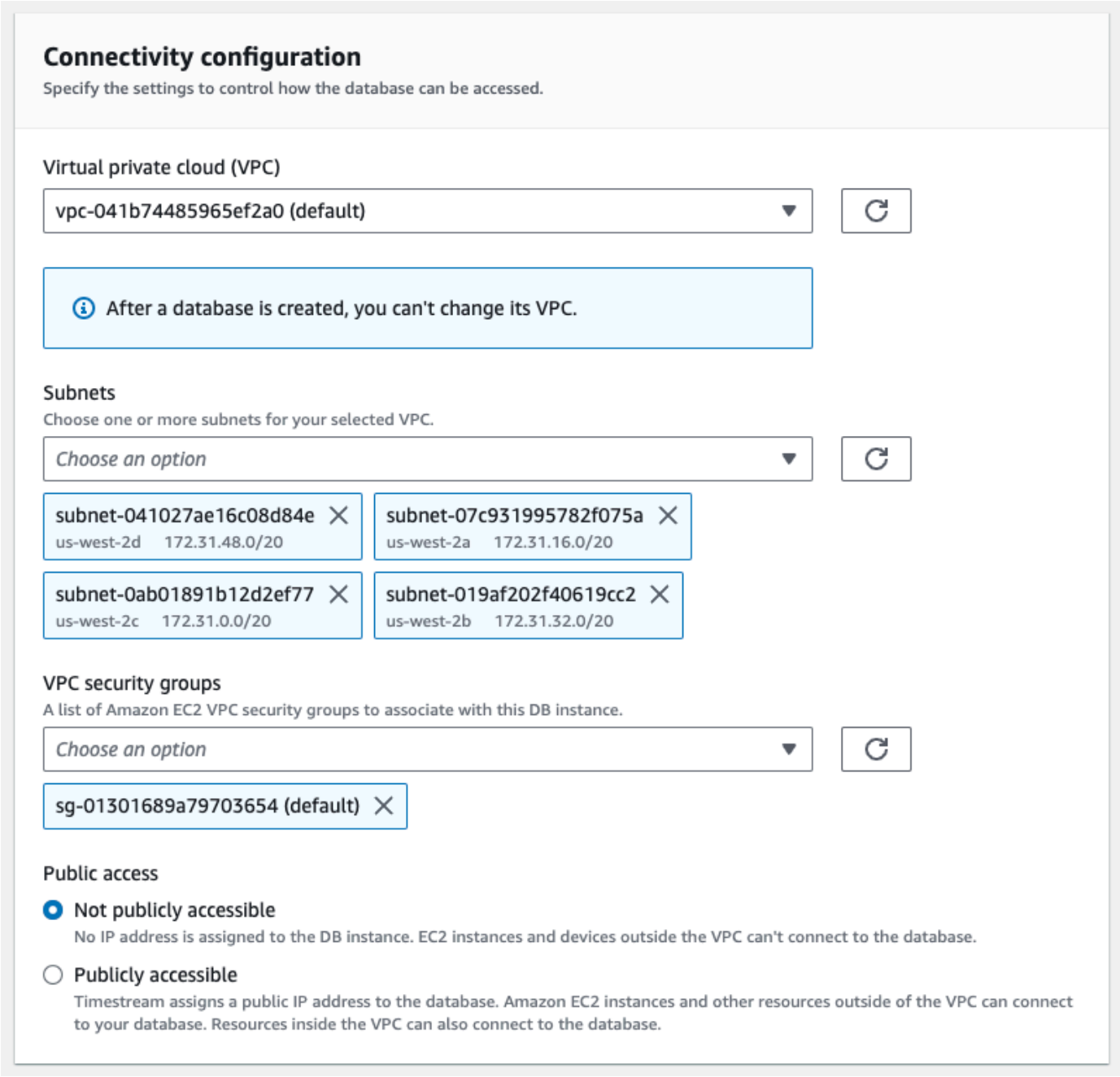


5. Em DB Instance Identifier, insira KronosTest -1.
6. Forneça os parâmetros básicos de configuração do InfluxDB: nome de usuário, organização, nome do bucket e senha.

> **Important**
>
> Você não poderá ver a senha do usuário novamente. Você não conseguirá acessar sua instância e obter um token de operador sem sua senha. Caso você não a registre, talvez seja necessário alterá-la. Consulte Criando um novo token de operador para sua instância do InfluxDB.
>
> Se você precisar alterar a senha do usuário depois que a instância de banco de dados estiver disponível, poderá modificar a instância de banco de dados para fazer isso. Para ter mais informações sobre a modificação de uma instância de banco de dados, consulte Atualizar instâncias de banco de dados.



7. Para Classe de instância de banco de dados, selecione db.influx.large.
8. Para classe de armazenamento de banco de dados, selecione influx IOPS Included 3K.
9. Configure seus registros. Para obter mais informações, consulte Configuração para visualizar os registros do InfluxDB nas instâncias do Timestream Influxdb.
10. Na seção Configuração de conectividade, certifique-se de que sua instância do InfluxDB esteja na mesma sub-rede da instância recém-criada. EC2



11. Escolha Criar banco de dados Influx.
12. Na lista de bancos de dados, escolha o nome da sua nova instância do InfluxDB para mostrar seus detalhes. A instância de banco de dados tem o status de Criação até que esteja pronta para uso.

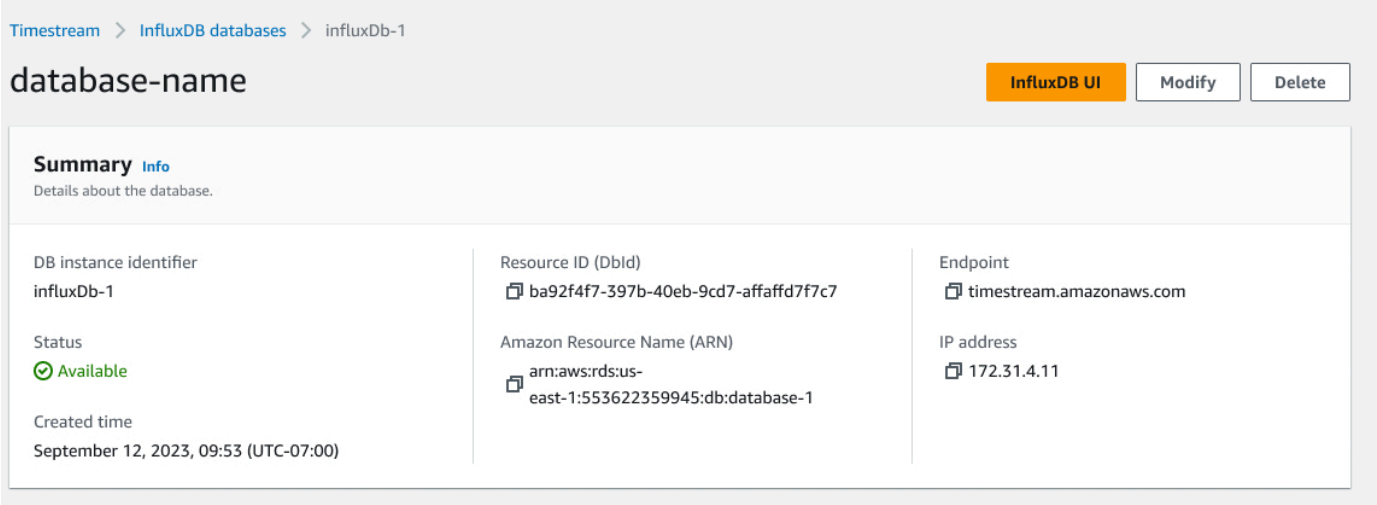
Você pode se conectar à instância de banco de dados quando o status mudar para Disponível. Dependendo da classe da instância de banco de dados e da quantidade de armazenamento, pode levar até 20 minutos para que a nova instância esteja disponível.

> **Important**
>
> No momento, você não pode modificar a configuração de computação (tipos de instância) e armazenamento (tipos de armazenamento) das instâncias existentes.

Etapa 3: envie dados do Telegraf para sua instância do InfluxDB

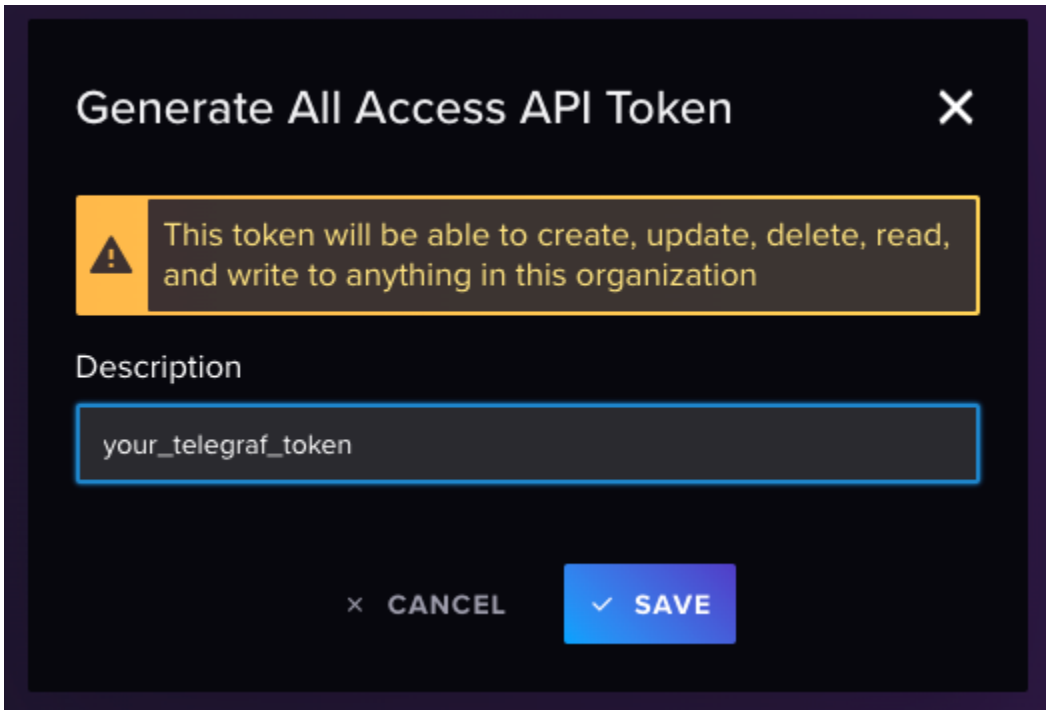
Agora você pode começar a enviar dados de telemetria para sua instância de banco de dados InfluxDB usando o agente Telegraf. Neste exemplo, você instalará e configurará um agente Telegraf para enviar métricas de desempenho para sua instância de banco de dados InfluxDB.

1. Encontre o endpoint (DNSnome) e o número da porta para sua instância de banco de dados.
   a. Faça login no AWS Management Console e abra o console do Amazon Timestream em. https://console.aws.amazon.com/timestream/
   b. No canto superior direito do console do Amazon Timestream, escolha AWS a região para a instância de banco de dados.
   c. No painel de navegação, escolha Bancos de dados InfluxDB.
   d. Escolha o nome da instância de banco de dados InfluxDB para exibir seus detalhes.
   e. Na seção Resumo, copie o endpoint. Além disso, anote o número da porta. Você precisa do endpoint e do número da porta para se conectar à instância de banco de dados (o número da porta padrão para o InfluxDB é 8086).
2. Em seguida, selecione InfluxDB UI.



3. Isso abrirá uma nova janela do navegador, na qual você deverá ver um prompt de login. Insira as credenciais que você usou anteriormente para criar sua instância de banco de dados InfluxDB.
4. No painel de navegação, clique na seta e selecione APITokens.
5. Para esse teste, gere um token de acesso total.

> **Note**
>
> Para cenários de produção, recomendamos criar tokens com acesso específico aos buckets necessários, criados para necessidades específicas do Telegraf.



6. Seu token aparecerá na tela.

> **Important**
>
> Certifique-se de copiar e salvar o Token, pois você não poderá exibi-lo novamente.

7. Conecte-se à EC2 instância que você criou anteriormente seguindo as etapas em Conecte-se à sua instância Linux no Guia do EC2 usuário da Amazon para instâncias Linux.

Recomendamos que você se conecte à sua EC2 instância usando SSH o. Se o utilitário SSH cliente estiver instalado no Windows, Linux ou Mac, você poderá se conectar à instância usando o seguinte formato de comando:

```
ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
```

Por exemplo, suponha que `ec2-database-connect-key-pair.pem` esteja armazenado `/dir1` no Linux e que o público IPv4 DNS da sua EC2 instância sejaec2-12-345-678-90.compute-1.amazonaws.com. Seu SSH comando teria a seguinte aparência:

```
ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
user@ec2-12-345-678-90.compute-1.amazonaws.com
```

8. Obtenha a versão mais recente do telegraf instalada na sua instância. Para fazer isso, use o seguinte comando:

```
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. Configure sua instância do Telegraf.

> **Note**
>
> Se o telegraf.conf não existir ou contiver uma `timestream` seção, você poderá gerar uma com:
>
> ```
> telegraf –section-filter agent:inputs:outputs –input-filter cpu:mem –output-
> filter timestream config > telegraf.conf
> ```

a. Edite o arquivo de configuração normalmente localizado em`/etc/telegraf`.

```
sudo nano /etc/telegraf/telegraf.conf
```

b. Configure entradas básicas para CPU MEM e. DISK

```
[[inputs.cpu]]
  percpu = true
  totalcpu = true
  collect_cpu_time = false
  report_active = false

[[inputs.mem]]

[[inputs.disk]]
  ignore_fs = ["tmpfs", "devtmpfs", "devfs"]
```

c. Configure o plug-in de saída para enviar dados para sua instância de banco de dados InfluxDB e salvar suas alterações.

```
[[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. Configure o destino do Timestream.

```
# Configuration for sending metrics to Amazon Timestream.
[[outputs.timestream]]

  ## Amazon Region and credentials
  region = "us-east-1"
  access_key = "<AWS key here>"
  secret_key = "<AWS secret key here>"
  database_name = "<timestream database name>" # needs to exist

  ## Specifies if the plugin should describe t start.
  describe_database_on_start = false
  mapping_mode = "multi-table" # allows multible tables for each input metrics
```

```
create_table_if_not_exists = true
create_table_magnetic_store_retention_period_in_days = 365
create_table_memory_store_retention_period_in_hours = 24

use_multi_measure_records = true # Important to use multi-measure records
measure_name_for_multi_measure_records = "telegraf_measure"
max_write_go_routines = 25
```

10. Ative e inicie o serviço Telegraf.

```
$ sudo systemctl enable telegraf
$ sudo systemctl start telegraf
```

Etapa 4: excluir a EC2 instância Amazon e a instância de banco de dados InfluxDB

Depois de explorar os dados gerados pelo Telegraf usando sua instância de banco de dados InfluxDB com o InfluxUI, exclua suas instâncias de banco de dados EC2 e suas instâncias de banco de dados InfluxDB para que você não seja mais cobrado por elas.

Para excluir a EC2 instância:

1. Faça login no AWS Management Console e abra o EC2 console da Amazon em https://console.aws.amazon.com/ec2/.
2. No painel de navegação, escolha Instances (Instâncias).
3. Selecione a EC2 instância, escolha Estado da instância e Encerre a instância.
4. Quando a confirmação for solicitada, escolha Terminate (Encerrar).

Para obter mais informações sobre a exclusão de uma EC2 instância, consulte Encerrar sua instância no Guia EC2 do usuário da Amazon.

Para excluir a instância de banco de dados sem o DB snapshot final:

1. Faça login AWS Management Console e abra o console Amazon Timestream for InfluxDB em. https://console.aws.amazon.com/timestream/
2. No painel de navegação, escolha Bancos de dados InfluxDB.
3. Escolha a instância de banco de dados que você deseja excluir.
4. Em Actions, selecione Delete.

5. Conclua a confirmação e escolha Excluir.

(Opcional) Conecte-se à sua instância de banco de dados usando o Amazon Managed Grafana

Você pode usar o Amazon Managed Grafana para criar painéis e monitorar o desempenho de suas EC2 instâncias usando o Amazon Timestream para InfluxDB. O Amazon Managed Grafana é um serviço totalmente gerenciado para o Grafana, uma popular plataforma de análise de código aberto que permite consultar, visualizar e alertar sobre suas métricas, registros e rastreamentos.

Criando um novo token de operador para sua instância do InfluxDB

Se você precisar obter o token de operador para sua nova instância do InfluxDB, execute as seguintes etapas:

1. Para alterar seu token de operador, recomendamos usar o InfluxCLI. Para obter instruções, consulte: Instale e use o influx CLI.
2. Configure seu CLI `--username-password` para usar para poder criar o operador:

```
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. Crie seu novo token de operador. Você será solicitado a fornecer sua senha para confirmar esta etapa.

```
influx auth create --org [YOURORG] --operator
```

> **Important**
>
> Depois que um novo token de operador for criado, você precisará atualizar qualquer cliente que esteja usando o antigo.

Migração de dados do InfluxDB autogerenciado para o Timestream for InfluxDB

O script de migração Influx é um script Python que migra dados entre instâncias do OSS InfluxDB, independentemente de essas instâncias serem gerenciadas ou não. AWS

O InfluxDB é um banco de dados de séries temporais. O InfluxDB contém pontos, que contêm vários pares de valores-chave e um carimbo de data/hora. Quando os pontos são agrupados por pares de valores-chave, eles formam uma série. Uma série é agrupada por um identificador de string chamado medição. O InfluxDB é frequentemente usado para monitoramento de operações, IOT dados e análises. Um bucket é um tipo de contêiner dentro do InfluxDB para armazenar dados. AWS O InfluxDB gerenciado é o InfluxDB dentro do ecossistema. AWS O InfluxDB fornece o InfluxDB v2 API para acessar dados e fazer alterações no banco de dados. O InfluxDB v2 API é o que o script de migração do Influx usa para migrar dados.

- O script de migração do Influx pode migrar buckets e seus metadados, migrar todos os buckets de todas as organizações ou fazer uma migração completa, que substitui todos os dados na instância de destino.
- O script faz backup dos dados da instância de origem localmente, em qualquer sistema que execute o script e, em seguida, restaura os dados na instância de destino. Os dados são mantidos nos diretórios code>influxdb-backup-<timestamp></timestamp>, um para cada migração.
- O script fornece várias opções e configurações, incluindo a montagem de buckets S3 para limitar o uso do armazenamento local durante a migração e a escolha de quais organizações usar durante a migração.

Tópicos

- Preparação
- Como usar o script
- Visão geral da migração

Preparação

A migração de dados para o InfluxDB é realizada com um script Python que utiliza os recursos do InfluxDB e o CLI InfluxDB v2. API A execução do script de migração requer a seguinte configuração de ambiente:

- Versões suportadas: Uma versão mínima de 2.3 do InfluxDB e do Influx CLI é suportada.
- Variáveis de ambiente de token
  - Crie a variável de ambiente `INFLUX_SRC_TOKEN` contendo o token para sua instância de origem do InfluxDB.
  - Crie a variável de ambiente `INFLUX_DEST_TOKEN` contendo o token para sua instância de destino do InfluxDB.
- Python 3
  - Verifique a instalação:`python3 --version`.
  - Se não estiver instalado, instale a partir do site do Python. É necessária a versão mínima 3.7. No Windows, o alias padrão do Python 3 é simplesmente python.
  - As solicitações do módulo Python são obrigatórias. Instale-o com: `shell python3 -m pip install requests`
  - TTheO módulo Python influxdb_client é necessário. Instale-o com: `shell python3 -m pip install influxdb_client`
- InfluxDB CLI
  - Confirme a instalação:`influx version`.
  - Se não estiver instalado, siga o guia de instalação na documentação do InfluxDB.

    Adicione influxo ao seu $PATH.
- Ferramentas de montagem S3 (opcionais)

  Quando a montagem do S3 é usada, todos os arquivos de backup são armazenados em um bucket do S3 definido pelo usuário. A montagem em S3 pode ser útil para economizar espaço na máquina executora ou quando os arquivos de backup precisam ser compartilhados. Se a montagem do S3 não for usada, ao omitir a `--s3-bucket` opção, um `influxdb-backup-<millisecond timestamp>` diretório local será criado para armazenar os arquivos de backup no mesmo diretório em que o script foi executado.

  Para Linux: mountpoint-s3.

  Para Windows: rclone (é necessária uma configuração anterior do rclone).
- Espaço em disco
  - O processo de migração cria automaticamente diretórios exclusivos para armazenar conjuntos de arquivos de backup e retém esses diretórios de backup no S3 ou no sistema de arquivos local, dependendo dos argumentos do programa fornecidos.

  - Certifique-se de que haja espaço em disco suficiente para backup do banco de dados, idealmente dobre o tamanho do banco de dados InfluxDB existente se você optar por omitir a `--s3-bucket` opção e usar o armazenamento local para backup e restauração.
  - Verifique o espaço com `df -h (UNIX/Linux)` ou verificando as propriedades da unidade no Windows.
- Conexão direta

  Verifique se existe uma conexão de rede direta entre o sistema que executa o script de migração e os sistemas de origem e destino. `influx ping --host <host>`é uma forma de verificar uma conexão direta.

Como usar o script

Um exemplo simples de execução do script é o comando:

```
python3 influx_migration.py --src-host <source host> --src-bucket <source bucket> --
dest-host <destination host>
```

Que migra um único bucket.

Todas as opções podem ser visualizadas executando:

```
python3 influx_migration.py -h
```

Uso

```
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]
```

Opções

- -confirm-full (opcional): o uso `--full` sem `--csv` substituirá todos os tokens, usuários, compartimentos, painéis e quaisquer outros dados de valor-chave no banco de dados de destino pelos tokens, usuários, compartimentos, painéis e quaisquer outros dados de valor-chave no banco de dados de origem. `--full`with migra `--csv` apenas todos os metadados do bucket e do bucket, incluindo organizações do bucket. Essa opção (`--confirm-full`) confirmará a migração

completa e prosseguirá sem a intervenção do usuário. Se essa opção não for fornecida e `--full` tiver sido fornecida e `--csv` não fornecida, o script será pausado para execução e aguardará a confirmação do usuário. Esta é uma ação crítica, proceda com cuidado. O padrão é falso.

- -csv (opcional): se os arquivos csv devem ser usados para backup e restauração. Se também `--full` for aprovado, todos os buckets definidos pelo usuário em todas as organizações serão migrados, não os buckets, usuários, tokens ou painéis do sistema. Se uma organização singular for desejada para todos os buckets no servidor de destino em vez de suas organizações de origem já existentes, use. `--dest-org`
- -dest-bucket DEST _ BUCKET (opcional): O nome do bucket do InfluxDB no servidor de destino não deve ser um bucket já existente. O padrão é o valor de `--src-bucket` ou, `None` se `--src-bucket` não for fornecido.
- -dest-host DEST _ HOST: O host do servidor de destino. Exemplo: http://localhost:8086.
- -dest-org DEST _ ORG (opcional): o nome da organização para a qual restaurar os buckets no servidor de destino. Se isso for omitido, todos os buckets migrados do servidor de origem manterão sua organização original e os buckets migrados podem não ficar visíveis no servidor de destino sem a criação e troca de organizações. Esse valor será usado em todas as formas de restauração, seja em um único bucket, em uma migração completa ou em qualquer migração usando arquivos csv para backup e restauração.
- -dir-name DIR _ NAME (opcional): o nome do diretório de backup a ser criado. Padronizado como `influxdb-backup-<timestamp>`. Já não deve existir.
- -full (opcional): se deve realizar uma restauração completa, substituindo todos os dados no servidor de destino por todos os dados do servidor de origem de todas as organizações, incluindo todos os dados de valor-chave, como tokens, painéis, usuários etc. Substituições e. `--src-bucket --dest-bucket` Se usado com`--csv`, migra somente dados e metadados de buckets. O padrão é falso.
- h, --help: mostra a mensagem de ajuda e sai.
- -log-level LOG _ LEVEL (opcional): o nível de log a ser usado durante a execução. As opções são depuração, erro e informações. O padrão é info.
- - retry-restore-dir RETRY _ RESTORE _ DIR (opcional): O diretório a ser usado para restauração quando uma restauração anterior falhar, ignorará o backup e a criação do diretório, falhará se o diretório não existir, pode ser um diretório dentro de um bucket do S3. Se uma restauração falhar, o caminho do diretório de backup que pode ser usado para restauração será indicado em relação ao diretório atual. Os buckets S3 estarão no formulário. `influxdb-backups/<s3 bucket>/<backup directory>` O nome do diretório de backup padrão é`influxdb-backup-<timestamp>`.

- -s3-bucket S3_ BUCKET (opcional): o nome do bucket S3 a ser usado para armazenar arquivos de backup. No Linux, isso é simplesmente o nome do bucket S3, comomy-bucket, dado AWS_ACCESS_KEY_ID e as variáveis de AWS_SECRET_ACCESS_KEY ambiente foram definidas ou ${HOME}/.aws/credentials existem. No Windows, esse é o nome remoto e do bucket rclone configurado, comomy-remote:my-bucket. Todos os arquivos de backup serão deixados no bucket do S3 após a migração em um influxdb-backups-<timestamp> diretório criado. Um diretório de montagem temporário chamado influx-backups será criado no diretório de onde esse script é executado. Se não for fornecido, todos os arquivos de backup serão armazenados localmente em um influxdb-backups-<timestamp> diretório criado a partir do qual esse script será executado.
- -skip-verify (opcional): ignore a verificação do certificado. TLS
- -src-bucket SRC _ BUCKET (opcional): o nome do bucket do InfluxDB no servidor de origem. Se não for fornecido, --full deve ser fornecido.
- -src-host SRC _ HOST (opcional): O host do servidor de origem. O padrão é http://localhost:8086.

Conforme observado anteriormente, mountpoint-s3 rclone são necessários para serem usados, mas podem ser ignorados se o usuário não fornecer um valor para. Nesse caso--s3-bucket, os arquivos de backup serão armazenados localmente em um diretório exclusivo. --s3-bucket

Visão geral da migração

Depois de atender aos pré-requisitos:

1. Execute o script de migração: usando um aplicativo de terminal de sua escolha, execute o script Python para transferir dados da instância do InfluxDB de origem para a instância do InfluxDB de destino.
2. Forneça credenciais: forneça endereços de host e portas como CLI opções.
3. Verifique os dados: verifique se os dados foram transferidos corretamente por:

    a. Usando a interface do usuário do InfluxDB e inspecionando buckets.
    b. Listando compartimentos cominflux bucket list -t <destination token> --host <destination host address> --skip-verify.
    c. Usando influx v1 shell -t <destination token> --host <destination host address> --skip-verify e executando 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> para verificar se o número correto de registros foi migrado.

Example Exemplo de execução

1. Abra um aplicativo de terminal de sua escolha e verifique se os pré-requisitos necessários estão instalados corretamente:

```
~ > 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. Navegue até o script de migração:

```
~ > 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. Prepare as seguintes informações:

   a. Nome do bucket de origem a ser migrado.

   b. (Opcional) Escolha um novo nome de bucket para o bucket migrado no servidor de destino.

   c. Token raiz para instâncias de influxo de origem e destino.

   d. Endereço do host das instâncias de fluxo de origem e destino.

   e. (Opcional) Nome e credenciais do bucket S3; as AWS Command Line Interface credenciais devem ser definidas nas variáveis de ambiente do sistema operacional.

```
# AWS credentials (for timestream testing)
    export AWS_ACCESS_KEY_ID="xxx"
    export AWS_SECRET_ACCESS_KEY="xxx"
```

   f. Construa o comando como:

```
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. Execute o script:

```
~/sample-code/influxdb-sample/migration/influxdb > python3 influx_migration.py --src-bucket primary-bucket --src-host $INFLUXDB_1_HOST --dest-host $KRO
NOS_HOST --dest-bucket new-bucket-name
```

h. Aguarde a conclusão da execução do script.

i. Verifique a integridade dos dados no bucket recém-migrado. `performance.txt` Esse arquivo, localizado no mesmo diretório em que o script foi executado, contém algumas informações básicas sobre a duração de cada etapa.

Cenários de migração

Example Exemplo 1: migração simples usando armazenamento local

Você deseja migrar um único bucket, bucket primário, do servidor de origem (`http://localhost:8086`) para um servidor de destino. (`http://dest-server-address:8086`)

Depois de garantir que você tenha TCP acesso (para HTTP acesso) às duas máquinas que hospedam as instâncias do InfluxDB na porta 8086 e tenha os tokens de origem e de destino e os tenha armazenado como variáveis de ambiente `INFLUX_SRC_TOKEN` e`INFLUX_DEST_TOKEN`, respectivamente, para maior segurança:

```
python3 influx_migration.py --src-bucket primary-bucket --src-host http://
localhost:8086 --dest-host http://dest-server-address:8086
```

A saída deve ser semelhante à seguinte:

```
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
```

O diretório `influxdb-backup-<timestamp>` será criado e armazenado no diretório de onde o script foi executado, contendo arquivos de backup.

Example Exemplo 2: Migração completa usando armazenamento local e registro de depuração

O mesmo que acima, exceto que você deseja migrar todos os buckets, tokens, usuários e painéis, excluindo os buckets no servidor de destino e prosseguindo sem a confirmação do usuário de uma migração completa do banco de dados usando a opção. `--confirm-full` Você também quer ver quais são as medidas de desempenho para ativar o registro de depuração.

```
python3 influx_migration.py --full --confirm-full --src-host http://localhost:8086 --
dest-host http://dest-server-address:8086 --log-level debug
```

A saída deve ser semelhante à seguinte:

```
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 Exemplo 3: Migração completa usandoCSV, organização de destino e bucket do S3

Igual ao exemplo anterior, mas usando Linux ou Mac e armazenando os arquivos no bucket do S3,`my-s3-bucket`. Isso evita que os arquivos de backup sobrecarreguem a capacidade de armazenamento local.

```
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
```

A saída deve ser semelhante à seguinte:

```
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
```

Configurar uma instância de banco de dados

Esta seção mostra como configurar sua instância de banco de dados Amazon Timestream para InfluxDB. Antes de criar uma instância de banco de dados, decida sobre a classe de instância de banco de dados que executará a instância de banco de dados. Além disso, decida onde a instância de banco de dados será executada escolhendo uma AWS região. Depois, crie a instância de banco de dados.

Você pode configurar uma instância de banco de dados com um grupo de parâmetros de banco de dados. Um grupo de parâmetros de banco de dados atua como um contêiner para valores de configuração do mecanismo que são aplicados a uma ou mais instâncias de banco de dados.

Os parâmetros disponíveis dependem do mecanismo de banco de dados e da versão do mecanismo de banco de dados. Você pode especificar um grupo de parâmetros de banco de dados ao criar uma instância de banco de dados. Você também pode modificar uma instância de banco de dados para especificá-los.

> **Important**
>
> No momento, você não pode modificar a configuração de computação (tipos de instância) e armazenamento (tipos de armazenamento) das instâncias existentes.

Criar uma instância de banco de dados

Usar o console

1. Faça login AWS Management Console e abra o Amazon Timestream para o InfluxDB.
2. No canto superior direito do console Amazon Timestream for InfluxDB, escolha a região na qual você deseja criar AWS a instância de banco de dados.
3. No painel de navegação, escolha Bancos de dados InfluxDB.
4. Escolha Criar banco de dados Influx.
5. Em DB Instance Identifier, insira um nome que identificará sua instância.
6. Forneça os parâmetros básicos de configuração do InfluxDB Nome de usuário, organização, nome do bucket e senha.

   > **Important**
   >
   > Seu nome de usuário, organização, nome do bucket e senha serão armazenados como um segredo no AWS Secrets Manager, que será criado para sua conta.

   Se você precisar alterar a senha do usuário depois que a instância de banco de dados estiver disponível, poderá modificá-la usando o Influx CLI.
7.
8. Para Classe de instância de banco de dados, selecione um tamanho de instância que melhor atenda às suas necessidades de carga de trabalho.

9. Para a classe de armazenamento de banco de dados, selecione uma classe de armazenamento que atenda às suas necessidades. Em todos os casos, você só precisará configurar o armazenamento alocado.
10. Na seção Configuração de conectividade, certifique-se de que sua instância do InfluxDB esteja na mesma sub-rede dos novos clientes que precisam de conectividade com sua instância de banco de dados Timestream for InfluxDB. Você também pode optar por disponibilizar publicamente sua instância de banco de dados.
11. Escolha Criar banco de dados Influx.
12. Na lista Bancos de dados, escolha o nome da sua nova instância do InfluxDB para mostrar seus detalhes. A instância de banco de dados tem o status de Criação até que esteja pronta para uso.
13. Quando o status muda para Available (Disponível), você pode se conectar à instância de banco de dados. Dependendo da classe da instância de banco de dados e da quantidade de armazenamento, pode levar até 20 minutos para que a nova instância esteja disponível.

Usando o CLI

Para criar uma instância de banco de dados usando o AWS Command Line Interface, chame o `create-db-instance` comando com os seguintes parâmetros:

```
--name
--vpc-subnet-ids
--vpc-security-group-ids
--db-instance-type
--db-storage-type
--username
--organization
--password
--allocated-storage
```

Para obter informações sobre cada configuração, consulte Configurações para instâncias de banco de dados.

Example Exemplo: usando configurações de mecanismo padrão

Para Linux, macOS ou 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
```

Para 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
```

Usando o API

Para criar uma instância de banco de dados usando o AWS Command Line Interface, chame o `CreateDBInstance` comando com os seguintes parâmetros:

Para obter informações sobre cada configuração, consulte Configurações para instâncias de banco de dados.

> **Important**
>
> Parte do objeto de DBInstance resposta que você recebe um influxAuthParametersSecretArn. Isso manterá um ARN SecretsManager segredo em sua conta. Ele só será preenchido depois que suas instâncias de banco de dados InfluxDB estiverem disponíveis. O segredo contém parâmetros de autenticação de influxo fornecidos durante o `CreateDbInstance` processo. Essa é uma READONLY cópia, pois qualquer outra updates/modifications/deletions desse segredo não afeta a instância de banco de dados criada. Se você excluir esse segredo, nossa API resposta ainda se referirá ao segredo excluídoARN.

Depois de terminar de criar sua instância de banco de dados Timestream para InfluxDB, recomendamos que você baixe, instale e configure o Influx. CLI

O influx CLI fornece uma maneira simples de interagir com o InfluxDB a partir de uma linha de comando. Para obter instruções detalhadas de instalação e configuração, consulte Usar o Influx CLI.

Configurações para instâncias de banco de dados

Você pode criar uma instância de banco de dados usando o console, o `create-db-instance` CLI comando ou a operação `CreateDBInstance` Timestream for API InfluxDB.

A tabela a seguir fornece detalhes sobre as configurações que você escolhe ao criar uma instância de banco de dados.

Configuração do console	Descrição	CLIopção e parâmetro Timestream API
Armazenamento alocado	O valor de armazenamento a ser alocado para a sua instância de banco de dados (em gigabytes). Em alguns casos, alocar uma quantidade de armazenamento para a instância de banco de dados maior do que o tamanho do banco de dados pode melhorar a performance de E/S. Para obter mais informações, consulte Armazenamento de instâncias do InfluxDB.	CLI: `allocated-storage` API: `allocatedstorage`
Nome do bucket	Um nome para o bucket para inicializar a instância InfluxDb	CLI: `bucket` API: `bucket`
Tipo de instância do banco de dados	A configuração da sua instância de banco de dados. Por exemplo, uma classe de instância de banco de dados db.influx.large tem 16 GiB de memória, 2, otimizada para memória. vCPUs Se possível, escolha um tipo de instância de banco de dados grande o suficiente para que um conjunto de trabalho de consulta típico possa ser mantido na memória. Quando os conjuntos de trabalho são mantidos na memória o sistema pode evitar a	CLI: `db-instance-type` API: `Dbinstancetype`

Configuração do console	Descrição	CLIopção e parâmetro Timestream API
	gravação em disco, o que aprimora a performan ce. Para obter mais informações, consulte Tipos de classe de instância de banco de dados.	
DB instance identifier (identifi cador de instância de DB)	O nome da sua instância de banco de dados. Nomeie suas instâncias de banco de dados da mesma forma que nomeia seus servidores no local. Seu identific ador de instância de banco de dados pode conter até 63 caracteres alfanuméricos e deve ser exclusivo para sua conta na AWS região que você escolheu.	CLI: `db-instan ce-identi fier` API: `Dbinstanc eidentifier`
DB parameter group (grupo de parâmetros de banco de dados)	Um parameter group para a sua instância de banco de dados. Você pode escolher um grupo de parâmetros padrão ou criar o seu próprio grupo personalizado de parâmetros. Para obter mais informações, consulteTrabalhar com grupos de parâmetros de banco de dados..	CLI: `db-parame ter-group- name` API: `DBParamet erGroupName`
Configuração de entrega de registros	O nome do bucket S3 onde os registros do InfluxDB serão armazenados.	CLI: `LogDelive ryConfigu ration` API: `log-deliv ery-confi guration`

Configuração do console	Descrição	CLIopção e parâmetro Timestream API
implantação multi-AZ	Create a standby instance (Criar uma instância em espera) para criar uma réplica secundária passiva da instância de banco de dados em outra zona de disponibilidade para oferece suporte a failover. Recomendamos o multi-AZ para workloads de produção a fim de manter a alta disponibilidade. Para desenvolvimento e teste, você pode seleciona r Do not create a standby instance (Não criar uma instância em espera). Para obter mais informações, consulte Configurando e gerenciando uma implantação Multi-AZ.	CLI: `MultiAz` API: `multi-az`
Senha	Esta será sua senha de uso principal para inicializar sua instância de banco de dados do InfluxDB. Você usará essa senha para fazer login no InfluxUI e obter seu token de operador.	CLI: `password` API: `password`
Acesso público	Sim, fornecer à instância de banco de dados um endereço IP público, o que significa que ela pode ser acessada fora doVPC. Para ser acessível publicame nte, a instância de banco de dados também precisa estar em uma sub-rede pública noVPC. Não, para tornar a instância de banco de dados acessível somente de dentro doVPC. Para se conectar a uma instância de banco de dados de fora delaVPC, a instância de banco de dados deve estar acessível ao público. Além disso, o acesso deve ser concedido usando as regras de entrada do grupo de segurança da instância de banco de dados. Além disso, outros requisitos devem ser atendidos.	CLI: `publicly-accessible` API: `PubliclyAccessible`

Configuração do console	Descrição	CLIopção e parâmetro Timestream API
Tipo de armazenamento	O tipo de armazenamento para sua instância de banco de dados Você pode escolher entre três tipos diferentes de armazenamento IOPS incluído com influxo provision ado de acordo com seus requisitos de cargas de trabalho: * Influxo IOPS incluído 3000 IOPS * Influxo IOPS incluído: 12000 IOPS * INflux IOPS Incluído 16000 IOPS Para obter mais informações, consulte Armazenam ento de instâncias do InfluxDB.	CLI: `db-storag e-type` API: `DbStorage Type`
Nome de usuário inicial	Esse será o usuário principal com o qual inicializ ar sua instância de banco de dados InfluxDB. Você usará esse nome de usuário para fazer login no InfluxUI e obter seu token de operador.	CLI: `username` API: `Username`
Sub-redes	Uma sub-rede vpc para associar a essa instância de banco de dados.	CLI: `vpc-subne t-ids` API: `VPCSubnet Ids`
VPCGrupo de segurança (firewall)	O grupo de segurança a ser associado à instância de banco de dados.	CLI: `vpc-secur ity-group-ids` API: `VPCSecuri tyGroupIds`

Conectando-se a uma instância de banco de dados Amazon Timestream para InfluxDB

Antes de se conectar a uma instância de banco de dados, você deve criar a instância de banco de dados. Para obter mais informações, consulte Criar uma instância de banco de dados. Depois que o Amazon Timestream provisionar sua instância de banco de dados, use o API InfluxDB, o CLI Influx ou qualquer cliente ou utilitário compatível com o InfluxDB para se conectar à instância de banco de dados.

Tópicos

- Encontrando as informações de conexão para uma instância de banco de dados Amazon Timestream for InfluxDB
- Opções de autenticação do banco de dados
- Trabalhar com parameter groups

Encontrando as informações de conexão para uma instância de banco de dados Amazon Timestream for InfluxDB

As informações de conexão de uma instância de banco de dados incluem seu endpoint, porta, nome de usuário, senha e um token de acesso válido, como o operador ou todo o token de acesso. Por exemplo, para uma instância de banco de dados InfluxDB, suponha que o valor do endpoint seja. `influxdb1-123456789.us-east-1.timestream-influxdb.amazonaws.com` Nesse caso, o valor da porta é 8086 e o usuário do banco de dados é admin. Com essas informações, você especifica os seguintes valores em uma string de conexão:

- Para host ou nome ou DNS nome do host, especifique`influxdb1-123456789.us-east-1.timestream-influxdb.amazonaws.com`.
- Para porta, especifique 8086.
- Para usuário, especifique admin.
- Para senha, especifique aquela que você forneceu ao criar sua instância de banco de dados.

> **Important**
>
> Quando você criou sua instância Timestream for InfluxDB Db, parte do objeto de DBInstance resposta você recebe uma. `influxAuthParametersSecretArn` Isso manterá um

SecretsManager segredo em sua conta. Ele só será preenchido depois que suas instâncias de banco de dados InfluxDB estiverem disponíveis. O segredo contém parâmetros de autenticação de influxo fornecidos durante o `CreateDbInstance` processo. Essa é uma READONLY cópia, pois qualquer outra updates/modifications/deletions desse segredo não afeta a instância de banco de dados criada. Se você excluir esse segredo, nossa API resposta ainda se referirá ao aviso secreto excluído.

O endpoint é exclusivo para cada instância de banco de dados e os valores da porta e do usuário podem variar. Para se conectar a uma instância de banco de dados, você pode usar o InfluxCLI, o Influx API ou qualquer outro cliente compatível com o InfluxDB.

Para encontrar as informações de conexão de uma instância de banco de dados, use o AWS Management Console. Você também pode usar o AWS comando Command Line Interface (AWS CLI) ou a `describe-db-instances` operação API `GetDBInstance` Timestream-InfluxDB.

Usando o AWS Management Console

1. Faça login AWS Management Console e abra o console do Amazon Timestream.
2. No painel de navegação, escolha Bancos de dados InfluxDB para exibir uma lista de suas instâncias de banco de dados.
3. Escolha o nome da instância de banco de dados para mostrar os detalhes.
4. Na seção Resumo, copie o endpoint. Além disso, anote o número da porta. Você precisará do endpoint e do número da porta para conectar-se à instância de banco de dados.

Se você precisar encontrar as informações de nome de usuário e senha, escolha a guia Detalhes da configuração e escolha a `influxAuthParametersSecretArn` para acessar seu Secrets Manager.

Usando o CLI

- Para encontrar as informações de conexão de uma instância de banco de dados InfluxDB usando o AWS CLI, chame o get-db-instance comando. Na chamada, consulte o ID, o endpoint, a porta e o influxAuthParameters segredo da instância de banco de dados.

  Para Linux, macOS ou Unix:

  ```
  aws timestream-influxdb get-db-instance --identifier id \
   --query "[name,endpoint,influxAuthParametersSecretArn]"
  ```

Para Windows:

```
aws timestream-influxdb get-db-instance --identifier id \
 --query "[name,endpoint,influxAuthParametersSecretArn]"
```

Sua saída deve ser similar à seguinte. Para acessar as informações do nome de usuário, você precisará verificar `InfluxAuthParameterSecret` o.

```
[
    [
        "mydb",
        "mydb-123456789012.us-east-1.timestream-influxdb.amazonaws.com",
        8086,
    ]
]
```

Criação de tokens de acesso

Com essas informações, você poderá se conectar à sua instância para recuperar ou criar seus tokens de acesso. Existem várias maneiras de fazer isso:

Usando o CLI

1. Se ainda não o fez, baixe, instale e configure o influx CLI.
2. Ao configurar sua configuração do Influx, use `--username-password` para CLI autenticar.

```
influx config create --config-name YOUR_CONFIG_NAME  --host-url "https://
yourinstance.timestream-influxdb.amazonaws.com:8086"  --org yourorg --username-
password admin --active
```

3. Use o comando influx auth create para recriar seu token de operador. Leve em conta que esse processo invalidará o antigo token do operador.

```
influx auth create --org kronos --operator
```

4. Depois de ter o token do operador, você pode usar o comando influx auth list para visualizar todos os tokens de todos os seus tokens. Você pode usar o comando influx auth create para criar um token de acesso total.

> **Important**
>
> Você precisará realizar esta etapa para obter primeiro seu token de operador e, em seguida, poder criar novos tokens usando o InfluxDB API ou. CLI

Usando a interface do usuário Influx

1. Navegue até sua instância Timestream for InfluxDB usando o endpoint criado para fazer login e acessar a interface do usuário do InfluxDB. Você precisará usar o nome de usuário e a senha usados para criar sua instância de banco de dados InfluxDB. Você pode recuperar essas informações do `influxAuthParametersSecretArn` que foi especificado no objeto de resposta do`CreateDbInstance`.

   Como alternativa, você pode abrir o InfluxUI a partir do console de gerenciamento Timestream for InfluxDB:

   a. Faça login AWS Management Console e abra o console Amazon Timestream for InfluxDB em. https://console.aws.amazon.com/timestream/

   b. No canto superior direito do console Amazon Timestream for InfluxDB, escolha a região na qual você criou AWS a instância de banco de dados.

   c. Na lista Bancos de dados, escolha o nome da sua instância do InfluxDB para mostrar seus detalhes. No canto superior direito, escolha Open Influx UI.

2. Depois de fazer login em seu InfluxUI, navegue até Carregar dados e depois APITokens usando a barra de navegação esquerda.
3. Escolha + GENERATE API TOKEN e selecione All Access API Token.
4. Insira uma descrição para o API token e escolha SAVE.
5. Copie o token gerado e guarde-o para guardá-lo em segurança.

> **Important**
>
> Ao criar tokens a partir do InfluxUI, os tokens recém-criados serão exibidos apenas uma vez. Certifique-se de copiá-los, caso contrário, você precisará recriá-los.

Usando o InfluxDB API

- Envie uma solicitação para o API `/api/v2/authorizations` endpoint do InfluxDB usando o POST método de solicitação.

  Inclua o seguinte em sua solicitação:

  a. Cabeçalhos:

     i. Autorização: Token < INFLUX _ OPERATOR _ TOKEN >

     ii. Tipo de conteúdo: application/json

  b. Corpo da solicitação: JSON corpo com as seguintes propriedades:

     i. status: “ativo”

     ii. descrição: descrição do API token

     iii. OrgID: ID da organização do InfluxDB

     iv. permissões: matriz de objetos em que cada objeto representa permissões para um tipo de recurso do InfluxDB ou um recurso específico. Cada permissão contém as seguintes propriedades:

        A. ação: “ler” ou “escrever”

        B. recurso: JSON objeto que representa o recurso do InfluxDB para o qual conceder permissão. Cada recurso contém pelo menos a seguinte propriedade: OrgID: ID da organização do InfluxDB

        C. tipo: Tipo de recurso. Para obter informações sobre quais tipos de recursos do InfluxDB existem, use the /api/v2/resources endpoint.

O exemplo a seguir usa `curl` o InfluxDB API para gerar um token de acesso total:

```
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"}}
   ]
```

```
  }
'
```

Opções de autenticação do banco de dados

O Amazon Timestream para InfluxDB oferece suporte às seguintes formas de autenticar usuários do banco de dados:

- Com a autenticação com senha, –sua instância de banco de dados executa toda a administração de contas do usuário. Você cria usuários, especifica senhas e administra tokens usando o InfluxUI, o Influx ou o CLI influx. API
- Autenticação de token — Sua instância de banco de dados executa toda a administração das contas de usuário. Você pode criar usuários, especificar a senha e administrar tokens usando seu token de operador usando o Influx e o InfluxCLI. API

Conexões criptografadas

Você pode usar Secure Socket Layer (SSL) ou Transport Layer Security (TLS) do seu aplicativo para criptografar uma conexão com uma instância de banco de dados. Os certificados necessários para o TLS handshake entre o InfluxDB e os aplicativos criados e gerenciados pelo serviço Kronos. Quando o certificado é renovado, a instância é atualizada automaticamente com a versão mais recente sem exigir nenhuma intervenção do usuário.

Trabalhar com parameter groups

Database parameters (Parâmetros do banco de dados) especifica como o banco de dados é configurado. Por exemplo, os parâmetros do banco de dados podem especificar a quantidade de recursos, como memória, a serem alocados para um banco de dados.

Você gerencia a configuração do seu banco de dados associando suas instâncias de banco de dados a grupos de parâmetros. O Amazon Timestream para InfluxDB define grupos de parâmetros com configurações padrão. Você também pode definir seus próprios grupos de parâmetros com configurações personalizadas.

Visão geral dos grupos de parâmetros

Um grupo de parâmetros de banco de dados atua como um contêiner para valores de configuração de mecanismo que são aplicados a uma ou mais instâncias de bancos de dados.

Tópicos

- Grupos de parâmetros padrão e personalizados
- Criar um grupo de parâmetros de banco de dados
- Parâmetros estáticos e dinâmicos de instância de banco de dados
- Parâmetros e valores de parâmetros compatíveis

Grupos de parâmetros padrão e personalizados

As instâncias de banco de dados usam grupos de parâmetros de banco de dados. As seções a seguir descrevem a configuração e o gerenciamento de grupos de parâmetros de instância de banco de dados.

Criar um grupo de parâmetros de banco de dados

Você pode criar um novo grupo de parâmetros de banco de dados usando o AWS Management Console AWS Command Line Interface, o ou o TimestreamAPI.

As seguintes limitações se aplicam ao nome do grupo de parâmetros de banco de dados:

- O nome deve ter de 1 a 255 letras, números ou hifens.
- Os nomes de grupos de parâmetros padrão podem incluir um ponto, como `default.InfluxDB.2.7`. No entanto, nomes de grupos de parâmetros personalizados não podem incluir um ponto.
- O primeiro caractere deve ser uma letra.
- O nome não pode começar com “dbpg-”
- O nome não pode terminar com hífen nem conter dois hífens consecutivos.
- Se você criar uma instância de banco de dados sem especificar um grupo de parâmetros de banco de dados, a instância de banco de dados usa os padrões do mecanismo InfluxDB.

Não é possível modificar as configurações de parâmetros de um grupo de parâmetros padrão. Em vez disso, você pode fazer o seguinte:

1. Crie um novo grupo de parâmetros.
2. Altere as configurações dos parâmetros desejados. Nem todos os parâmetros de mecanismo de banco de dados em um grupo de parâmetros podem ser modificados.

3. Atualize sua instância de banco de dados para usar o grupo de parâmetros personalizado. Para obter informações sobre como atualizar uma instância de banco de dados, consulteAtualizar instâncias de banco de dados.

> **Note**
>
> Se você modificou sua instância de banco de dados para usar um grupo de parâmetros personalizado e iniciou a instância de banco de dados, o Amazon Timestream for InfluxDB reinicializa automaticamente a instância de banco de dados como parte do processo de inicialização.
> Atualmente, você não poderá modificar grupos de parâmetros personalizados depois que eles forem criados. Se você precisar alterar um parâmetro, é necessário criar um novo grupo de parâmetros personalizado e atribuí-lo às instâncias que exigem essa alteração na configuração. Se você atualizar uma instância de banco de dados existente para atribuir um novo grupo de parâmetros, ela sempre será aplicada imediatamente e reinicializará sua instância.

Parâmetros estáticos e dinâmicos de instância de banco de dados

Os parâmetros da instância de banco de dados InfluxDB são sempre estáticos. Eles se comportam da seguinte maneira:

Quando você altera um parâmetro estático, salva o grupo de parâmetros do banco de dados e o atribui a uma instância, a alteração do parâmetro entra em vigor automaticamente após a reinicialização da instância.

Quando você associa um novo grupo de parâmetros de banco de dados a uma instância de banco de dados, o Timestream aplica os parâmetros estáticos modificados somente após a reinicialização da instância de banco de dados. Atualmente, a única opção é aplicar imediatamente.

Para obter mais informações sobre como alterar o grupo de parâmetros de banco de dados, consulte Atualizar instâncias de banco de dados.

Parâmetros e valores de parâmetros compatíveis

Para determinar os parâmetros compatíveis com sua instância de banco de dados, visualize os parâmetros no grupo de parâmetros de banco de dados usado pela instância de banco de dados.

Para obter mais informações, consulte Visualização de valores de parâmetros para um grupo de parâmetros de banco de dados.

Para obter mais informações sobre todos os parâmetros suportados pela versão de código aberto do InfluxDB, consulte as opções de configuração do InfluxDB. Atualmente, você só poderá modificar os seguintes parâmetros do InfluxDB:

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
flux-log-enabled	Inclui opção para mostrar registros detalhados para consultas do Flux	FALSE	true, false	N/D	
nível de registro	Nível de saída do log. O InfluxDB gera entradas de log com níveis de severidade maiores ou iguais ao nível especificado.	info	depuração, informações, erro	N/D	
sem tarefas	Número de consultas permitidas para execução simultânea. Definir como 0 permite	FALSE	true, false	N/D	

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
	um número ilimitado de consultas simultâneas.				
simultaneidade de consultas	Desative o agendador de tarefas. Se tarefas problemáticas impedirem o início do InfluxDB, use essa opção para iniciar o InfluxDB sem agendar ou executar tarefas.	1024		N/D	

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
query-queue-size	Número máximo de consultas permitidas na fila de execução. Quando o limite da fila é atingido, novas consultas são rejeitadas. Definir como 0 permite um número ilimitado de consultas na fila.	1024		N/D	
tipo de rastreamento	Ative o rastreamento no InfluxDB e especifique o tipo de rastreamento. O rastreamento está desativado por padrão.	""	blog, jaeger	N/D	

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
métricas desativadas	Desative o endpoint HTTP / metrics que expõe as métricas internas do InfluxDB.	FALSE		N/D	

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
http-idle-timeout	Duração máxima em que o servidor deve manter as conexões estabelecidas ativas enquanto aguarda novas solicitações. Defina como `0` sem tempo limite.	30m0s	Duração com unidade `hours`, `nds` . Exemplo: `durationType=minutes,value=10`	Horas: -Mínimo: 0 -Máximo: 256205 Minutos: -Mínimo: 0 -Máximo: 15372286 Segundos: -Mínimo: 0 -Máximo: 922337203 Milissegundos: -Mínimo: 0 -Máximo: 9223372036685	

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
http-read-header-timeout	Duração máxima em que o servidor deve tentar ler HTTP os cabeçalhos de novas solicitações. Defina `0` para não perder tempo.	10s	Duração com unidade`hours`, `nds` . Exemplo: `durationType=minutes,value=10`	Horas: -Mínimo: 0 -Máximo: 256205 Minutos: -Mínimo: 0 -Máximo: 15372286 Segundos: -Mínimo: 0 -Máximo: 922337203 Milissegundos: -Mínimo: 0 -Máximo: 9223372036685	

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
http-read-timeout	Duração máxima em que o servidor deve tentar ler a totalidade das novas solicitações. Defina `0` para não perder tempo.	0	Duração com unidade`hours`, `nds` . Exemplo: `durationType=minutes,value=10`	Horas: -Mínimo: 0 -Máximo: 256205 Minutos: -Mínimo: 0 -Máximo: 15372286 Segundos: -Mínimo: 0 -Máximo: 922337203 Milissegundos: -Mínimo: 0 -Máximo: 922337203685	

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
http-write-timeout	Duração máxima que o servidor deve gastar processando e respondendo às solicitações de gravação. Defina `0` para não perder tempo.	0	Duração com unidade`hours`, `nds` . Exemplo: `durationType=minutes,value=10`	Horas: -Mínimo: 0 -Máximo: 256205 Minutos: -Mínimo: 0 -Máximo: 15372286 Segundos: -Mínimo: 0 -Máximo: 922337203 Milissegundos: -Mínimo: 0 -Máximo: 9223372036685	

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
influxql-max-select-buckets	Número máximo de grupos por intervalos de tempo que uma SELECT declaração pode criar. 0permite um número ilimitado de baldes.	0	Longo	Mínimo: 0 Máximo: 9.223.372 .036.854. 775.807	

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
influxql-max-select-point	Número máximo de pontos que uma SELECT declaração pode processar. 0permite um número ilimitado de pontos. O InfluxDB verifica a contagem de pontos a cada segundo (portanto, as consultas que excedem o máximo não são abortadas imediatamente).	0	Longo	Mínimo: 0 Máximo: 9.223.372.036.854.775.807	

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
influxql-max-select-series	Número máximo de séries que uma `SELECT` declaração pode retornar. `0`permite um número ilimitado de séries.	0	Longo	Mínimo: 0 Máximo: 9.223.372.036.854.775.807	
pprof-desativado	Desative o `/debug/pprof` HTTP endpoint. Esse endpoint fornece dados de criação de perfil de tempo de execução e pode ser útil na depuração.	FALSE	Booleano	N/D	
query-initial-memory-bytes	Bytes iniciais de memória alocados para uma consulta.	0	Longo	Mínimo: 0 Máximo: query-memory-bytes	

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
query-max-memory-bytes	Total máximo de bytes de memória permitidos para consultas.	0	Longo	Mínimo: 0 Máximo: 9.223.372.036.854.775.807	
query-memory-bytes	Especifica o tempo de vida (TTL) em minutos para sessões de usuário recém-criadas.	0	Longo	Mínimo: 0 Máximo: 2.147.483.647	Deve ser maior ou igual query-initial-memory-bytes a.
duração da sessão	Especifica o tempo de vida (TTL) em minutos para sessões de usuário recém-criadas.	60	Inteiro	Mínimo: 0 Máximo: 2880	

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
session-renew-disabled	Desativa a extensão automática da sessão de um usuário TTL em cada solicitação. Por padrão, cada solicitação define o tempo de expiração da sessão para cinco minutos a partir de agora. Quando desativadas, as sessões expiram após a duração especificada da sessão e o usuário é redirecionado para a página de login, mesmo que esteja ativa recentemente.	FALSE	Booleano	N/D	

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
storage-cache-max-memory-tamanho	Tamanho máximo (em bytes) que o cache do fragmento pode atingir antes de começar a rejeitar gravações.	1073741824	Longo	Mínimo: 0 Máximo: 549755813888	Deve ser menor que a capacidade total de memória da instância. Recomendamos configurá-lo para menos de 15% da capacidade total de memória.
storage-cache-snapshot-memory-tamanho	Tamanho (em bytes) no qual o mecanismo de armazenamento capturará o cache e o gravará em um TSM arquivo para disponibilizar mais memória.	26214400	Longo	Mínimo: 0 Máximo: 549755813888	Deve ser menor que storage-cache-max-memory - size.

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
storage-cache-snapshot-write-duração do frio	Duração na qual o mecanismo de armazenamento fará uma captura instantânea do cache e o gravará em um novo TSM arquivo se o fragmento não tiver recebido gravações ou exclusões.	100m0s	Duração com unidadehours, nds . Exemplo: `durationType=minutes,value=10`	Horas: -Mínimo: 0 -Máximo: 256205 Minutos: -Mínimo: 0 -Máximo: 15372286 Segundos: -Mínimo: 0 -Máximo: 922337203 Milissegundos: -Mínimo: 0 -Máximo: 922337203685	

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
storage-compact-full-write-duração do frio	Duração na qual o mecanismo de armazenamento compactará todos os TSM arquivos em um fragmento se não tiver recebido gravações ou exclusões.	40h00m	Duração com unidadehours, nds . Exemplo: `durationType=minutes,value=10`	Horas: -Mínimo: 0 -Máximo: 256205 Minutos: -Mínimo: 0 -Máximo: 15372286 Segundos: -Mínimo: 0 -Máximo: 922337203 Milissegundos: -Mínimo: 0 -Máximo: 922337203685	

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
storage-compact-throughput-burst	Limite de taxa (em bytes por segundo) que TSM as compactaç ões podem gravar no disco.	50331648	Longo	Mínimo: 0 Máximo: 9.223.372 .036.854. 775.807	

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
storage-max-concurrent-compactions	Número máximo de compactações completas e niveladas que podem ser executadas simultaneamente. Um valor de `0` resulta em 50% do `runtime.GOMAXPROCS(0)` usado em tempo de execução. Qualquer número maior que zero limita as compactações a esse valor. Essa configuração não se aplica à captura instantânea de cache.	0	Inteiro	Mínimo: 0 Máximo: 64	

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
storage-max-index-log-tamanho do arquivo	Tamanho (em bytes) no qual um arquivo log (WAL) de gravação antecipada de índice será compactado em um arquivo de índice. Tamanhos menores farão com que os arquivos de log sejam compactados mais rapidamente e resultarão em menor uso do heap em detrimento da taxa de transferência de gravação.	1048576	Longo	Mínimo: 0 Máximo: 9.223.372.036.854.775.807	

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
storage-no-validate-field-tamanho	Ignore a validação do tamanho do campo nas solicitações de gravação recebidas.	FALSE	Booleano	N/D	
storage-retention-check-interval	Intervalo de verificação da aplicação da política de retenção.	30m0s	Duração com unidadehours, nds . Exemplo: `durationType=minutes,value=10`	N/D	Horas: -Mínimo: 0 -Máximo: 256205 Minutos: -Mínimo: 0 -Máximo: 15372286 Segundos: -Mínimo: 0 -Máximo: 922337203 Milissegundos: -Mínimo: 0 -Máximo: 922337203685

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
storage-series-file-max-concurrent-snapshot-compactions	Número máximo de compactações de instantâneos que podem ser executadas simultaneamente em todas as partições de série em um banco de dados.	0	Inteiro	Mínimo: 0 Máximo: 64	

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
storage-series-id-set-tamanho do cache	Tamanho do cache interno usado no TSI índice para armazenar resultados de séries calculados anteriormente. Os resultados em cache são retornados rapidamente, em vez de precisarem ser recalculados quando uma consulta subsequente com o mesmo predicado de chave/valor de tag é executada. Definir esse valor como `0` desativará o cache e poderá diminuir o	100	Longo	Mínimo: 0 Máximo: 9.223.372.036.854.775.807	

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
	desempenho da consulta.				
storage-wal-max-concurrent-escreve	Número máximo de gravações no WAL diretório para tentar ao mesmo tempo.	0	Inteiro	Mínimo: 0 Máximo: 256	

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
storage-wal-max-write-atraso	O tempo máximo que uma solicitação de gravação no WAL diretório aguardará quando o número máximo de gravações ativas simultâneas no WAL diretório for atingido. Defina como `0` para desativar o tempo limite.	10 m	Duração com unidadehours, nds . Exemplo: `durationType=minutes,value=10`	Horas: -Mínimo: 0 -Máximo: 256205 Minutos: -Mínimo: 0 -Máximo: 15372286 Segundos: -Mínimo: 0 -Máximo: 922337203 Milissegundos: -Mínimo: 0 -Máximo: 922337203685	

Parâmetro	Descrição	Valor padrão	Valor	Intervalo válido	Observação
interface do usuário desativada	Desative a interface de usuário (UI) do InfluxDB. A interface do usuário está ativada por padrão.	FALSE	Booleano	N/D	

Definir incorretamente os parâmetros em um grupo de parâmetros pode causar efeitos adversos não intencionais, inclusive diminuição da performance e instabilidade no sistema. Sempre tenha cuidado ao modificar os parâmetros do banco de dados. Experimente as alterações na configuração do grupo de parâmetros em uma instância de banco de dados de teste antes de aplicar essas alterações em uma instância de banco de dados de produção.

Trabalhar com grupos de parâmetros de banco de dados

As instâncias de banco de dados usam grupos de parâmetros de banco de dados. As seções a seguir descrevem a configuração e o gerenciamento de grupos de parâmetros de instância de banco de dados.

Tópicos

- Criar um grupo de parâmetros de banco de dados
- Associando um grupo de parâmetros de banco de dados a uma instância de banco de dados
- Listar grupos de parâmetros de banco de dados
- Visualizar valores de parâmetros para um grupo de parâmetros de banco de dados

Criar um grupo de parâmetros de banco de dados

Usando o AWS Management Console

1. Faça login AWS Management Console e abra o console Amazon Timestream for InfluxDB.
2. No painel de navegação, escolha Parameter groups (Grupos de parâmetros).

3. Escolha Create parameter group (Criar grupo de parâmetros).
4. Insira o nome do novo grupo de parâmetros de banco de dados na caixa Group name (Nome do grupo).
5. Insira uma descrição para o novo grupo de parâmetros de banco de dados na caixa Description (Descrição).
6. Escolha os parâmetros para modificar e aplicar os valores desejados. Para obter mais informações sobre os parâmetros suportados, consulte Parâmetros e valores de parâmetros compatíveis.
7. Escolha Salvar.

Usando o AWS Command Line Interface

- Para criar um grupo de parâmetros de banco de dados usando o AWS CLI, chame o `create-db-parameter-group` comando com os seguintes parâmetros:

```
--db-parameter-group-name <value>
--description <value>
--endpoint_url <value>
--region <value>
--parameters (list) (string)
```

Example Exemplo

Para obter informações sobre cada configuração, consulte Configurações para instâncias de banco de dados. Este exemplo usa configurações de mecanismo padrão.

```
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
```

Associando um grupo de parâmetros de banco de dados a uma instância de banco de dados

Você pode criar seus próprios grupos de parâmetros de banco de dados com configurações personalizadas. Você pode associar um grupo de parâmetros de banco de dados a uma instância

de banco de dados usando o AWS Management Console, o ou o AWS Command Line Interface API Timestream-InfluxDB. Você pode fazer isso ao criar ou modificar uma instância de banco de dados.

Para obter mais informações sobre como criar um parameter group de banco de dados, consulte Criar um grupo de parâmetros de banco de dados. Para obter informações sobre como criar uma instância de banco de dados, consulte Criar uma instância de banco de dados. Para obter informações sobre como modificar uma instância de banco de dados, consulte.. Atualizar instâncias de banco de dados

> **Note**
>
> Quando você associa um novo grupo de parâmetros de banco de dados a uma instância de banco de dados, os parâmetros estáticos modificados são aplicados somente após a reinicialização da instância de banco de dados. Atualmente, somente a inscrição imediata é suportada. O Timestream para InfluxDB suporta apenas parâmetros estáticos.

Usando o AWS Management Console

1. Faça login AWS Management Console e abra o console Amazon Timestream for InfluxDB.
2. No painel de navegação, escolha Bancos de dados InfluxDB e, em seguida, escolha a instância de banco de dados que você deseja modificar.
3. Selecione Atualizar. A página Atualizar instância de banco de dados é exibida.
4. Altere a configuração do grupo de parâmetros de banco de dados.
5. Escolha Continue (Continuar) e verifique o resumo de modificações.
6. Atualmente, somente a opção Aplicar imediatamente é suportada. Essa opção pode causar uma interrupção em alguns casos, pois reinicializará sua instância de banco de dados.
7. Na página de confirmação, revise suas alterações. Se estiverem corretas, escolha Atualizar instância de banco de dados para salvar suas alterações e aplicá-las. Ou escolha Back (Voltar) para editar as alterações ou Cancel (Cancelar) para cancelar as alterações.

Usando o AWS Command Line Interface

Para Linux, macOS ou 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 }}"
```

Para 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 }}"
```

Listar grupos de parâmetros de banco de dados

Você pode listar os grupos de parâmetros de banco de dados que você criou para sua AWS conta.

Usando o AWS Management Console

1. Faça login AWS Management Console e abra o console Amazon Timestream for InfluxDB.
2. No painel de navegação, selecione Parameter groups.
3. Os grupos de parâmetros de banco de dados aparecem em uma lista.

Usando o AWS Command Line Interface

Para listar todos os grupos de parâmetros de banco de dados de uma AWS conta, use o AWS Command Line Interface `list-db-parameter-groups` comando.

```
aws timestream-influxdb list-db-parameter-groups --region region
```

Para retornar grupos de parâmetros de banco de dados específicos para uma AWS conta, use o AWS Command Line Interface `get-db-parameter-group` comando.

```
aws timestream-influxdb get-db-parameter-group --region region --identifier identifier
```

Visualizar valores de parâmetros para um grupo de parâmetros de banco de dados

Você pode obter uma lista de todos os parâmetros em um grupo de parâmetros de banco de dados e seus valores.

Usando o AWS Management Console

1. Faça login AWS Management Console e abra o console Amazon Timestream for InfluxDB.
2. No painel de navegação, selecione Parameter groups.
3. Os grupos de parâmetros de banco de dados aparecem em uma lista.
4. Escolha o nome do grupo de parâmetros para ver sua lista de parâmetros.

Usando o AWS Command Line Interface

Para visualizar os valores dos parâmetros de um grupo de parâmetros do banco de dados, use o AWS Command Line Interface `get-db-parameters` comando com o seguinte parâmetro obrigatório.

```
--db-parameter-group-name
```

Usando o API

Para visualizar os valores dos parâmetros de um grupo de parâmetros do banco de dados, use o API `GetDBParameters` comando Timestream com o seguinte parâmetro obrigatório.

```
DBParameterGroupName
```

Gerenciar instâncias de banco de dados

Esta seção aborda vários aspectos do gerenciamento do Timestream para a instância do InfluxDB para garantir desempenho, disponibilidade e recursos de monitoramento ideais. Ele fornece orientação sobre como atualizar as configurações de suas instâncias de banco de dados, lidar com implantações Multi-AZ e processos de failover. Também explica como excluir instâncias de banco de dados e configurar a visualização de registros para suas instâncias do InfluxDB.

Tópicos

- Atualizar instâncias de banco de dados
- Manutenção de uma instância de banco de dados

- Excluir uma instância de banco de dados
- Implantações de instâncias de banco de dados multi-AZ
- Configuração para visualizar os registros do InfluxDB nas instâncias do Timestream Influxdb

Atualizar instâncias de banco de dados

Você pode atualizar os seguintes parâmetros de configuração da sua instância Timestream for InfluxDB:

- Classe de instância
- Tipo de implantação
- Grupo de parâmetros
- Configuração de entrega de registros

> **Important**
>
> Recomendamos que você teste todas as alterações em uma instância de teste antes de modificar a instância de produção para entender seu impacto, especialmente ao atualizar as versões do banco de dados. Analise o impacto em seu banco de dados e aplicativos antes de atualizar as configurações. Algumas modificações exigem a reinicialização da instância de banco de dados, resultando em tempo de inatividade.

Usando o AWS Management Console

1. Faça login AWS Management Console e abra o console Amazon Timestream for InfluxDB.
2. No painel de navegação, escolha Bancos de dados InfluxDB e, em seguida, escolha a instância de banco de dados que você deseja modificar.
3. Escolha Modificar.
4. Na página Modificar instância de banco de dados, faça as alterações desejadas.
5. Escolha Continue (Continuar) e verifique o resumo de modificações.
6. Escolha Próximo.
7. Revise suas alterações.
8. Escolha Modificar instância para aplicar suas alterações.

> **Note**
>
> Essas modificações exigem a reinicialização da instância de banco de dados Influx e podem causar uma interrupção em alguns casos.

Usando o AWS Command Line Interface

Para atualizar uma instância de banco de dados usando o AWS Command Line Interface, chame o `update-db-instance` comando. Especifique o identificador da instância de banco de dados e os valores para as configurações que deseja modificar. Para obter mais informações sobre cada opção, consulte Configurações para instâncias de banco de dados.

Example Exemplo

O código a seguir modifica mydbinstance definindo um dbparametergroup diferente. As alterações são aplicadas imediatamente.

Para Linux, macOS ou 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
```

Para 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
```

Manutenção de uma instância de banco de dados

Periodicamente, o Amazon Timestream para InfluxDB realiza manutenção nos recursos do Amazon Timestream para InfluxDB. A manutenção geralmente envolve atualizações dos seguintes recursos em sua instância de banco de dados:

- Hardware subjacente
- Sistema operacional subjacente
- Versão do mecanismo de banco de dados

As atualizações no sistema operacional geralmente ocorrem para problemas de segurança.

Alguns itens de manutenção exigem que o Amazon Timestream for InfluxDB deixe sua instância de banco de dados off-line por um curto período. Entre os itens de manutenção que exigem um recurso esteja offline estão sistema operacional obrigatório ou patches de banco de dados. A aplicação obrigatória de patches é automaticamente programada somente para patches relacionados à segurança e à confiabilidade da instância. Essa correção ocorre com pouca frequência, normalmente uma vez a cada poucos meses. Raramente requer mais do que uma fração de sua janela de manutenção.

- As janelas de manutenção são configuradas para ocorrer todos os dias, entre 12h e 4h, horário local, para a região em que sua instância está hospedada.
- Os recursos do cliente podem ser corrigidos uma vez por semana em qualquer uma das sete janelas de manutenção da semana.

Excluir uma instância de banco de dados

A exclusão de uma instância de banco de dados afeta a capacidade de recuperação da instância e a disponibilidade de snapshots. Considere os seguintes problemas:

- Se você quiser excluir todos os recursos do Timestream for InfluxDB, observe que os recursos das instâncias de banco de dados incorrem em cobranças.
- Quando o status de uma instância de banco de dados é excluído, o valor do certificado CA não aparece no console Timestream for InfluxDB ou na saída de comandos ou operações de Timestream. AWS Command Line Interface API

- O tempo necessário para excluir uma instância de banco de dados varia de acordo com a quantidade de dados excluídos e se um snapshot final foi obtido.

Você pode excluir uma instância de banco de dados usando o AWS Management Console AWS Command Line Interface, o ou o TimestreamAPI. Você deve fornecer o nome da instância de banco de dados:

Usando o AWS Management Console

1. Faça login AWS Management Console e abra o console Amazon Timestream for InfluxDB.
2. No painel de navegação, escolha Bancos de dados InfluxDB e, em seguida, escolha a instância de banco de dados que você deseja excluir.
3. Escolha Excluir.
4. Digite confirmar na caixa.
5. Escolha Excluir.

Usando o AWS Command Line Interface

Para encontrar a instância IDs das instâncias de banco de dados em sua conta, chame o `list-db-instances` comando:

```
aws timestream-influxdb list-db-instances \
--endpoint-url YOUR_ENDPOINT \
--region YOUR_REGION
```

Para excluir uma instância de banco de dados usando o AWSCLI, chame o `delete-db-instance` comando com as seguintes opções:

```
aws timestream-influxdb list-db-instances \
--identifier YOUR_DB_INSTANCE \
```

Example Exemplo

Para Linux, macOS ou Unix:

```
aws timestream-influxdb delete-db-instance \
    --identifier mydbinstance
```

Para Windows:

```
aws timestream-influxdb delete-db-instance ^
    --identifier mydbinstance
```

Implantações de instâncias de banco de dados multi-AZ

O Amazon Timestream for InfluxDB fornece alta disponibilidade e suporte de failover para instâncias de banco de dados usando implantações Multi-AZ com uma única instância de banco de dados em espera. Esse tipo de implantação é chamado de implantação de instância de banco de dados multi-AZ. O Amazon Timestream para InfluxDB usa a tecnologia de failover da Amazon.

Em uma implantação de instância de banco de dados Multi-AZ, o Amazon Timestream provisiona e mantém automaticamente uma réplica síncrona em espera em uma zona de disponibilidade diferente. A instância de banco de dados primária é sincronicamente replicada nas zonas de disponibilidade para uma réplica em espera a fim de oferecer redundância de dados. A execução de uma instância de banco de dados com alta disponibilidade pode aumentar a disponibilidade durante a falha da instância de banco de dados e a interrupção da zona de disponibilidade. Para ter mais informações sobre zonas de disponibilidade, consulte AWS Regiões e zonas de disponibilidade.

> **Note**
>
> A opção de alta disponibilidade não é uma solução de escalabilidade para cenários somente leitura. Não é possível utilizar uma réplica em espera para servir tráfego de leitura.

Usando o console do Amazon Timestream, você pode criar uma implantação de instância de banco de dados Multi-AZ simplesmente especificando a opção Criar uma instância em espera na seção Configuração de disponibilidade e durabilidade ao criar uma instância de banco de dados. Você também pode especificar a implantação de uma instância de banco de dados Multi-AZ com o AWS Command Line Interface ou o Amazon TimestreamAPI. Use o CLI comando `create-db-instance` ou ou a `CreateDBInstance` API operação.

Instâncias de banco de dados que usam implantações de instância de banco de dados multi-AZ podem ter maior latência de gravação e confirmação em comparação com uma implantação single-AZ. Isso pode acontecer devido à replicação de dados síncrona que ocorre. Você pode ter uma alteração na latência se sua implantação passar para a réplica em espera, embora tenha sido AWS projetada com conectividade de rede de baixa latência entre as zonas de disponibilidade. Para

cargas de trabalho de produção, recomendamos que você use o armazenamento IOPS incluído de 12K ou 16K IOPS para um desempenho rápido e consistente. Para ter mais informações sobre classes de instância de banco de dados, consulte Classes da instância de banco de dados.

Configurando e gerenciando uma implantação Multi-AZ

O Timestream para implantações do InfluxDB Multi-AZ só pode ter um modo de espera. Quando a implantação tem uma instância de banco de dados em espera, ela é chamada de implantação de instância de banco de dados Multi-AZ. Uma implantação de instância de banco de dados Multi-AZ tem uma instância de banco de dados em espera que fornece suporte para failover, mas não serve tráfego de leitura.

> **Important**
>
> Sua instância deve ter pelo menos duas sub-redes associadas a ela para executar atualizações de Single-AZ a Multi-AZ. Depois que a instância é criada, você não pode modificar seu modo de implantação de Single-AZ para Multi-AZ.

Você pode usar o AWS Management Console para determinar se sua instância de banco de dados é uma implantação Single-AZ ou Multi-AZ.

Usando o AWS Management Console

1. Faça login AWS Management Console e abra o console Amazon Timestream for InfluxDB.
2. No painel de navegação, escolha Bancos de dados InfluxDB e, em seguida, escolha DB identifier.

Uma implantação de instância de banco de dados multi-AZ tem as seguintes características:

- Há apenas uma linha para a instância de banco de dados.
- O valor de Role (Função) é Instance (Instância) ou Primary (Principal).
- O valor de multi-AZ é Yes (Sim).

Processo de failover para o Amazon Timestream

Se uma interrupção planejada ou não planejada de sua instância de banco de dados resultar de um defeito na infraestrutura, o Amazon Timestream for InfluxDB mudará automaticamente

para uma réplica em espera em outra zona de disponibilidade se você tiver ativado o Multi-AZ. O tempo de conclusão do failover depende da atividade do banco de dados e de outras condições no momento em que a instância de banco de dados primária se tornou indisponível. Em geral, os tempos de failover variam de 60 a 120 segundos. No entanto, transações grandes ou um processo de recuperação longo podem aumentar o tempo de failover. Quando o failover estiver concluído, pode levar mais tempo para que o console do Timestream reflita a nova zona de disponibilidade.

> **Note**
>
> O Amazon Timestream gerencia os failovers automaticamente para que você possa retomar as operações do banco de dados o mais rápido possível sem intervenção administrativa. A instância de banco de dados principal muda automaticamente para a réplica em espera se alguma das condições descritas na tabela a seguir ocorrer.

Motivo do failover	Descrição
O sistema operacional subjacente à instância do banco de dados Timestream está sendo corrigido em uma operação off-line.	Um failover foi acionado durante a janela de manutenção para um patch de SO ou uma atualização de segurança.
O host principal da instância Multi-AZ do Timestream não está íntegro.	A implantação de instância de banco de dados multi-AZ detectou uma instância de banco de dados primária danificada e executou failover.
O host principal da instância Multi-AZ do Timestream está inacessível devido à perda de conectividade de rede.	O monitoramento do Timestream detectou uma falha na acessibilidade da rede na instância de banco de dados primária e acionou um failover.
A instância Timestream foi modificada pelo cliente.	Uma modificação da instância de banco de dados Timesteam for InfluxDB acionou um failover. Para obter mais informações, consulte Atualizar instâncias de banco de dados.
A instância primária Multi-AZ do Timestream está ocupada e não responde.	A instância de banco de dados principal não responde. Recomendamos que você faça o seguinte: * Examine o evento quanto ao uso excessivo CPU de memória ou espaço

Motivo do failover	Descrição
	de troca. * Avalie sua carga de trabalho para determinar se você está usando a classe de instância de banco de dados apropriada. Para ter mais informações, consulte Classes de instância de banco de dados.
O volume de armazenamento subjacente ao host principal da instância Multi-AZ do Timestream sofreu uma falha.	A implantação de instância de banco de dados multi-AZ detectou um problema de armazenam ento na instância de banco de dados primária e executou o failover.

Configurando as pesquisas de DNS nomes JVM TTL para

O mecanismo de failover altera automaticamente o registro Domain Name System (DNS) da instância de banco de dados para apontar para a instância de banco de dados em espera. Como resultado, você precisará restabelecer todas as conexões existentes para sua instância de banco de dados. Em um ambiente Java virtual machine (JVM), devido ao funcionamento do mecanismo de DNS cache Java, talvez seja necessário redefinir as configuraçõesJVM.

As pesquisas de DNS nomes JVM dos caches. Quando o JVM resolve um nome de host para um endereço IP, ele armazena o endereço IP em cache por um período de tempo especificado, conhecido como time-to-live()TTL.

Como AWS os recursos usam entradas de DNS nome que mudam ocasionalmente, recomendamos que você configure seu JVM com um TTL valor não superior a 60 segundos. Isso garante que, quando o endereço IP de um recurso for alterado, seu aplicativo possa receber e usar o novo endereço IP do recurso solicitando o. DNS

Em algumas configurações Java, o JVM padrão TTL é definido para que ele nunca atualize as DNS entradas até que JVM seja reiniciado. Portanto, se o endereço IP de um AWS recurso mudar enquanto seu aplicativo ainda estiver em execução, ele não poderá usar esse recurso até que você reinicie manualmente JVM e as informações IP em cache sejam atualizadas. Nesse caso, é crucial definir o s para TTL que JVM ele atualize periodicamente as informações de IP em cache.

Você pode obter o JVM padrão TTL recuperando o valor da `networkaddress.cache.ttl` propriedade:

```
String ttl = java.security.Security.getProperty("networkaddress.cache.ttl");
```

> **Note**
>
> O padrão TTL pode variar de acordo com a versão do seu JVM e se um gerenciador de segurança está instalado. Muitos JVMs fornecem um padrão de TTL menos de 60 segundos. Se você estiver usando esse JVM e não usando um gerenciador de segurança, poderá ignorar o restante deste tópico.
> Para modificar o sTTL, defina o valor JVM da propriedade networkaddress.cache.ttl. Use um dos seguintes métodos, dependendo das necessidades:
>
> - Para definir o valor da propriedade globalmente para todos os aplicativos que usam oJVM, defina `networkaddress.cache.ttl` no `$JAVA_HOME/jre/lib/security/java.security` arquivo.
>
>   ```
>   networkaddress.cache.ttl=60
>   ```
>
> - Para definir a propriedade localmente somente para seu aplicativo, defina `networkaddress.cache.ttl` no código de inicialização do aplicativo antes de quaisquer conexões de rede serem estabelecidas.
>
>   ```
>   java.security.Security.setProperty("networkaddress.cache.ttl" , "60");
>   ```

Configuração para visualizar os registros do InfluxDB nas instâncias do Timestream Influxdb

Por padrão, o InfluxDB gera registros que vão para stdout. Para obter mais informações, consulte Gerenciar registros do InfluxDB

Para visualizar os registros do InfluxDB gerados a partir da instância que você criou por meio do Timestream InfluxDB, oferecemos a oportunidade de fornecer registros por hora. Esses registros irão para um bucket S3 específico que você deve criar antes de criar sua instância.

- Antes de criar a instância, o bucket Amazon S3 fornecido também deve dar permissão ao Timestream-InfluxDB para enviar registros para esse bucket, fornecendo uma política de bucket

com o Timestream InfluxDB Service Principal da seguinte forma (substituir *{BUCKET_NAME}* com o nome real do seu bucket Amazon S3:

```
{
    "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/*"
        }
    ]
}
```

- O bucket fornecido deve estar na mesma conta e na mesma região da instância criada do Timestream InfluxDB

  Aqui está o comando que você pode chamar para criar uma instância para receber registros de influxo:

```
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
```

  Aqui está o formato desse parâmetro.

```
-- log-delivery-configuration
{
    "S3Configuration": {
      "BucketName": "string",
      "Enabled": true|false
    }
```

```
}
```

- Esse campo não é obrigatório e o registro não está habilitado por padrão.
- Não definir esse campo é o mesmo que não ter os registros habilitados.
- Os registros serão enviados para o bucket especificado com um prefixo `deInfluxLogs/`.
- Depois de criar a instância, você pode modificar a configuração de entrega de registros com o `update-db-instance` API comando.

O InfluxDB oferece diferentes tipos de registros. Eles podem ser configurados definindo os parâmetros do InfluxDB. Use os parâmetros flux-log-enabled e em nível de log para configurar o tipo de registros emitidos pela instância. Para obter mais informações, consulte Parâmetros e valores de parâmetros compatíveis.

Adicionar tags e rótulos a recursos

Você pode rotular o Amazon Timestream para recursos do InfluxDB usando tags. As tags permitem categorizar recursos de diferentes maneiras, por exemplo, por finalidade, proprietário, ambiente ou outros critérios. As tags podem ajudar a fazer o seguinte:

- Identificar rapidamente um recurso com base nas tags que você atribuiu a ele.
- Veja AWS as contas divididas por etiquetas.

A marcação é suportada por AWS serviços como Amazon Elastic Compute Cloud (AmazonEC2), Amazon Simple Storage Service (Amazon S3), Timestream for InfluxDB e muito mais. Uma marcação eficiente é capaz de fornecer insights de custos, permitindo criar relatórios entre serviços que possuem uma tag específica.

Por fim, é recomendável seguir estratégias de marcação ideais. Para obter informações, consulte Estratégias de marcação da AWS.

Restrições de marcação

Cada tag consiste em uma chave e um valor, ambos definidos por você. As seguintes restrições são aplicáveis:

- Cada instância de banco de dados Timestream for InfluxDB pode ter apenas uma tag com a mesma chave. Se você tentar adicionar uma tag existente, o valor da tag existente será atualizado para o novo valor.
- Um valor atua como um descritor dentro de uma categoria de tag. No Timestream for InfluxDB, o valor não pode ser vazio ou nulo.
- As chaves e valores das tags diferenciam maiúsculas de minúsculas.
- O comprimento máximo da chave é 128 caracteres Unicode.
- O número máximo de tags que você pode atribuir a um recurso é 50.
- Os caracteres permitidos são letras, espaço em branco e números, além dos seguintes caracteres especiais: `+ - = . _ : /`
- O número máximo de tags por recurso é 50.
- AWS- os nomes e valores das tags atribuídos recebem automaticamente o `aws:` prefixo, que você não pode atribuir. AWS-os nomes de tag atribuídos não contam para o limite de 50. Nomes de tags atribuídos pelo usuário têm o prefixo `user:` no relatório de alocação de custos.
- Não é possível colocar uma data retroativa na aplicação de uma tag.

Melhores práticas de segurança para Timestream for InfluxDB

Otimize gravações no InfluxDB

Como qualquer outro banco de dados de séries temporais, o InfluxDB foi criado para poder ingerir e processar dados em tempo real. Para manter o melhor desempenho do sistema, recomendamos as seguintes otimizações ao gravar dados no InfluxDB:

- Gravações em lote: ao gravar dados no InfluxDB, grave dados em lotes para minimizar a sobrecarga da rede relacionada a cada solicitação de gravação. O tamanho ideal do lote é de 5.000 linhas de protocolo de linha por solicitação de gravação. Para escrever várias linhas em uma solicitação, cada linha do protocolo de linha deve ser delimitada por uma nova linha (\n).
- Classifique as tags por chave: antes de gravar pontos de dados no InfluxDB, classifique as tags por chave em ordem lexicográfica.

```
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
```

- Use a maior precisão de tempo possível: — O InfluxDB grava dados com precisão de nanossegundos, no entanto, se seus dados não forem coletados em nanossegundos, não há necessidade de gravar com essa precisão. Para um melhor desempenho, use a maior precisão possível para registros de data e hora. Você pode especificar a precisão de gravação quando:
  - Ao usar o, SDK você pode especificar o WritePrecision ao definir o atributo de hora do seu ponto. Para obter mais informações sobre as bibliotecas de cliente do InfluxDB, consulte a documentação do InfluxDB.
  - Ao usar o Telegraf, você configura a precisão do tempo na configuração do agente Telegraf. A precisão é especificada como um intervalo com uma unidade inteira + (por exemplo, 0s,10ms,2us,4s). As unidades de tempo válidas são “ns”, “us”, “ms” e “s”.

```
[agent]
 interval ="10s"
 metric_batch_size="5000"
 precision = "0s"
```

- Use a compressão gzip: — Use a compressão gzip para acelerar as gravações no InfluxDB e reduzir a largura de banda da rede. Os benchmarks mostraram uma melhoria de velocidade de até 5x quando os dados são compactados.
  - Ao usar o Telegraf, na configuração do plug-in de saída InfluxDB_v2 em seu telegraf.conf, defina a opção content_encoding como gzip:

```
[[outputs.influxdb_v2]]
  urls = ["http://localhost:8086"]
  # ...
  content_encoding = "gzip"
```

  - Ao usar bibliotecas de cliente, cada biblioteca cliente do InfluxDB fornece opções para compactar solicitações de gravação ou impõe a compactação por padrão. O método para ativar a compactação é diferente para cada biblioteca. Para obter instruções específicas, consulte a documentação do InfluxDB
  - Ao usar o API `/api/v2/write` endpoint do InfluxDB para gravar dados, compacte os dados com gzip e defina o cabeçalho Content-Encoding como gzip.

Design para desempenho

Crie seu esquema para consultas mais simples e de maior desempenho. As diretrizes a seguir garantirão que seu esquema seja fácil de consultar e maximize o desempenho da consulta:

- Design para consultar: escolha medidas, chaves de tag e chaves de campo que sejam fáceis de consultar. Para atingir esse objetivo, siga estes princípios:
  - Use medidas que tenham um nome simples e descrevam o esquema com precisão.
  - Evite usar o mesmo nome para uma chave de tag e uma chave de campo dentro do mesmo esquema.
  - Evite usar palavras-chave reservadas do Flux e caracteres especiais nas teclas de tag e campo.
  - As tags armazenam metadados que descrevem os campos e são comuns em muitos pontos de dados.
  - Os campos armazenam dados exclusivos ou altamente variáveis, geralmente pontos de dados numéricos.
  - As medições e chaves não devem conter dados, mas devem ser usadas para agregar ou descrever dados. Os dados serão armazenados em valores de tag e campo.
- Mantenha sua cardinalidade de série temporal sob controle A alta cardinalidade de série é uma das principais causas da diminuição do desempenho de gravação e leitura no InfluxDB. No contexto do InfluxDB, a alta cardinalidade se refere à presença de um número muito grande de valores de tag exclusivos. Os valores das tags são indexados no InfluxDB, o que significa que um número muito alto de valores exclusivos gerará um índice maior, o que pode diminuir a ingestão de dados e o desempenho da consulta.

  Para entender melhor e resolver possíveis problemas relacionados à alta cardinalidade, você pode seguir estas etapas:
  - Entenda as causas da alta cardinalidade
  - Meça a cardinalidade de seus baldes
  - Tome medidas para resolver a alta cardinalidade
- Causas da alta cardinalidade de série O InfluxDB indexa os dados com base em medições e tags para acelerar as leituras de dados. Cada conjunto de elementos de dados indexados forma uma chave de série. Tags contendo informações altamente variáveis, como strings exclusivasIDs, hashes e aleatórias, levam a um grande número de séries, também conhecido como alta cardinalidade de série. A cardinalidade de alta série é o principal fator de alto uso de memória no InfluxDB.

- Medindo a cardinalidade da série Se você tiver lentidão no desempenho ou observar um uso cada vez maior de memória em sua instância do Timestream for InfluxDB, recomendamos medir a cardinalidade da série de seus buckets.

  O InfluxDB fornece funções que permitem medir a cardinalidade da série no Flux e no InfluxQL.

  - No Flux, use a função `influxdb.cardinality()`
  - No FluxQL, use o comando `SHOW SERIES CARDINALITY`

  Em ambos os casos, o mecanismo retornará o número de chaves de série exclusivas em seus dados. Lembre-se de que não é recomendável ter mais de 10 milhões de chaves de série em nenhuma de suas instâncias do Timestream for InfluxDB.

- Causas da alta cardinalidade em série Se você descobrir que algum de seus compartimentos tem alta cardinalidade, há algumas etapas de correção que você pode seguir para corrigi-lo:

  - Analise suas tags: garanta que suas cargas de trabalho não gerem casos em que as tags tenham valores exclusivos para a maioria das entradas. Isso pode acontecer nos casos em que o número de valores de tag exclusivos sempre aumenta com o tempo ou se mensagens do tipo log estão sendo gravadas no banco de dados, onde cada mensagem teria uma combinação exclusiva de timestamp, tags etc. Você pode usar o seguinte código Flux para ajudá-lo a descobrir quais tags estão contribuindo mais para seus problemas de alta cardinalidade:

```
// 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")
```

Se você estiver enfrentando uma cardinalidade muito alta, a consulta acima pode expirar. Se você tiver um tempo limite, execute as consultas abaixo, uma de cada vez.

Gere uma lista de tags:

```
// Generate a list of tagsimport "influxdata/influxdb/schema"

schema.tagKeys(bucket: "example-bucket")
```

Conte valores de tag exclusivos para cada 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()
```

Recomendamos que você os execute em momentos diferentes para identificar qual tag está crescendo mais rápido.

- Melhore seu esquema: siga as recomendações de modelagem discutidas em nossoMelhores práticas de segurança para Timestream for InfluxDB.
- Remova ou agregue dados antigos para reduzir a cardinalidade: considere se seus casos de uso precisam ou não de todos os dados que estão causando seus problemas de alta cardinalidade. Se esses dados não forem mais necessários ou acessados com frequência, você poderá agregá-los, excluí-los ou exportá-los para outro mecanismo, como o Timestream for Live Analytics, para armazenamento e análise de longo prazo.

Solução de problemas

Aviso de que a versão “dev” não é reconhecida

O aviso 'WARN: Não foi possível analisar a versão “dev” relatada pelo servidor, supondo que o último backup/restauração APIs seja suportado' pode ser exibido durante a migração. Esse aviso pode ser ignorado.

A migração falhou durante a fase de restauração

No caso de uma falha na migração durante o estágio de restauração, os usuários podem usar o `--retry-restore-dir` sinalizador para tentar novamente a restauração. Use o `--retry-restore-dir` sinalizador com um caminho para um diretório previamente salvo para pular o estágio de backup e tentar novamente o estágio de restauração. O diretório de backup criado usado para uma migração será indicado se a migração falhar durante a restauração.

Os possíveis motivos para uma falha na restauração incluem:

- Token de destino inválido do InfluxDB — Um bucket existente na instância de destino com o mesmo nome da instância de origem. Para migrações de bucket individuais, use a `--dest-bucket` opção de definir um nome exclusivo para o bucket migrado.
- Falha de conectividade, seja com os hosts de origem ou de destino ou com um bucket S3 opcional.

Diretrizes operacionais básicas do Amazon Timestream para InfluxDB

A seguir estão as diretrizes operacionais básicas que todos devem seguir ao trabalhar com o Amazon Timestream para o InfluxDB. Observe que o Acordo de Nível de Serviço do Amazon Timestream para InfluxDB exige que você siga estas diretrizes:

- Use métricas para monitorar seu uso CPU de memória e armazenamento. Você pode configurar CloudWatch a Amazon para notificá-lo quando os padrões de uso mudarem ou quando você se aproximar da capacidade de sua implantação. Dessa maneira, é possível manter a disponibilidade e a performance do sistema.
- Escale sua instância de banco de dados quando estiver se aproximando dos limites de capacidade de armazenamento. É preciso ter algum buffer de armazenamento e memória para acomodar aumentos imprevistos na demanda de seus aplicativos. Lembre-se de que, neste momento, você precisará criar uma nova instância e migrar seus dados para conseguir isso.
- Se a workload do banco de dados exigir mais E/S do que você provisionou, a recuperação após um failover ou a falha no banco de dados será lenta. Para aumentar a capacidade de E/S de uma instância de banco de dados, realize uma ou todas as ações a seguir:
  - Migre para uma instância de banco de dados diferente com maior capacidade de E/S.
  - Se você já estiver usando o armazenamento de armazenamento IOPS incluído no Influx, provisione um tipo de armazenamento com maior IOPS incluído.

- Se seu aplicativo cliente estiver armazenando em cache os dados do Domain Name Service (DNS) de suas instâncias de banco de dados, defina um valor time-to-live (TTL) de menos de 30 segundos. O endereço IP subjacente de uma instância de banco de dados pode ser alterado após um failover. Armazenar os DNS dados em cache por um longo período pode, portanto, levar a falhas na conexão. Sua aplicação pode tentar se conectar a um endereço IP que não está mais em serviço.

RAMRecomendações de instâncias de banco

Uma prática recomendada de desempenho do Amazon Timestream para InfluxDB é alocar o RAM suficiente para que seu conjunto de trabalho resida quase completamente na memória. O conjunto de trabalho é composto de dados e índices que são usados frequentemente em sua instância. Quanto mais você usar a instância de banco de dados, mais o conjunto de trabalho crescerá.

Segurança no Timestream para InfluxDB

A segurança na nuvem AWS é a maior prioridade. Como AWS cliente, você se beneficia de uma arquitetura de data center e rede criada para atender aos requisitos das organizações mais sensíveis à segurança.

A segurança é uma responsabilidade compartilhada entre você AWS e você. O modelo de responsabilidade compartilhada descreve isto como segurança da nuvem e segurança na nuvem:

- Segurança da nuvem — AWS é responsável por proteger a infraestrutura que executa AWS os serviços na AWS nuvem. AWS também fornece serviços que você pode usar com segurança. A eficácia da nossa segurança é regularmente testada e verificada por auditores de terceiros como parte dos Programas de conformidade da AWS. Para saber mais sobre os programas de conformidade que se aplicam ao Timestream for InfluxDB, consulte AWS Services in Scope by Compliance Program.
- Segurança na nuvem — Sua responsabilidade é determinada pelo AWS serviço que você usa. Você também é responsável por outros fatores, inclusive a confidencialidade dos dados, os requisitos da organização, as leis e as regulamentações vigentes.

Esta documentação ajudará você a entender como aplicar o modelo de responsabilidade compartilhada ao usar o Timestream para o InfluxDB. Os tópicos a seguir mostram como configurar o Timestream para o InfluxDB para atender aos seus objetivos de segurança e conformidade. Você

também aprenderá a usar outros AWS serviços que podem ajudá-lo a monitorar e proteger seus recursos do Timestream for InfluxDB.

Tópicos

- Visão geral
- Autenticação de banco de dados com Amazon Timestream para InfluxDB
- Como o Amazon Timestream para InfluxDB usa segredos
- Proteção de dados no Timestream for InfluxDB
- Identity and Access Management para Amazon Timestream para InfluxDB
- Registro e monitoramento no Timestream para InfluxDB
- Validação de conformidade do Amazon Timestream para InfluxDB
- Resiliência no Amazon Timestream para InfluxDB
- Segurança de infraestrutura no Amazon Timestream para InfluxDB
- Configuração e análise de vulnerabilidade no Timestream for InfluxDB
- Resposta a incidentes no Timestream para InfluxDB
- Amazon Timestream para API InfluxDB e endpoints de interface () VPC AWS PrivateLink
- Melhores práticas de segurança para Timestream for InfluxDB

Visão geral

Essa documentação ajuda você a entender como aplicar o modelo de responsabilidade compartilhada ao usar o Amazon Timestream para InfluxDB. Os tópicos a seguir mostram como configurar o Amazon Timestream para InfluxDB para atender aos seus objetivos de segurança e conformidade. Você também aprende a usar outros AWS serviços que ajudam a monitorar e proteger seus recursos do Amazon Timestream for InfluxDB.

Você pode gerenciar o acesso aos recursos do Amazon Timestream for InfluxDB e aos seus bancos de dados em uma instância de banco de dados. O método que você usa para gerenciar o acesso depende do tipo de tarefa que o usuário precisa realizar com o Amazon Timestream para InfluxDB:

- Execute sua instância de banco de dados em uma Virtual Private Cloud (VPC) com base no VPC serviço Amazon para controle de acesso à rede.
- Use as políticas de AWS Identity and Access Management (IAM) para atribuir permissões que determinam quem tem permissão para gerenciar os recursos do Amazon Timestream for InfluxDB.

Por exemplo, você pode usar IAM para determinar quem tem permissão para criar, descrever, modificar e excluir instâncias de banco de dados, marcar recursos ou modificar grupos de segurança.

- Use grupos de segurança para controlar quais endereços IP ou EC2 instâncias da Amazon podem se conectar aos seus bancos de dados em uma instância de banco de dados. Quando você cria uma instância de banco de dados pela primeira vez, ela só pode ser acessada por meio de regras especificadas por um grupo de segurança associado.
- Use conexões Secure Socket Layer (SSL) ou Transport Layer Security (TLS) com suas instâncias de banco de dados.
- Use os recursos de segurança do seu mecanismo InfluxDB para controlar quem pode fazer login nos bancos de dados em uma instância de banco de dados. Esses recursos funcionam como se o banco de dados estivesse em sua rede local. Para obter mais informações, consulte Segurança no Timestream para InfluxDB.

> **Note**
>
> Basta configurar a segurança para os casos de uso. Você não precisa configurar o acesso de segurança aos processos gerenciados pelo Amazon Timestream for InfluxDB. Isso inclui a criação de backups, a replicação de dados entre uma instância do banco de dados primário e uma réplica de leitura e outros processos.

Tópicos

- Segurança geral

Segurança geral

Tópicos

- Permissões
- Acesso à rede
- Dependências
- Buckets do S3

Permissões

Os usuários do InfluxDB devem receber permissões de privilégio mínimo. Somente tokens concedidos a usuários específicos, em vez dos tokens do operador, devem ser usados durante a migração.

O Timestream for InfluxDB usa IAM permissões para controlar as permissões do usuário. Recomendamos que os usuários tenham acesso às ações e recursos específicos de que necessitam. Para obter mais informações, consulte Conceder acesso com privilégios mínimos.

Acesso à rede

O script de migração do Influx pode funcionar localmente, migrando dados entre duas instâncias do InfluxDB no mesmo sistema, mas presume-se que o principal caso de uso para migrações será a migração de dados pela rede, seja uma rede local ou pública. Com isso, vêm as considerações de segurança. O script de migração do Influx, por padrão, verificará os TLS certificados das instâncias TLS habilitadas: recomendamos que os usuários habilitem TLS em suas instâncias do InfluxDB e não usem a `--skip-verify` opção do script.

Recomendamos que você use uma lista de permissões para restringir o tráfego de rede às fontes que você espera. Você pode fazer isso limitando o tráfego de rede às instâncias do InfluxDB somente de instâncias conhecidas. IPs

Dependências

As versões principais mais recentes de todas as dependências devem ser usadas, incluindo Influx, InfluxDBCLI, Python, o módulo Requests e dependências opcionais, como e. `mountpoint-s3` `rclone`

Buckets do S3

Se os buckets do S3 forem usados como armazenamento temporário para migração, recomendamos ativarTLS, controlar o controle de versão e desativar o acesso público.

Usando buckets S3 para migração

1. Abra o AWS Management Console, navegue até o Amazon Simple Storage Service e escolha Buckets.
2. Escolha o balde que você deseja usar.
3. Escolha a aba Permissões.

4. Em Block public access (bucket settings) (Bloqueio de acesso público (configurações de bucket), escolha Edit (Editar).
5. Marque Bloquear todo o acesso público.
6. Escolha Salvar alterações.
7. Em Bucket policy (Política de bucket), escolha Edit (Editar).
8. Digite o seguinte, **<example-bucket>**substituindo pelo nome do seu bucket, para impor o uso da TLS versão 1.2 ou posterior para conexões:

```
{
    "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. Escolha Salvar alterações.
10. Escolha a guia Properties (Propriedades).
11. Em Bucket Versioning (Versionamento de bucket), escolha Edit (Editar).
12. Marque Ativar.
13. Escolha Salvar alterações.

Para obter informações sobre as melhores práticas de segurança do bucket Amazon S3, consulte Melhores práticas de segurança para o Amazon Simple Storage Service.

Autenticação de banco de dados com Amazon Timestream para InfluxDB

O Amazon Timestream para InfluxDB oferece suporte a duas maneiras de autenticar usuários do banco de dados.

Senha e acesso A autenticação do banco de dados Token usa métodos diferentes de autenticação no banco de dados. Portanto, um usuário específico pode fazer login em um banco de dados usando apenas um método de autenticação. Em ambos os casos, o InfluxDB realiza toda a administração de contas e API tokens de usuário.

Autenticação com senha

Durante o processo de criação da instância de banco de dados InfluxDB, você criou uma organização, um usuário e uma senha. O usuário tem permissões para gerenciar tudo em sua instância de banco de dados Timestream for InfluxDB. Com essa combinação de nome de usuário e senha, você poderá acessar LogIn sua instância usando o InfluxUI e também usar o Influx CLI para gerar um token de operador.

É necessário um token de operador para criar usuários, excluir buckets, organizações etc. Para obter mais informações, consulte Opções de autenticação do banco de dados.

APIfichas

APIOs tokens do InfluxDB garantem uma interação segura entre o InfluxDB e ferramentas externas, como clientes ou aplicativos. Um API token pertence a um usuário específico e identifica as permissões do InfluxDB na organização do usuário.

Existem três tipos de API tokens no InfluxDB:

- Token de operador: concede acesso completo de leitura e gravação a todas as organizações e a todos os recursos da organização no InfluxDB 2.xOSS. Algumas operações, por exemplo, recuperar a configuração do servidor, exigem permissões do operador. Para criar um token de operador manualmente com a interface do usuário do InfluxDB ou Influx CLI após a conclusão do processo de configuração, você deve usar um token de operador existente ou seu nome de usuário e senha. `api/v2` API Para criar um novo token de operador sem usar um existente, consulte a autenticação CLI de recuperação do influxd.

> **Important**
>
> Como os tokens do operador têm acesso total de leitura e gravação a todas as organizações no banco de dados, recomendamos criar um token de acesso total para cada organização e usá-lo para gerenciar o InfluxDB. Isso ajuda a evitar interações acidentais entre organizações.

- APIToken de acesso total: concede acesso total de leitura e gravação a todos os recursos de uma organização.
- Tokens de leitura/gravação: concede acesso de leitura, acesso de gravação ou ambos a buckets específicos em uma organização.

Todos os InfluxDb tokens são tokens de longa duração sem data de expiração definida, portanto, não é recomendável usar sua operadora ou todos os tokens de acesso para enviar dados de monitoramento de seus clientes ou agentes do Telegraf nem incorporá-los em seus aplicativos de painel. Para esses aplicativos, crie tokens de leitura/gravação apenas com as permissões necessárias para realizar o trabalho. Para obter mais informações sobre como criar um token InfluxDB, consulte Criar um token.

Segredos

Os tokens do operador InfluxDB são gerados na configuração da instância; outros tipos de tokens, como tokens de acesso total e de leitura/gravação, podem ser criados usando a função de rotação multiusuário InfluxCLI, Influx v2 API ou Timestream for InfluxDB. Consulte Gerenciar API tokens para saber como gerar, visualizar, atribuir e excluir tokens.

Recomendamos que você alterne o Timestream para tokens do InfluxDB, geralmente usando AWS Secrets Manager e armazenando tokens por meio de variáveis de ambiente. Consulte Usar tokens para uso de tokens em variáveis de ambiente e Girando o segredo para saber como alternar o Timestream para usuários e tokens do InfluxDB.

Consulte também:

- Segurança de infraestrutura no Amazon Timestream para InfluxDB
- Melhores práticas de segurança para Timestream for InfluxDB

Como o Amazon Timestream para InfluxDB usa segredos

O Timestream for InfluxDB suporta autenticação de nome de usuário e senha por meio da interface do usuário e credenciais de token para conexões de clientes e aplicativos com privilégios mínimos. Os usuários do Timestream for InfluxDB têm `allAccess` permissões em sua organização, enquanto os tokens podem ter qualquer conjunto de permissões. Seguindo as melhores práticas para o gerenciamento seguro de API tokens, os usuários devem ser criados para gerenciar tokens para acesso refinado dentro de uma organização. Informações adicionais sobre as melhores práticas administrativas com o Timestream for InfluxDB podem ser encontradas na documentação do Influxdata.

AWS Secrets Manager é um serviço de armazenamento secreto que você pode usar para proteger credenciais, API chaves e outras informações secretas do banco de dados. Em seguida, em seu código, você pode substituir as credenciais codificadas por uma API chamada para o Secrets Manager. Isso ajuda a garantir que o segredo não possa ser comprometido por alguém examinando seu código, porque o segredo não está lá. Para obter uma visão geral do Secrets Manager, consulte O que é o AWS Secrets Manager.

Quando você cria uma instância de banco de dados, o Timestream for InfluxDB cria automaticamente um segredo de administrador para você usar com a função de rotação multiusuário. AWS Lambda Para alternar o Timestream para usuários e tokens do InfluxDB, você deve criar um novo segredo manualmente para cada usuário ou token que deseja alternar. Cada segredo pode ser configurado para alternar de acordo com uma programação com o uso de uma função Lambda. O processo para configurar um novo segredo rotativo consiste em carregar o código da função Lambda, configurar a função do Lambda, definir o novo segredo e configurar o cronograma de rotação do segredo.

O que há no segredo?

Ao armazenar as credenciais de usuário do Timestream for InfluxDB no segredo, use o seguinte formato.

Usuário único:

```
{
  "engine": "<required: must be set to 'timestream-influxdb'>",
  "username": "<required: username>",
  "password": "<required: password>",
  "dbIdentifier": "<required: DB identifier>"
```

```
}
```

Quando você cria uma instância Timestream para o InfluxDB, um segredo administrativo é armazenado automaticamente no Secrets Manager com credenciais a serem usadas com a função Lambda multiusuário. `adminSecretArn`Defina o como o `Authentication Properties Secret Manager ARN` valor encontrado na página de resumo da instância de banco de dados ou como o ARN de um segredo de administrador. Para criar um novo segredo de administrador, você já deve ter as credenciais associadas e as credenciais devem ter privilégios de administrador.

Ao armazenar as credenciais do token Timestream for InfluxDB no segredo, use o seguinte formato.

Multiusuário:

```
{
  "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']>"
}
```

Ao armazenar as credenciais de administrador do Timestream for InfluxDB no segredo, use o seguinte formato:

Segredo do administrador:

```
{
  "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>"
}
```

Para ativar a rotação automática do segredo, o segredo deve estar na JSON estrutura correta. Veja como alternar o Timestream para Girando o segredo obter segredos do InfluxDB.

Modificar o segredo

As credenciais geradas durante o processo de criação da instância Timestream for InfluxDB são armazenadas em um segredo do Secrets Manager em sua conta. O objeto de GetDbInstanceresposta contém um `influxAuthParametersSecretArn` que mantém o Amazon Resource Name (ARN) em tal segredo. O segredo só será preenchido depois que sua instância Timestream for InfluxDB estiver disponível. Essa é uma READONLY cópia, pois qualquer outra updates/modifications/deletions desse segredo não afeta a instância de banco de dados criada. Se você excluir esse segredo, a APIresposta ainda se referirá ao segredo excluídoARN.

Para criar um novo token na instância Timestream for InfluxDB em vez de armazenar as credenciais de token existentes, você pode criar tokens que não sejam operadores deixando o `token` valor em branco no segredo e usando a função de rotação multiusuário com a variável de ambiente Lambda `AUTHENTICATION_CREATION_ENABLED` definida como. `true` Se você criar um novo token, as permissões definidas no segredo serão atribuídas ao token e não poderão ser alteradas após a primeira rotação bem-sucedida. Para obter mais informações sobre a rotação de segredos, consulte Rotating AWS Secrets Manager Secrets.

Se um segredo for excluído, o usuário ou token associado na instância Timestream for InfluxDB não será excluído.

Girando o segredo

Você usa as funções Lambda de rotação de usuário único e multiusuário do Timestream for InfluxDB para alternar o Timestream para as credenciais de usuário e token do InfluxDB. Use a função Lambda de usuário único para alternar as credenciais do usuário para sua instância Timestream for InfluxDB e use a função Lambda multiusuário para alternar as credenciais de token para sua instância Timestream for InfluxDB.

A rotação de usuários e tokens com as funções Lambda para um ou vários usuários é opcional. As credenciais do Timestream for InfluxDB nunca expiram e qualquer credencial exposta representa um risco de ações maliciosas contra sua instância de banco de dados. A vantagem de alternar as credenciais do Timestream for InfluxDB com o Secrets Manager é uma camada de segurança adicional que limita o vetor de ataque das credenciais expostas à janela de tempo até o próximo ciclo de rotação. Se nenhum mecanismo de rotação estiver em vigor para sua instância de banco de dados, todas as credenciais expostas serão válidas até serem excluídas manualmente.

Você pode configurar o Secrets Manager para alterar automaticamente os segredos para você de acordo com a programação que você especificar. Isso permite substituir segredos de longo prazo por outros de curto prazo, ajudando a reduzir de maneira significativa o risco de comprometimento. Para obter mais informações sobre a rotação de segredos com o Secrets Manager, consulte Rotate AWS Secrets Manager Secrets.

Usuários rotativos

Quando você alterna usuários com a função Lambda de usuário único, uma nova senha aleatória será atribuída ao usuário após cada rotação. Para obter mais informações sobre como ativar a rotação automática, consulte Configurar a rotação automática para AWS segredos do Secrets Manager que não sejam do banco de dados.

Segredos administrativos rotativos

Para alternar um segredo administrativo, você usa a função de rotação de usuário único. Você precisa adicionar `dbIdentifier` os valores `engine` e ao segredo, pois esses valores não são preenchidos automaticamente na inicialização do banco de dados. Veja O que há no segredo? o modelo secreto completo.

Para localizar um segredo administrativo para uma instância do Timestream for InfluxDB, você usa o segredo do administrador da página de resumo da instância ARN do Timestream for InfluxDB. É recomendável alternar todos os segredos de administração do Timestream for InfluxDB, pois os usuários administradores têm permissões elevadas para a instância Timestream for InfluxDB.

Função de alternância do Lambda

Você pode girar um Timestream para um usuário do InfluxDB com a função de rotação de usuário único usando o O que há no segredo? com um novo segredo e adicionando os campos obrigatórios para o usuário do Timestream for InfluxDB. Para obter mais informações sobre funções Lambda de rotação secreta, consulte Rotação por função Lambda.

A função de rotação de usuário único se autentica com a instância de banco de dados Timestream for InfluxDB usando as credenciais definidas no segredo, depois gera uma nova senha aleatória e define a nova senha para o usuário. Para obter mais informações sobre funções Lambda de rotação secreta, consulte Rotação por função Lambda.

Permissões da função de execução da função Lambda

Use a IAM política a seguir como função para a função Lambda de usuário único. A política dá à função Lambda as permissões necessárias para realizar uma rotação secreta para o Timestream para usuários do InfluxDB.

Substitua todos os itens listados abaixo na IAM política pelos valores da sua AWS conta:

- {rotating_secret_arn} — O ARN valor do segredo que está sendo rotacionado pode ser encontrado nos detalhes secretos do Secrets Manager.
- {db_instance_arn} — O Timestream para a instância do InfluxDB ARN pode ser encontrado na página de resumo da instância do Timestream for InfluxDB.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "secretsmanager:UpdateSecretVersionStage"
            ],
     "Resource": "{rotating_secret_arn}"
        },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword"
            ],
            "Resource": "*"
        },
        {
            "Action": [
                "timestream-influxdb:GetDbInstance"
            ],
     "Resource": "{db_instance_arn}",
            "Effect": "Allow"
        }
    ]
```

```
}
```

Tokens rotativos

Você pode girar um token Timestream for InfluxDB com a função de rotação multiusuário usando o O que há no segredo? com um novo segredo e adicionando os campos obrigatórios para seu token Timestream for InfluxDB. Para obter mais informações sobre funções Lambda de rotação secreta, consulte Rotação por função Lambda.

Você pode alternar um token Timestream for InfluxDB usando a função Lambda multiusuário Timestream for InfluxDB. Defina a variável de `AUTHENTICATION_CREATION_ENABLED` ambiente como `true` na configuração do Lambda para permitir a criação de tokens. Para criar um novo token, use o O que há no segredo? como seu valor secreto. Omita o par `token` de valores-chave no novo segredo e defina o como`allAccess`, ou `type` defina as permissões específicas e defina o tipo como. `custom` A função de rotação criará um novo token durante o primeiro ciclo de rotação. Você não pode alterar as permissões do token editando o segredo após a rotação, e qualquer rotação subsequente usará as permissões definidas na instância de banco de dados.

Função de alternância do Lambda

A função de rotação de vários usuários alterna as credenciais do token criando um novo token idêntico de permissão usando as credenciais do administrador no segredo do administrador. A função Lambda valida o valor do token no segredo antes de criar o token substituto, armazenar o novo valor do token no segredo e excluir o token antigo. Se a função Lambda estiver criando um novo token, ela primeiro validará se a variável de `AUTHENTICATION_CREATION_ENABLED` ambiente está definida como`true`, se não há valor de token no segredo e se o tipo de token não é operador de tipo.

Permissões da função de execução da função Lambda

Use a IAM política a seguir como função para a função Lambda multiusuário. A política concede à função Lambda as permissões necessárias para realizar uma rotação secreta para tokens Timestream for InfluxDB.

Substitua todos os itens listados abaixo na IAM política pelos valores da sua AWS conta:

- {rotating_secret_arn} — O ARN valor do segredo que está sendo rotacionado pode ser encontrado nos detalhes secretos do Secrets Manager.
- {authentication_properties_admin_secret_arn} — O segredo do administrador do Timestream for InfluxDB pode ser encontrado na página de resumo da instância do Timestream for InfluxDB. ARN

- {db_instance_arn} — O Timestream para a instância do InfluxDB ARN pode ser encontrado na página de resumo da instância do Timestream for InfluxDB.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "secretsmanager:UpdateSecretVersionStage"
            ],
     "Resource": "{rotating_secret_arn}"
        },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
     "Resource": "{authentication_properties_admin_secret_arn}"
        },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword"
            ],
            "Resource": "*"
        },
        {
            "Action": [
                "timestream-influxdb:GetDbInstance"
            ],
     "Resource": "{db_instance_arn}",
            "Effect": "Allow"
        }
    ]
}
```

Proteção de dados no Timestream for InfluxDB

O modelo de responsabilidade AWS compartilhada O modelo se aplica à proteção de dados no Amazon Timestream for InfluxDB. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa todos os Nuvem AWS. Você é responsável por manter o controle sobre seu conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para obter mais informações sobre privacidade de dados, consulte Privacidade de dados FAQ. Para obter informações sobre proteção de dados na Europa, consulte o Modelo de Responsabilidade AWS Compartilhada e GDPR a postagem no blog AWS de segurança.

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure usuários individuais com AWS IAM Identity Center ou AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use a autenticação multifator (MFA) com cada conta.
- UseSSL/TLSpara se comunicar com AWS os recursos. Exigimos TLS 1,2 e recomendamos TLS 1,3.
- Configure API e registre as atividades do usuário com AWS CloudTrail. Para obter informações sobre o uso de CloudTrail trilhas para capturar AWS atividades, consulte Como trabalhar com CloudTrail trilhas no Guia AWS CloudTrail do usuário.
- Use soluções de AWS criptografia, juntamente com todos os controles de segurança padrão Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sigilosos armazenados no Amazon S3.
- Se você precisar de FIPS 140-3 módulos criptográficos validados ao acessar AWS por meio de uma interface de linha de comando ou umaAPI, use um endpoint. FIPS Para obter mais informações sobre os FIPS endpoints disponíveis, consulte Federal Information Processing Standard (FIPS) 140-3.

É altamente recomendável que nunca sejam colocadas informações de identificação confidenciais, como endereços de e-mail dos seus clientes, em marcações ou campos de formato livre, como um campo Nome. Isso inclui quando você trabalha com o Timestream para InfluxDB ou outro Serviços da AWS usando o console,,, API ou. AWS CLI AWS SDKs Quaisquer dados inseridos em tags ou

campos de texto de formato livre usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, é altamente recomendável que você não inclua informações de credenciais no URL para validar sua solicitação para esse servidor.

Para obter informações mais detalhadas sobre os tópicos de proteção de dados do Timestream for InfluxDB, como criptografia em repouso e gerenciamento de chaves, selecione qualquer um dos tópicos disponíveis abaixo.

Tópicos

- Criptografia inativa
- Criptografia em trânsito

Criptografia inativa

O Timestream para criptografia do InfluxDB em repouso fornece segurança aprimorada ao criptografar todos os seus dados em repouso usando chaves de criptografia armazenadas em ().AWS Key Management ServiceAWS KMS Essa funcionalidade ajuda a reduzir a carga e complexidade operacionais necessárias para proteger dados confidenciais. Com a criptografia de dados em repouso, você pode criar aplicativos confidenciais que atendem a requisitos rigorosos de conformidade e regulamentação de criptografia.

- A criptografia está ativada por padrão em sua instância de banco de dados Timestream for InfluxDB e não pode ser desativada. O algoritmo de criptografia AES -256 padrão do setor é o algoritmo de criptografia padrão usado.
- AWS KMS é usado para criptografia em repouso no Timestream for InfluxDB.
- Você não precisa modificar seus aplicativos cliente de instância de banco de dados para usar criptografia.

Criptografia em trânsito

Todos os seus dados do Timestream for InfluxDB são criptografados em trânsito. Por padrão, todas as comunicações de e para o Timestream for InfluxDB são protegidas usando a criptografia Transport Layer Security (). TLS

O tráfego de e para o Amazon Timestream para InfluxDB é protegido usando TLS as versões 1.2 ou 1.3 suportadas.

Identity and Access Management para Amazon Timestream para InfluxDB

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. IAMos administradores controlam quem pode ser autenticado (conectado) e autorizado (tem permissões) para usar o Timestream para recursos do InfluxDB. IAMé um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Tópicos

- Autenticando com identidades
- Gerenciando acesso usando políticas
- Como o Amazon Timestream para InfluxDB funciona com IAM
- Exemplos de políticas baseadas em identidade para Amazon Timestream for InfluxDB
- Solução de problemas do Amazon Timestream para identidade e acesso ao InfluxDB
- Controlando o acesso a uma instância de banco de dados em um VPC
- Usando funções vinculadas a serviços para Amazon Timestream para InfluxDB
- AWS políticas gerenciadas para Amazon Timestream para InfluxDB
- Conectando-se ao Timestream para InfluxDB por meio de um endpoint VPC

Autenticando com identidades

A autenticação é a forma como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado (conectado AWS) como IAM usuário ou assumindo uma IAM função. Usuário raiz da conta da AWS

Você pode entrar AWS como uma identidade federada usando credenciais fornecidas por meio de uma fonte de identidade. AWS IAM Identity Center Os usuários (do IAM Identity Center), a autenticação de login único da sua empresa e suas credenciais do Google ou do Facebook são exemplos de identidades federadas. Quando você entra como uma identidade federada, seu administrador configurou previamente a federação de identidades usando IAM funções. Ao acessar AWS usando a federação, você está assumindo indiretamente uma função.

Dependendo do tipo de usuário que você é, você pode entrar no AWS Management Console ou no portal de AWS acesso. Para obter mais informações sobre como fazer login em AWS, consulte Como fazer login Conta da AWS no Guia do Início de Sessão da AWS usuário.

Se você acessar AWS programaticamente, AWS fornece um kit de desenvolvimento de software (SDK) e uma interface de linha de comando (CLI) para assinar criptograficamente suas solicitações usando suas credenciais. Se você não usa AWS ferramentas, você mesmo deve assinar as solicitações. Para obter mais informações sobre como usar o método recomendado para você mesmo assinar solicitações, consulte AWS Signature versão 4 para API solicitações no Guia IAM do usuário.

Independente do método de autenticação usado, também pode ser exigido que você forneça informações adicionais de segurança. Por exemplo, AWS recomenda que você use a autenticação multifator (MFA) para aumentar a segurança da sua conta. Para saber mais, consulte Autenticação multifator no Guia do AWS IAM Identity Center usuário e Autenticação AWS multifator IAM no Guia do IAMusuário.

Grupos e usuários do IAM

Um IAMusuário é uma identidade dentro da sua Conta da AWS que tem permissões específicas para uma única pessoa ou aplicativo. Sempre que possível, recomendamos confiar em credenciais temporárias em vez de criar IAM usuários que tenham credenciais de longo prazo, como senhas e chaves de acesso. No entanto, se você tiver casos de uso específicos que exijam credenciais de longo prazo com IAM os usuários, recomendamos que você alterne as chaves de acesso. Para obter mais informações, consulte Alterne as chaves de acesso regularmente para casos de uso que exigem credenciais de longo prazo no Guia do IAMusuário.

Um IAMgrupo é uma identidade que especifica uma coleção de IAM usuários. Não é possível fazer login como um grupo. É possível usar grupos para especificar permissões para vários usuários de uma vez. Os grupos facilitam o gerenciamento de permissões para grandes conjuntos de usuários. Por exemplo, você pode ter um grupo chamado IAMAdminse conceder a esse grupo permissões para administrar IAM recursos.

Usuários são diferentes de perfis. Um usuário é exclusivamente associado a uma pessoa ou a uma aplicação, mas um perfil pode ser assumido por qualquer pessoa que precisar dele. Os usuários têm credenciais permanentes de longo prazo, mas os perfis fornecem credenciais temporárias. Para saber mais, consulte Casos de uso para IAM usuários no Guia IAM do usuário.

IAMfunções

Uma IAMfunção é uma identidade dentro da sua Conta da AWS que tem permissões específicas. É semelhante a um IAM usuário, mas não está associado a uma pessoa específica. Para assumir temporariamente uma IAM função no AWS Management Console, você pode alternar de usuário para IAM função (console). Você pode assumir uma função chamando uma AWS API operação AWS

CLI or ou usando uma personalizadaURL. Para obter mais informações sobre métodos de uso de funções, consulte Métodos para assumir uma função no Guia IAM do usuário.

IAMfunções com credenciais temporárias são úteis nas seguintes situações:

- Acesso de usuário federado: para atribuir permissões a identidades federadas, você pode criar um perfil e definir permissões para ele. Quando uma identidade federada é autenticada, essa identidade é associada ao perfil e recebe as permissões definidas pelo mesmo. Para obter informações sobre funções para federação, consulte Criação de uma função para um provedor de identidade terceirizado no Guia IAM do usuário. Se você usa o IAM Identity Center, configura um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o IAM Identity Center correlaciona o conjunto de permissões a uma função em. IAM Para obter informações sobre conjuntos de permissões, consulte Conjuntos de Permissões no Manual do Usuário do AWS IAM Identity Center .
- Permissões temporárias IAM de IAM usuário — Um usuário ou função pode assumir uma IAM função para assumir temporariamente permissões diferentes para uma tarefa específica.
- Acesso entre contas — Você pode usar uma IAM função para permitir que alguém (um diretor confiável) em uma conta diferente acesse recursos em sua conta. Os perfis são a principal forma de conceder acesso entre contas. No entanto, com alguns Serviços da AWS, você pode anexar uma política diretamente a um recurso (em vez de usar uma função como proxy). Para saber a diferença entre funções e políticas baseadas em recursos para acesso entre contas, consulte Acesso a recursos entre contas IAM no Guia do IAM usuário.
- Acesso entre serviços — Alguns Serviços da AWS usam recursos em outros Serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicativos na Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso usando as permissões do principal de chamada, usando um perfil de serviço ou um perfil vinculado a serviço.
  - Sessões de acesso direto (FAS) — Quando você usa um IAM usuário ou uma função para realizar ações em AWS, você é considerado principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. FASusa as permissões do diretor chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) para fazer solicitações aos serviços posteriores. FASas solicitações são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou com recursos para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer FAS solicitações, consulte Encaminhar sessões de acesso.

  - Função de serviço — Uma função de serviço é uma IAMfunção que um serviço assume para realizar ações em seu nome. Um IAM administrador pode criar, modificar e excluir uma função de serviço internamenteIAM. Para obter mais informações, consulte Criação de uma função para delegar permissões a uma AWS service (Serviço da AWS) no Guia do IAM usuário.
  - Função vinculada ao serviço — Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode presumir a função de executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um IAM administrador pode visualizar, mas não editar, as permissões das funções vinculadas ao serviço.
- Aplicativos em execução na Amazon EC2 — Você pode usar uma IAM função para gerenciar credenciais temporárias para aplicativos que estão sendo executados em uma EC2 instância e fazendo AWS CLI AWS API solicitações. Isso é preferível ao armazenamento de chaves de acesso na EC2 instância. Para atribuir uma AWS função a uma EC2 instância e disponibilizá-la para todos os aplicativos, você cria um perfil de instância anexado à instância. Um perfil de instância contém a função e permite que programas em execução na EC2 instância recebam credenciais temporárias. Para obter mais informações, consulte Como usar uma IAM função para conceder permissões a aplicativos executados em EC2 instâncias da Amazon no Guia IAM do usuário.

Gerenciando acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política é um objeto AWS que, quando associada a uma identidade ou recurso, define suas permissões. AWS avalia essas políticas quando um principal (usuário, usuário raiz ou sessão de função) faz uma solicitação. As permissões nas políticas determinam se a solicitação será permitida ou negada. A maioria das políticas é armazenada AWS como JSON documentos. Para obter mais informações sobre a estrutura e o conteúdo dos documentos de JSON política, consulte Visão geral das JSON políticas no Guia IAM do usuário.

Os administradores podem usar AWS JSON políticas para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

Por padrão, usuários e funções não têm permissões. Para conceder permissão aos usuários para realizar ações nos recursos de que precisam, um IAM administrador pode criar IAM políticas. O administrador pode então adicionar as IAM políticas às funções e os usuários podem assumir as funções.

IAMas políticas definem permissões para uma ação, independentemente do método usado para realizar a operação. Por exemplo, suponha que você tenha uma política que permite a ação `iam:GetRole`. Um usuário com essa política pode obter informações de função do AWS Management Console AWS CLI, do ou do AWS API.

Políticas baseadas em identidade

Políticas baseadas em identidade são documentos de políticas de JSON permissões que você pode anexar a uma identidade, como um IAM usuário, grupo de usuários ou função. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Definir IAM permissões personalizadas com políticas gerenciadas pelo cliente no Guia do IAMusuário.

As políticas baseadas em identidade podem ser categorizadas ainda adicionalmente como políticas em linha ou políticas gerenciadas. As políticas em linha são anexadas diretamente a um único usuário, grupo ou perfil. As políticas gerenciadas são políticas autônomas que você pode associar a vários usuários, grupos e funções em seu Conta da AWS. As políticas AWS gerenciadas incluem políticas gerenciadas e políticas gerenciadas pelo cliente. Para saber como escolher entre uma política gerenciada ou uma política em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do IAMusuário.

Políticas baseadas no recurso

Políticas baseadas em recursos são documentos JSON de política que você anexa a um recurso. Exemplos de políticas baseadas em recursos são políticas de confiança de IAM funções e políticas de bucket do Amazon S3. Em serviços que suportem políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações um principal especificado pode executar nesse recurso e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas de uma política baseada IAM em recursos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLssão semelhantes às políticas baseadas em recursos, embora não usem o formato de documento JSON de política.

Amazon S3, AWS WAF, e Amazon VPC são exemplos de serviços que oferecem suporte. ACLs Para saber maisACLs, consulte a visão geral da lista de controle de acesso (ACL) no Guia do desenvolvedor do Amazon Simple Storage Service.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais menos comuns. Esses tipos de política podem definir o máximo de permissões concedidas a você pelos tipos de política mais comuns.

- Limites de permissões — Um limite de permissões é um recurso avançado no qual você define as permissões máximas que uma política baseada em identidade pode conceder a uma IAM entidade (IAMusuário ou função). É possível definir um limite de permissões para uma entidade. As permissões resultantes são a interseção das políticas baseadas em identidade de uma entidade com seus limites de permissões. As políticas baseadas em recurso que especificam o usuário ou o perfil no campo `Principal` não são limitadas pelo limite de permissões. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações sobre limites de permissões, consulte Limites de permissões para IAM entidades no Guia IAM do usuário.
- Políticas de controle de serviço (SCPs) — SCPs são JSON políticas que especificam as permissões máximas para uma organização ou unidade organizacional (OU) em AWS Organizations. AWS Organizations é um serviço para agrupar e gerenciar centralmente vários Contas da AWS que sua empresa possui. Se você habilitar todos os recursos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as suas contas. Os SCP limites de permissões para entidades nas contas dos membros, incluindo cada uma Usuário raiz da conta da AWS. Para obter mais informações sobre Organizations eSCPs, consulte Políticas de controle de serviços no Guia AWS Organizations do Usuário.
- Políticas de sessão: são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para um perfil ou um usuário federado. As permissões da sessão resultante são a interseção das políticas baseadas em identidade do usuário ou do perfil e das políticas de sessão. As permissões também podem ser provenientes de uma política baseada em atributo. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações, consulte Políticas de sessão no Guia IAM do usuário.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de política estão envolvidos, consulte Lógica de avaliação de políticas no Guia IAM do usuário.

Como o Amazon Timestream para InfluxDB funciona com IAM

IAMrecursos que você pode usar com o Amazon Timestream para InfluxDB

IAMrecurso	Timestream para suporte ao InfluxDB
Políticas baseadas em identidade	Sim
Políticas baseadas em recursos	Não
Ações das políticas	Sim
Atributos de políticas	Sim
Chaves de condição de políticas	Não
ACLs	Não
ABAC(tags nas políticas)	Sim
Credenciais temporárias	Sim
Permissões de entidade principal	Sim
Perfis de serviço	Não
Funções vinculadas ao serviço	Sim

Para obter uma visão de alto nível de como o Timestream for InfluxDB e outros AWS serviços funcionam com a maioria dos IAM recursos, consulte AWS os serviços que funcionam no Guia do Usuário. IAM IAM

Políticas baseadas em identidade para Timestream for InfluxDB

Compatível com políticas baseadas em identidade: Sim

Políticas baseadas em identidade são documentos de políticas de JSON permissões que você pode anexar a uma identidade, como um IAM usuário, grupo de usuários ou função. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Definir IAM permissões personalizadas com políticas gerenciadas pelo cliente no Guia do IAMusuário.

Com políticas IAM baseadas em identidade, você pode especificar ações e recursos permitidos ou negados, bem como as condições sob as quais as ações são permitidas ou negadas. Você não pode especificar a entidade principal em uma política baseada em identidade porque ela se aplica ao usuário ou perfil ao qual ela está anexada. Para saber mais sobre todos os elementos que você pode usar em uma JSON política, consulte a referência IAM JSON de elementos de política no Guia IAM do usuário.

Exemplos de políticas baseadas em identidade para Timestream for InfluxDB

Para ver exemplos de políticas baseadas em identidade do Timestream for InfluxDB, consulte.. Exemplos de políticas baseadas em identidade para Amazon Timestream for InfluxDB

Políticas baseadas em recursos no Timestream for InfluxDB

Suporte a políticas baseadas em recursos: não

Políticas baseadas em recursos são documentos JSON de política que você anexa a um recurso. Exemplos de políticas baseadas em recursos são políticas de confiança de IAM funções e políticas de bucket do Amazon S3. Em serviços que suportem políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações um principal especificado pode executar nesse recurso e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Para habilitar o acesso entre contas, você pode especificar uma conta ou IAM entidades inteiras em outra conta como principal em uma política baseada em recursos. Adicionar uma entidade principal entre contas à política baseada em recurso é apenas metade da tarefa de estabelecimento da relação de confiança. Quando o principal e o recurso são diferentes Contas da AWS, um IAM administrador na conta confiável também deve conceder permissão à entidade principal (usuário

ou função) para acessar o recurso. Eles concedem permissão ao anexar uma política baseada em identidade para a entidade. No entanto, se uma política baseada em recurso conceder acesso a uma entidade principal na mesma conta, nenhuma política baseada em identidade adicional será necessária. Para obter mais informações, consulte Acesso a recursos entre contas IAM no Guia do IAM usuário.

Ações políticas para Timestream for InfluxDB

Compatível com ações de políticas: Sim

Os administradores podem usar AWS JSON políticas para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O `Action` elemento de uma JSON política descreve as ações que você pode usar para permitir ou negar acesso em uma política. As ações de política geralmente têm o mesmo nome da AWS API operação associada. Há algumas exceções, como ações somente de permissão que não têm uma operação correspondente. API Algumas operações também exigem várias ações em uma política. Essas ações adicionais são chamadas de ações dependentes.

Incluem ações em uma política para conceder permissões para executar a operação associada.

Para ver uma lista de ações do Timestream para o InfluxDB, consulte Ações definidas pelo Amazon Timestream para o InfluxDB na Referência de autorização de serviço.

As ações de política no Timestream for InfluxDB usam o seguinte prefixo antes da ação:

```
timestream-influxdb
```

Para especificar várias ações em uma única instrução, separe-as com vírgulas.

```
"Action": [
      "timestream-influxdb:action1",
      "timestream-influxdb:action2"
         ]
```

Você também pode especificar várias ações usando caracteres-curinga (*). Por exemplo, para especificar todas as ações que começam com a palavra `Describe`, inclua a seguinte ação:

```
"Action": "timestream-influxdb:Describe*"
```

Recursos de política para Timestream for InfluxDB

Compatível com recursos de políticas: Sim

Os administradores podem usar AWS JSON políticas para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento `Resource` JSON de política especifica o objeto ou objetos aos quais a ação se aplica. As instruções devem incluir um elemento `Resource` ou `NotResource`. Como prática recomendada, especifique um recurso usando seu Amazon Resource Name (ARN). Isso pode ser feito para ações que oferecem compatibilidade com um tipo de recurso específico, conhecido como permissões em nível de recurso.

Para ações que não oferecem compatibilidade com permissões em nível de recurso, como operações de listagem, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

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

Para ver uma lista dos tipos de recursos do Timestream for InfluxDB e seusARNs, consulte Resources Defined by Amazon Timestream for InfluxDB na Referência de Autorização de Serviço. Para saber com quais ações você pode especificar cada recurso, consulte Ações definidas pelo Amazon Timestream para InfluxDB. ARN

Chaves de condição de política para Timestream for InfluxDB

Suporta chaves de condição de política específicas do serviço: Não

Os administradores podem usar AWS JSON políticas para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento `Condition` (ou bloco `Condition`) permite que você especifique condições nas quais uma instrução estiver em vigor. O elemento `Condition` é opcional. É possível criar expressões condicionais que usem agentes de condição, como “igual a” ou “menor que”, para fazer a condição da política corresponder aos valores na solicitação.

Se você especificar vários elementos `Condition` em uma instrução ou várias chaves em um único `Condition` elemento, a AWS os avaliará usando uma operação lógica `AND`. Se você especificar vários valores para uma única chave de condição, AWS avalia a condição usando uma `OR` operação lógica. Todas as condições devem ser atendidas antes que as permissões da instrução sejam concedidas.

Você também pode usar variáveis de espaço reservado ao especificar condições. Por exemplo, você pode conceder permissão a um IAM usuário para acessar um recurso somente se ele estiver marcado com o nome de IAM usuário. Para obter mais informações, consulte elementos de IAM política: variáveis e tags no Guia IAM do usuário.

AWS suporta chaves de condição globais e chaves de condição específicas do serviço. Para ver todas as chaves de condição AWS globais, consulte as chaves de contexto de condição AWS global no Guia IAM do usuário.

Listas de controle de acesso (ACLs) no Timestream para InfluxDB

SuportesACLs: Não

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLssão semelhantes às políticas baseadas em recursos, embora não usem o formato de documento JSON de política.

Controle de acesso baseado em atributos (ABAC) com Timestream para InfluxDB

Suportes ABAC (tags nas políticas): Sim

O controle de acesso baseado em atributos (ABAC) é uma estratégia de autorização que define permissões com base em atributos. Em AWS, esses atributos são chamados de tags. Você pode anexar tags a IAM entidades (usuários ou funções) e a muitos AWS recursos. Marcar entidades e recursos é a primeira etapa doABAC. Em seguida, você cria ABAC políticas para permitir operações quando a tag do diretor corresponde à tag do recurso que ele está tentando acessar.

ABACé útil em ambientes que estão crescendo rapidamente e ajuda em situações em que o gerenciamento de políticas se torna complicado.

Para controlar o acesso baseado em tags, forneça informações sobre as tags no elemento de condição de uma política usando as `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou chaves de condição `aws:TagKeys`.

Se um serviço oferecer suporte às três chaves de condição para cada tipo de recurso, o valor será Sim para o serviço. Se um serviço oferecer suporte às três chaves de condição somente para alguns tipos de recursos, o valor será Parcial

Para obter mais informações sobreABAC, consulte Definir permissões com ABAC autorização no Guia IAM do usuário. Para ver um tutorial com etapas de configuraçãoABAC, consulte Usar controle de acesso baseado em atributos (ABAC) no Guia do IAMusuário.

Usando credenciais temporárias com Timestream para InfluxDB

Compatível com credenciais temporárias: Sim

Alguns Serviços da AWS não funcionam quando você faz login usando credenciais temporárias. Para obter informações adicionais, incluindo quais Serviços da AWS funcionam com credenciais temporárias, consulte Serviços da AWS esse trabalho IAM no Guia do IAM usuário.

Você está usando credenciais temporárias se fizer login AWS Management Console usando qualquer método, exceto um nome de usuário e senha. Por exemplo, quando você acessa AWS usando o link de login único (SSO) da sua empresa, esse processo cria automaticamente credenciais temporárias. Você também cria automaticamente credenciais temporárias quando faz login no console como usuário e, em seguida, alterna perfis. Para obter mais informações sobre a troca de funções, consulte Alternar de um usuário para uma IAM função (console) no Guia IAM do usuário.

Você pode criar manualmente credenciais temporárias usando o AWS CLI ou AWS API. Em seguida, você pode usar essas credenciais temporárias para acessar AWS. AWS recomenda que você gere credenciais temporárias dinamicamente em vez de usar chaves de acesso de longo prazo. Para obter mais informações, consulte Credenciais de segurança temporárias emIAM.

Permissões principais entre serviços para Timestream for InfluxDB

Suporta sessões de acesso direto (FAS): Sim

Quando você usa um IAM usuário ou uma função para realizar ações em AWS, você é considerado principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. FASusa as permissões do diretor chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) para fazer solicitações aos serviços posteriores. FASas solicitações são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou com recursos para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer FAS solicitações, consulte Encaminhar sessões de acesso.

Funções de serviço do Timestream for InfluxDB

Compatível com perfis de serviço: não

Uma função de serviço é uma IAMfunção que um serviço assume para realizar ações em seu nome. Um IAM administrador pode criar, modificar e excluir uma função de serviço internamenteIAM. Para obter mais informações, consulte Criação de uma função para delegar permissões a uma AWS service (Serviço da AWS) no Guia do IAM usuário.

> **Warning**
>
> Alterar as permissões de uma função de serviço pode interromper a funcionalidade do Timestream for InfluxDB. Edite as funções de serviço somente quando o Timestream for InfluxDB fornecer orientação para fazer isso.

Funções vinculadas ao serviço para Timestream for InfluxDB

Suporte a perfis vinculados a serviços: sim

Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode presumir a função de executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um IAM administrador pode visualizar, mas não editar, as permissões das funções vinculadas ao serviço.

Para obter detalhes sobre como criar ou gerenciar funções vinculadas a serviços, consulte AWS serviços que funcionam com. IAM Encontre um serviço na tabela que inclua um `Yes` na coluna Função vinculada ao serviço. Escolha o link Sim para visualizar a documentação do perfil vinculado a serviço desse serviço.

Exemplos de políticas baseadas em identidade para Amazon Timestream for InfluxDB

Por padrão, usuários e funções não têm permissão para criar ou modificar o Timestream para recursos do InfluxDB. Eles também não podem realizar tarefas usando o AWS Management Console, AWS Command Line Interface (AWS CLI) ou AWS API. Para conceder permissão aos usuários para realizar ações nos recursos de que precisam, um IAM administrador pode criar IAM políticas. O administrador pode então adicionar as IAM políticas às funções e os usuários podem assumir as funções.

Para saber como criar uma política IAM baseada em identidade usando esses exemplos de documentos de JSON política, consulte Criar IAM políticas (console) no Guia do IAMusuário.

Para obter detalhes sobre ações e tipos de recursos definidos pelo Timestream para InfluxDB, incluindo o formato do ARNs para cada um dos tipos de recursos, consulte Ações, recursos e chaves de condição do Amazon Timestream para InfluxDB na Referência de autorização de serviço.

Tópicos

- Melhores práticas de política

- Usando o console Timestream para InfluxDB
- Permitir que usuários visualizem suas próprias permissões
- Acessar um bucket do Amazon S3
- Permitindo todas as operações
- Crie, descreva, exclua e atualize uma instância de banco de dados

Melhores práticas de política

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir recursos do Timestream for InfluxDB em sua conta. Essas ações podem incorrer em custos para seus Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com as políticas AWS gerenciadas e avance para as permissões de privilégios mínimos — Para começar a conceder permissões aos seus usuários e cargas de trabalho, use as políticas AWS gerenciadas que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis no seu Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo AWS cliente que sejam específicas para seus casos de uso. Para obter mais informações, consulte políticas AWS gerenciadas ou políticas AWS gerenciadas para funções de trabalho no Guia IAM do usuário.
- Aplique permissões com privilégios mínimos — Ao definir permissões com IAM políticas, conceda somente as permissões necessárias para realizar uma tarefa. Você faz isso definindo as ações que podem ser executadas em atributos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para obter mais informações sobre IAM como usar para aplicar permissões, consulte Políticas e permissões IAM no Guia IAM do usuário.
- Use condições nas IAM políticas para restringir ainda mais o acesso — Você pode adicionar uma condição às suas políticas para limitar o acesso a ações e recursos. Por exemplo, você pode escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usandoSSL. Você também pode usar condições para conceder acesso às ações de serviço se elas forem usadas por meio de uma ação específica AWS service (Serviço da AWS), como AWS CloudFormation. Para obter mais informações, consulte elementos IAM JSON da política: Condição no Guia IAM do usuário.
- Use o IAM Access Analyzer para validar suas IAM políticas e garantir permissões seguras e funcionais — o IAM Access Analyzer valida políticas novas e existentes para que as políticas sigam a linguagem da IAM política (JSON) e as melhores práticas. IAM IAMO Access Analyzer fornece mais de 100 verificações de políticas e recomendações práticas para ajudá-lo a criar

políticas seguras e funcionais. Para obter mais informações, consulte Validar políticas com o IAM Access Analyzer no Guia do IAMUsuário.

- Exigir autenticação multifatorial (MFA) — Se você tiver um cenário que exija IAM usuários ou um usuário root Conta da AWS, ative MFA para obter segurança adicional. Para exigir MFA quando API as operações são chamadas, adicione MFA condições às suas políticas. Para obter mais informações, consulte APIAcesso seguro MFA no Guia do IAM usuário.

Para obter mais informações sobre as melhores práticas emIAM, consulte as melhores práticas de segurança IAM no Guia IAM do usuário.

Usando o console Timestream para InfluxDB

Para acessar o console Amazon Timestream for InfluxDB, você deve ter um conjunto mínimo de permissões. Essas permissões devem permitir que você liste e visualize detalhes sobre os recursos do Timestream for InfluxDB em seu. Conta da AWS Se você criar uma política baseada em identidade que seja mais restritiva do que as permissões mínimas necessárias, o console não funcionará como pretendido para entidades (usuários ou perfis) com essa política.

Você não precisa permitir permissões mínimas do console para usuários que estão fazendo chamadas somente para AWS CLI o. ou AWS API o. Em vez disso, permita o acesso somente às ações que correspondam à API operação que eles estão tentando realizar.

Para garantir que usuários e funções ainda possam usar o console Timestream for InfluxDB, anexe também o Timestream for InfluxDB ou a política gerenciada às entidades`ConsoleAccess`. `ReadOnly` AWS Para obter mais informações, consulte Adicionar permissões a um usuário no Guia do IAM usuário.

Permitir que usuários visualizem suas próprias permissões

Este exemplo mostra como você pode criar uma política que permita IAM aos usuários visualizar as políticas embutidas e gerenciadas que estão anexadas à identidade do usuário. Essa política inclui permissões para concluir essa ação no console ou programaticamente usando o AWS CLI ou. 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": "*"
        }
    ]
}
```

Acessar um bucket do Amazon S3

Neste exemplo, você deseja conceder a um IAM usuário da sua AWS conta acesso a um dos seus buckets do Amazon S3,. `examplebucket` Você também deseja permitir que o usuário adicione, atualize e exclua objetos.

Além de conceder as permissões `s3:PutObject`, `s3:GetObject` e `s3:DeleteObject` ao usuário, a política também concede as permissões `s3:ListAllMyBuckets`, `s3:GetBucketLocation` e `s3:ListBucket`. Estas são permissões adicionais, exigidas pelo console. As ações `s3:PutObjectAcl` e `s3:GetObjectAcl` também são necessárias para copiar, recortar e colar objetos no console. Para obter uma demonstração de exemplo que concede permissões aos usuários e testa-os ao usar o console, consulte Demonstração de exemplo: Usar políticas de usuário para controlar o acesso a seu 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/*"
      }
   ]
}
```

Permitindo todas as operações

A seguir está um exemplo de política que permite todas as operações no Timestream para o InfluxDB.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "timestream-influxdb:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Crie, descreva, exclua e atualize uma instância de banco de dados

O exemplo de política a seguir permite que um usuário crie, descreva, exclua e atualize uma instância de banco de dados`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"
        }
    ]
}
```

Solução de problemas do Amazon Timestream para identidade e acesso ao InfluxDB

Use as informações a seguir para ajudá-lo a diagnosticar e corrigir problemas comuns que você pode encontrar ao trabalhar com o Timestream for InfluxDB e. IAM

Tópicos

- Não estou autorizado a realizar uma ação no Timestream for InfluxDB
- Quero permitir que pessoas fora da minha AWS conta acessem meus recursos do Timestream for InfluxDB

Não estou autorizado a realizar uma ação no Timestream for InfluxDB

Se isso AWS Management Console indicar que você não está autorizado a realizar uma ação, entre em contato com o administrador para obter ajuda. O administrador é a pessoa que forneceu o seu nome de usuário e senha.

O erro do exemplo a seguir ocorre quando o usuário `mateojackson` tenta usar o console para visualizar detalhes sobre um recurso do *`my-example-widget`* fictício, mas não tem as permissões fictícias do `timestream-influxdb:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 timestream-influxdb:GetWidget on resource: my-example-widget
```

Neste caso, Mateo pede ao administrador para atualizar suas políticas e permitir o acesso ao recurso *`my-example-widget`* usando a ação `timestream-influxdb:`*`GetWidget`*.

Quero permitir que pessoas fora da minha AWS conta acessem meus recursos do Timestream for InfluxDB

Você pode criar um perfil que os usuários de outras contas ou pessoas fora da sua organização podem usar para acessar seus recursos. Você pode especificar quem é confiável para assumir o perfil. Para serviços que oferecem suporte a políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Controlando o acesso a uma instância de banco de dados em um VPC
- Para saber se o Timestream for InfluxDB oferece suporte a esses recursos, consulte Como o Amazon Timestream for InfluxDB funciona com. IAM
- Para saber como fornecer acesso aos seus recursos em todas AWS as contas que você possui, consulte Fornecer acesso a um IAM usuário em outra AWS conta que você possui no Guia do IAM usuário.
- Para saber como fornecer acesso aos seus recursos para AWS contas de terceiros, consulte Como fornecer acesso a AWS contas pertencentes a terceiros no Guia do IAM usuário.
- Para saber como fornecer acesso por meio da federação de identidades, consulte Fornecendo acesso a usuários autenticados externamente (federação de identidades) no Guia do IAMusuário.
- Para saber a diferença entre usar funções e políticas baseadas em recursos para acesso entre contas, consulte Como as IAM funções diferem das políticas baseadas em recursos no Guia do usuário. IAM

Controlando o acesso a uma instância de banco de dados em um VPC

Usando a Amazon Virtual Private Cloud (AmazonVPC), você pode lançar AWS recursos, como o Amazon Timestream para instâncias de banco de dados InfluxDB, em uma nuvem privada virtual (). VPC Ao usar a AmazonVPC, você tem controle sobre seu ambiente de rede virtual. É possível escolher seu próprio intervalo de endereços IP, criar sub-redes e configurar o roteamento e listas de controle de acesso.

Um grupo VPC de segurança controla o acesso às instâncias de banco de dados dentro de umVPC. Cada regra VPC de grupo de segurança permite que uma fonte específica acesse uma instância de banco de dados em uma VPC que esteja associada a esse grupo VPC de segurança. A origem pode ser um intervalo de endereços (por exemplo, 203.0.113.0/24) ou outro grupo de segurança. VPC Ao especificar um grupo de VPC segurança como origem, você permite o tráfego de entrada de todas as instâncias (normalmente servidores de aplicativos) que usam o grupo de VPC segurança de origem. Antes de tentar se conectar à sua instância de banco de dados, configure-a VPC para seu caso de uso. A seguir estão os cenários comuns para acessar uma instância de banco de dados em umVPC:

Uma instância de banco de dados em uma instância VPC acessada por uma EC2 instância da Amazon na mesma VPC

> Um uso comum de uma instância de banco de dados em a VPC é compartilhar dados com um servidor de aplicativos que está sendo executado em uma EC2 instância na mesmaVPC. A EC2 instância pode executar um servidor web com um aplicativo que interage com a instância de banco de dados.

Uma instância de banco de dados em uma instância VPC acessada por uma EC2 instância em outra VPC

> Em alguns casos, sua instância de banco de dados está em uma instância VPC diferente da EC2 instância que você está usando para acessá-la. Nesse caso, você pode usar o VPC peering para acessar a instância de banco de dados.

Uma instância de banco de dados em um aplicativo cliente VPC acessado pela Internet

> Para acessar uma instância de banco VPC de dados em um aplicativo cliente pela Internet, você configura uma VPC com uma única sub-rede pública e usa as sub-redes públicas para criar a instância de banco de dados. Você também configura um gateway de Internet no VPC para permitir a comunicação pela Internet. Para se conectar a uma instância de banco de dados de fora delaVPC, a instância de banco de dados deve estar acessível ao público. Além disso, o acesso deve ser concedido usando as regras de entrada do grupo de segurança da instância de banco de dados, e os outros requisitos devem ser atendidos.

Para obter mais informações sobre grupos VPC de segurança, consulte Grupos de segurança no Guia do usuário da Amazon Virtual Private Cloud.

Para obter detalhes sobre como se conectar a uma instância de banco de dados Timestream for InfluxDB, consulte. Conectando-se a uma instância de banco de dados Amazon Timestream para InfluxDB

Cenário de grupos de segurança

Um uso comum de uma instância de banco de dados em um VPC é compartilhar dados com um servidor de aplicativos executado em uma EC2 instância da Amazon na mesmaVPC, que é acessado por um aplicativo cliente fora doVPC. Para esse cenário, você usa o Timestream para InfluxDB e VPC páginas no AWS Management Console ou o Timestream para InfluxDB e EC2 API operações para criar as instâncias e grupos de segurança necessários:

1. Crie um grupo VPC de segurança (por exemplo,`sg-0123ec2example`) e defina regras de entrada que usem os endereços IP do aplicativo cliente como origem. Esse grupo de segurança permite que seu aplicativo cliente se conecte a EC2 instâncias em um VPC que usa esse grupo de segurança.
2. Crie uma EC2 instância para o aplicativo e adicione a EC2 instância ao grupo VPC de segurança (`sg-0123ec2example`) que você criou na etapa anterior.
3. Crie um segundo grupo de VPC segurança (por exemplo,`sg-6789rdsexample`) e crie uma nova regra especificando o grupo de VPC segurança que você criou na etapa 1 (`sg-0123ec2example`) como origem.
4. Crie uma nova instância de banco de dados e adicione a instância de banco de dados ao grupo de VPC segurança (`sg-6789rdsexample`) que você criou na etapa anterior. Ao criar o banco de dados, use o mesmo número de porta especificado para a regra VPC security group (`sg-6789rdsexample`) que você criou na etapa 3.

Criando um grupo VPC de segurança

Você pode criar um grupo VPC de segurança para uma instância de banco de dados usando o VPC console. Para obter informações sobre a criação de um grupo de segurança, consulte Grupos de segurança no Guia do usuário da Amazon Virtual Private Cloud.

Associar um grupo de segurança a uma instância de banco de dados

Você pode associar um grupo de segurança a uma instância de banco de dados usando Update no console Timestream for InfluxDB, `UpdateDBInstance` Timestream for InfluxDB ou o comando. API `update-db-instance` AWS CLI

O CLI exemplo a seguir associa um grupo de VPC segurança específico e remove grupos de segurança de banco de dados da instância de banco de dados.

```
aws timestream-influxdb update-db-instance --identifier dbName --vpc-security-group-
ids sg-ID
```

Para ter mais informações sobre como modificar uma instância de banco de dados , consulte Atualizar instâncias de banco de dados.

Usando funções vinculadas a serviços para Amazon Timestream para InfluxDB

O Amazon Timestream para InfluxDB AWS Identity and Access Management usa () funções vinculadas ao serviço. IAM Uma função vinculada ao serviço é um tipo exclusivo de IAM função vinculada diretamente a um AWS serviço, como o Amazon Timestream for InfluxDB. As funções vinculadas ao serviço Amazon Timestream para InfluxDB são predefinidas pelo Amazon Timestream para InfluxDB. Eles incluem todas as permissões que o serviço exige para chamar AWS serviços em nome de suas instâncias de banco de dados.

Uma função vinculada ao serviço facilita a configuração do Amazon Timestream para o InfluxDB porque você não precisa adicionar manualmente as permissões necessárias. As funções já existem em sua AWS conta, mas estão vinculadas aos casos de uso do Amazon Timestream para InfluxDB e têm permissões predefinidas. Somente o Amazon Timestream for InfluxDB pode assumir essas funções, e somente essas funções podem usar a política de permissões predefinida. É possível excluir as funções somente depois de primeiro excluir seus recursos relacionados. Isso protege seus recursos do Amazon Timestream for InfluxDB porque você não pode remover inadvertidamente as permissões necessárias para acessar os recursos.

Para obter informações sobre outros serviços que oferecem suporte a funções vinculadas a serviços, consulte AWS Serviços que funcionam com IAM e procure os serviços que têm Sim na coluna Função vinculada ao serviço. Escolha um Sim com um link para visualizar a documentação da função vinculada a esse serviço.

Sumário

- Permissões de função vinculadas ao serviço para Amazon Timestream para InfluxDB

- Criando uma função vinculada ao serviço () IAM
- Editando a descrição de uma função vinculada ao serviço do Amazon Timestream para InfluxDB
  - Editando uma descrição de função vinculada ao serviço (console) IAM
  - Editando uma descrição de função vinculada ao serviço () IAM CLI
  - Editando uma descrição de função vinculada ao serviço () IAM API
- Excluindo uma função vinculada ao serviço do Amazon Timestream para InfluxDB
  - Limpar uma função vinculada ao serviço
  - Excluindo uma função vinculada ao serviço (console) IAM
  - Excluindo uma função vinculada ao serviço () IAM CLI
  - Excluindo uma função vinculada ao serviço () IAM API
- Regiões suportadas pelo Amazon Timestream para funções vinculadas ao serviço InfluxDB

Permissões de função vinculadas ao serviço para Amazon Timestream para InfluxDB

O Amazon Timestream for InfluxDB usa a função vinculada ao serviço AmazonTimestreamInfluxDBServiceRolePolicychamada — Essa política permite que o Timestream for InfluxDB gerencie recursos em seu nome conforme necessário para gerenciar seus clusters. AWS

A política de permissões AmazonTimestreamInflux DBServiceRolePolicy de função vinculada ao serviço permite que o Amazon Timestream for InfluxDB conclua as seguintes ações nos recursos especificados:

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

```
}
```

Para permitir que uma IAM entidade crie funções AmazonTimestreamInflux DBServiceRolePolicy vinculadas a serviços

Adicione a seguinte declaração de política às permissões dessa IAM entidade:

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

Para permitir que uma IAM entidade exclua funções AmazonTimestreamInflux DBServiceRolePolicy vinculadas ao serviço

Adicione a seguinte declaração de política às permissões dessa IAM entidade:

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

Como alternativa, você pode usar uma política AWS gerenciada para fornecer acesso total ao Amazon Timestream para InfluxDB.

Criando uma função vinculada ao serviço () IAM

Não é necessário criar manualmente uma função vinculada ao serviço. Quando você cria uma instância de banco de dados, o Amazon Timestream for InfluxDB cria a função vinculada ao serviço para você.

Se excluir essa função vinculada ao serviço e precisar criá-la novamente, você poderá usar esse mesmo processo para recriar a função em sua conta. Quando você cria uma instância de banco de dados, o Amazon Timestream for InfluxDB cria a função vinculada ao serviço para você novamente.

Editando a descrição de uma função vinculada ao serviço do Amazon Timestream para InfluxDB

O Amazon Timestream para InfluxDB não permite que você edite a função vinculada ao serviço. AmazonTimestreamInflux DBServiceRolePolicy Depois que criar um perfil vinculado ao serviço, você não poderá alterar o nome do perfil, pois várias entidades podem fazer referência a ele. No entanto, você pode editar a descrição da função usandoIAM.

Editando uma descrição de função vinculada ao serviço (console) IAM

Você pode usar o IAM console para editar uma descrição de função vinculada ao serviço.

Para editar a descrição de uma função vinculada ao serviço (console)

1. No painel de navegação esquerdo do IAM console, escolha Funções.
2. Escolha o nome da função a ser modificada.
3. No extremo direito da Descrição da função, escolha Editar.
4. Insira uma nova descrição na caixa e escolha Salvar.

Editando uma descrição de função vinculada ao serviço () IAM CLI

Você pode usar IAM as operações do AWS Command Line Interface para editar uma descrição de função vinculada ao serviço.

Para alterar a descrição de uma função vinculada ao serviço () CLI

1. (Opcional) Para ver a descrição atual de uma função, use a IAM operação AWS CLI for`get-role`.

Example

```
$ aws iam get-role --role-name AmazonTimestreamInfluxDBServiceRolePolicy
```

Use o nome da função, não oARN, para se referir às funções com as CLI operações. Por exemplo, se uma função tiver o seguinteARN:`arn:aws:iam::123456789012:role/myrole`, consulte a função como**`myrole`**.

2. Para atualizar a descrição de uma função vinculada ao serviço, use a operação AWS CLI forIAM. `update-role-description`

   Linux e macOS

```
$ aws iam update-role-description \
    --role-name AmazonTimestreamInfluxDBServiceRolePolicy \
    --description "new description"
```

   Windows

```
$ aws iam update-role-description ^
    --role-name AmazonTimestreamInfluxDBServiceRolePolicy ^
    --description "new description"
```

Editando uma descrição de função vinculada ao serviço () IAM API

Você pode usar o IAM API para editar uma descrição de função vinculada ao serviço.

Para alterar a descrição de uma função vinculada ao serviço () API

1. (Opcional) Para ver a descrição atual de uma função, use a IAM API operação GetRole.

   Example

```
https://iam.amazonaws.com/
   ?Action=GetRole
   &RoleName=AmazonTimestreamInfluxDBServiceRolePolicy
   &Version=2010-05-08
   &AUTHPARAMS
```

2. Para atualizar a descrição de uma função, use a IAM API operação UpdateRoleDescription.

Example

```
https://iam.amazonaws.com/
   ?Action=UpdateRoleDescription
   &RoleName=AmazonTimestreamInfluxDBServiceRolePolicy
   &Version=2010-05-08
   &Description="New description"
```

Excluindo uma função vinculada ao serviço do Amazon Timestream para InfluxDB

Se você não precisar mais usar um atributo ou serviço que requer uma função vinculada a serviço, é recomendável excluí-la. Dessa forma, você não tem uma entidade não utilizada que não seja monitorada ativamente ou mantida. No entanto, você deve limpar sua função vinculada ao serviço antes de excluí-la.

O Amazon Timestream para InfluxDB não exclui a função vinculada ao serviço para você.

Limpar uma função vinculada ao serviço

Antes de poder usar IAM para excluir uma função vinculada ao serviço, primeiro confirme se a função não tem recursos (clusters) associados a ela.

Para verificar se a função vinculada ao serviço tem uma sessão ativa no console IAM

1. Faça login no AWS Management Console e abra o IAM console em https://console.aws.amazon.com/iam/.
2. No painel de navegação esquerdo do IAM console, escolha Funções. Em seguida, escolha o nome (não a caixa de seleção) da AmazonTimestreamInflux DBServiceRolePolicy função.
3. Na página Resumo para a função selecionada, escolha a guia Consultor de Acesso.
4. Na guia Consultor de Acesso, revise a atividade recente para a função vinculada ao serviço.

Excluindo uma função vinculada ao serviço (console) IAM

Você pode usar o IAM console para excluir uma função vinculada ao serviço.

Para excluir uma função vinculada ao serviço (console)

1. Faça login no AWS Management Console e abra o IAM console em https://console.aws.amazon.com/iam/.

2. No painel de navegação esquerdo do IAM console, escolha Funções. Selecione a caixa de marcação ao lado do nome da função que você deseja excluir, não o nome ou a linha em si.
3. Em ações de Função na parte superior da página, escolha a função Excluir.
4. Na página de confirmação, revise os dados do último acesso ao serviço, que mostram quando cada uma das funções selecionadas acessou um AWS serviço pela última vez. Isso ajuda você a confirmar se a função está ativo no momento. Se quiser prosseguir, escolha Sim, Excluir para enviar a função vinculada ao serviço para exclusão.
5. Assista às notificações do IAM console para monitorar o progresso da exclusão da função vinculada ao serviço. Como a exclusão da função IAM vinculada ao serviço é assíncrona, depois de enviar a função para exclusão, a tarefa de exclusão pode ser bem-sucedida ou falhar. Se a tarefa obtiver êxito, você poderá escolher Visualizar Detalhes ou Visualizar Recursos a partir das notificações para saber por que a exclusão falhou.

Excluindo uma função vinculada ao serviço () IAM CLI

Você pode usar IAM as operações do AWS Command Line Interface para excluir uma função vinculada ao serviço.

Para excluir uma função vinculada ao serviço () CLI

1. Se você não souber o nome da função vinculada ao serviço que deseja excluir, insira o seguinte comando. Esse comando lista as funções e seus nomes de recursos da Amazon (ARNs) em sua conta.

   ```
   $ aws iam get-role --role-name role-name
   ```

   Use o nome da função, não oARN, para se referir às funções com as CLI operações. Por exemplo, se uma função tem o ARN`arn:aws:iam::123456789012:role/myrole`, você se refere à função como**`myrole`**.
2. Como uma função vinculada ao serviço não podem ser excluída se estiver sendo usada ou tiver recursos associados, você deverá enviar uma solicitação de exclusão. Essa solicitação poderá ser negada se essas condições não forem atendidas. Você deve capturar o `deletion-task-id` da resposta para verificar o status da tarefa de exclusão. Insira o seguinte para enviar uma solicitação de exclusão de função vinculada ao serviço.

   ```
   $ aws iam delete-service-linked-role --role-name role-name
   ```

3. Insita o seguinte para verificar o estado da tarefa de exclusão.

```
$ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
```

   O status da tarefa de exclusão pode ser `NOT_STARTED`, `IN_PROGRESS`, `SUCCEEDED`, ou `FAILED`. Se a exclusão falhar, a chamada informará o motivo de falha para que você possa solucionar o problema.

Excluindo uma função vinculada ao serviço () IAM API

Você pode usar o IAM API para excluir uma função vinculada ao serviço.

Para excluir uma função vinculada ao serviço () API

1. Para enviar uma solicitação de exclusão de um roll vinculada ao serviço, chame DeleteServiceLinkedRole. Na solicitação, especifique um nome de função.

   Como uma função vinculada ao serviço não podem ser excluída se estiver sendo usada ou tiver recursos associados, você deverá enviar uma solicitação de exclusão. Essa solicitação poderá ser negada se essas condições não forem atendidas. Você deve capturar o `DeletionTaskId` da resposta para verificar o status da tarefa de exclusão.
2. Para verificar o status da exclusão, chame GetServiceLinkedRoleDeletionStatus. Na solicitação, especifique `DeletionTaskId` o.

   O status da tarefa de exclusão pode ser `NOT_STARTED`, `IN_PROGRESS`, `SUCCEEDED`, ou `FAILED`. Se a exclusão falhar, a chamada informará o motivo de falha para que você possa solucionar o problema.

Regiões suportadas pelo Amazon Timestream para funções vinculadas ao serviço InfluxDB

O Amazon Timestream para InfluxDB suporta o uso de funções vinculadas a serviços em todas as regiões em que o serviço está disponível. Para obter mais informações, consulte Endpoints de serviço da AWS.

AWS políticas gerenciadas para Amazon Timestream para InfluxDB

Para adicionar permissões a usuários, grupos e funções, é mais fácil usar políticas AWS gerenciadas do que escrever políticas você mesmo. É preciso tempo e experiência para criar políticas gerenciadas pelo IAM cliente que forneçam à sua equipe somente as permissões necessárias. Para começar rapidamente, você pode usar nossas políticas AWS gerenciadas. Essas políticas abrangem casos de uso comuns e estão disponíveis em sua AWS conta. Para obter mais informações sobre políticas AWS gerenciadas, consulte políticas AWS gerenciadas no Guia IAM do usuário.

AWS os serviços mantêm e atualizam as políticas AWS gerenciadas. Você não pode alterar as permissões nas políticas AWS gerenciadas. Os serviços ocasionalmente acrescentam permissões adicionais a uma política gerenciada pela AWS para oferecer suporte a novos recursos. Esse tipo de atualização afeta todas as identidades (usuários, grupos e funções) em que a política está anexada. É mais provável que os serviços atualizem uma política gerenciada pela AWS quando um novo recurso for iniciado ou novas operações se tornarem disponíveis. Os serviços não removem as permissões de uma política AWS gerenciada, portanto, as atualizações de políticas não violarão suas permissões existentes.

Além disso, AWS oferece suporte a políticas gerenciadas para funções de trabalho que abrangem vários serviços. Por exemplo, a política ReadOnlyAccess AWS gerenciada fornece acesso somente de leitura a todos os AWS serviços e recursos. Quando um serviço lança um novo recurso, AWS adiciona permissões somente de leitura para novas operações e recursos. Para obter uma lista e descrições das políticas de funções de trabalho, consulte políticas AWS gerenciadas para funções de trabalho no Guia IAM do usuário.

AWS política gerenciada: AmazonTimestreamInflux DBServiceRolePolicy

Você não pode anexar a política AmazonTimestreamInflux DBServiceRolePolicy AWS gerenciada às identidades em sua conta. Essa política faz parte da função vinculada ao serviço de AWS TimestreamforInflux banco de dados. Essa função permite que o serviço gerencie interfaces de rede e grupos de segurança em sua conta.

O Timestream for InfluxDB usa as permissões desta política para gerenciar grupos de EC2 segurança e interfaces de rede. Isso é necessário para gerenciar o Timestream para instâncias de banco de dados InfluxDB.

Para revisar esta política em JSON formato, consulte AmazonTimestreamInfluxDBServiceRolePolicy.

AWS- políticas gerenciadas para Amazon Timestream para InfluxDB

AWS aborda muitos casos de uso comuns fornecendo IAM políticas autônomas que são criadas e administradas pela AWS. As políticas gerenciadas concedem permissões necessárias para casos de uso comuns, de maneira que você possa evitar a necessidade de investigar quais permissões são necessárias. Para obter mais informações, consulte Políticas AWS gerenciadas no Guia IAM do usuário.

As seguintes políticas AWS gerenciadas, que você pode anexar aos usuários em sua conta, são específicas do Timestream for InfluxDB:

AmazonTimestreamInfluxDBFullAccess

Você pode anexar a `AmazonTimestreamInfluxDBFullAccess` política às suas IAM identidades. Essa política concede permissões administrativas que permitem acesso total a todos os recursos do Timestream for InfluxDB.

Você também pode criar suas próprias IAM políticas personalizadas para permitir permissões para o Amazon Timestream para ações do InfluxDB. API Você pode anexar essas políticas personalizadas aos IAM usuários ou grupos que exigem essas permissões.

Para revisar esta política em JSON formato, consulte AmazonTimestreamInfluxDBFullAccess.

Timestream para atualizações do InfluxDB em políticas gerenciadas AWS

Veja detalhes sobre as atualizações das políticas AWS gerenciadas do Timestream for InfluxDB desde que esse serviço começou a rastrear essas alterações. Para alertas automáticos sobre alterações nesta página, assine o RSS feed na página de histórico de documentos do Timestream for InfluxDB.

Alteração	Descrição	Data
AmazonTimestreamInfluxDBFullAccess: atualizar para uma política existente	A `ec2:DescribeRouteTables` ação foi adicionad	10/08/2024

Alteração	Descrição	Data
	a à política `AmazonTimestreamInfluxDBFullAccess` gerenciada existente. Essa ação é usada para descrever suas tabelas de rotas	
AWS política gerenciada: AmazonTimestreamInfluxDBServiceRolePolicy – Nova política	O Amazon Timestream for InfluxDB adicionou uma nova política que permite ao serviço gerenciar interfaces de rede e grupos de segurança em sua conta.	14/03/2024
AmazonTimestreamInfluxDBFullAccess – Nova política	O Amazon Timestream for InfluxDB adicionou uma nova política para fornecer acesso administrativo completo para criar, atualizar, excluir e listar instâncias do Amazon Timestream InfluxDB e criar e listar grupos de parâmetros.	14/03/2024

Conectando-se ao Timestream para InfluxDB por meio de um endpoint VPC

Você pode se conectar diretamente ao Timestream for InfluxDB por meio de um endpoint de interface privada em sua nuvem privada virtual (). VPC Quando você usa um VPC endpoint de interface, a comunicação entre você VPC e o Timestream for InfluxDB é conduzida inteiramente dentro da rede. AWS

O Timestream for InfluxDB suporta endpoints da Amazon Virtual Private Cloud (AmazonVPC) alimentados por. AWS PrivateLink Cada VPC endpoint é representado por uma ou mais interfaces de rede elástica (ENIs) com endereços IP privados em suas VPC sub-redes.

O VPC endpoint da interface conecta você VPC diretamente ao Timestream for InfluxDB sem um gateway de internet, NAT dispositivo, VPN conexão ou conexão. AWS Direct Connect As instâncias

em seu VPC não precisam de endereços IP públicos para se comunicar com o Timestream for InfluxDB.

Regiões

O Timestream for InfluxDB suporta VPC endpoints e VPC políticas de endpoint em todos os quais o Timestream for InfluxDB é Regiões da AWS suportado.

Tópicos

- Considerações sobre o Timestream for InfluxDB endpoints VPC
- Criando um VPC endpoint para Timestream for InfluxDB
- Conectando-se a um Timestream para endpoint InfluxDB VPC
- Controlando o acesso a um VPC endpoint
- Usando um VPC endpoint em uma declaração de política
- Registrando seu VPC endpoint

Considerações sobre o Timestream for InfluxDB endpoints VPC

Antes de configurar um VPC endpoint de interface para Timestream for InfluxDB, revise o tópico Propriedades e limitações do endpoint de interface no Guia.AWS PrivateLink

O suporte do Timestream for InfluxDB para um VPC endpoint inclui o seguinte.

- Você pode usar seu VPC endpoint para chamar todas as operações do Timestream for API InfluxDB a partir do seu. VPC
- Você pode usar AWS CloudTrail registros para auditar o uso dos recursos do Timestream for InfluxDB por meio do endpoint. VPC Para obter detalhes, consulte Registrando seu VPC endpoint.

Criando um VPC endpoint para Timestream for InfluxDB

Você pode criar um VPC endpoint para o Timestream for InfluxDB usando o console VPC da Amazon ou o Amazon. VPC API Para obter mais informações, consulte Criar um endpoint de interface no Guia do usuário do AWS PrivateLink .

- Para criar um VPC endpoint para o Timestream for InfluxDB, use o seguinte nome de serviço:

```
com.amazonaws.region.timestream-influxdb
```

Por exemplo, na Região Oeste dos EUA (Oregon) (`us-west-2`), o nome do serviço seria:

```
com.amazonaws.us-west-2.timestream-influxdb
```

Para facilitar o uso do VPC endpoint, você pode habilitar um DNSnome privado para seu VPC endpoint. Se você selecionar a opção Ativar DNS nome, o Timestream padrão para o nome do DNS host do InfluxDB será resolvido para o seu endpoint. VPC Por exemplo, `https://timestream-influxdb.us-west-2.amazonaws.com` resolveria para um VPC endpoint conectado ao nome do `com.amazonaws.us-west-2.timestream-influxdb` serviço.

Essa opção facilita o uso do VPC endpoint. O AWS SDKs e AWS CLI usa o Timestream padrão para o DNS nome de host do InfluxDB por padrão, portanto, você não precisa especificar o VPC endpoint URL em aplicativos e comandos.

Para mais informações, consulte Acessar um serviço por meio de um endpoint de interface no Guia do AWS PrivateLink .

Conectando-se a um Timestream para endpoint InfluxDB VPC

Você pode se conectar ao Timestream for InfluxDB por meio do VPC endpoint usando um, o ou. AWS SDK AWS CLI AWS Tools for PowerShell Para especificar o VPC endpoint, use seu DNS nome.

Se você ativou nomes de host privados ao criar seu VPC endpoint, não precisará especificar o VPC endpoint URL em seus CLI comandos ou na configuração do aplicativo. O Timestream padrão para o DNS nome de host do InfluxDB é resolvido em seu endpoint. VPC O AWS CLI and SDKs usa esse nome de host por padrão, para que você possa começar a usar o VPC endpoint para se conectar a um endpoint regional Timestream for InfluxDB sem alterar nada em seus scripts e aplicativos.

Para usar nomes de host privados, os `enableDnsSupport` atributos `enableDnsHostnames` e do seu VPC devem ser definidos como. `true` Para definir esses atributos, use a ModifyVpcAttributeoperação. Para obter detalhes, consulte Visualizar e atualizar seus DNS atributos VPC no Guia do VPC usuário da Amazon.

Controlando o acesso a um VPC endpoint

Para controlar o acesso ao seu VPC endpoint para o Timestream for InfluxDB, anexe uma política de endpoint ao seu VPCendpoint. VPC A política de endpoint determina se os diretores podem usar o VPC endpoint para chamar as operações do Timestream for InfluxDB no Timestream for InfluxDB.

Você pode criar uma política de VPC endpoint ao criar seu endpoint e pode alterar a política de VPC endpoint a qualquer momento. Use o console VPC de gerenciamento CreateVpcEndpointou as ModifyVpcEndpointoperações. Você também pode criar e alterar uma política de VPC endpoint usando um AWS CloudFormation modelo. Para obter ajuda sobre como usar o console VPC de gerenciamento, consulte Criar um endpoint de interface e Modificar um endpoint de interface no Guia.AWS PrivateLink

> **Note**
>
> O Timestream for InfluxDB suporta políticas de VPC endpoint a partir de julho de 2020. VPCos endpoints do Timestream for InfluxDB que foram criados antes dessa data têm a política de VPC endpoint padrão, mas você pode alterá-la a qualquer momento.

Tópicos

- Sobre as VPC políticas de endpoint
- Política de VPC endpoint padrão
- Criação de uma política VPC de endpoint
- Visualizando uma política VPC de endpoint

Sobre as VPC políticas de endpoint

Para que uma solicitação do Timestream for InfluxDB que usa um VPC endpoint seja bem-sucedida, o principal requer permissões de duas fontes:

- Uma IAMpolítica deve dar permissão principal para chamar a operação no recurso.
- Uma política de VPC endpoint deve dar permissão ao principal para usar o endpoint para fazer a solicitação.

Política de VPC endpoint padrão

Cada VPC endpoint tem uma política de VPC endpoint, mas você não precisa especificar a política. Se você não especificar uma política, a política de endpoint padrão permitirá todas as operações por todas as entidades principais em todos os recursos sobre o endpoint.

No entanto, para recursos do Timestream for InfluxDB, o diretor também deve ter permissão para chamar a operação a partir de uma IAMpolítica. Portanto, na prática, a política padrão diz que, se

um diretor tiver permissão para chamar uma operação em um recurso, ele também poderá chamá-la usando o endpoint.

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Principal": "*",
      "Resource": "*"
    }
  ]
}
```

Para permitir que os diretores usem o VPC endpoint somente para um subconjunto de suas operações permitidas, crie ou atualize a política do VPC endpoint.

Criação de uma política VPC de endpoint

Uma política de VPC endpoint determina se um principal tem permissão para usar o VPC endpoint para realizar operações em um recurso. Para recursos do Timestream for InfluxDB, o diretor também deve ter permissão para realizar as operações a partir de uma política, IAM

Cada declaração de política de VPC endpoint exige os seguintes elementos:

- A entidade principal que pode executar ações
- As ações que podem ser executadas
- Os recursos nos quais as ações podem ser executadas

A declaração de política não especifica o VPC endpoint. Em vez disso, ela se aplica a qualquer VPC endpoint ao qual a política esteja anexada. Para obter mais informações, consulte Controle do acesso a serviços com VPC endpoints no Guia do VPC usuário da Amazon.

AWS CloudTrail registra todas as operações que usam o VPC endpoint.

Visualizando uma política VPC de endpoint

Para visualizar a política de VPC endpoint de um endpoint, use o console VPC de gerenciamento ou a DescribeVpcEndpointsoperação.

O AWS CLI comando a seguir obtém a política para o endpoint com o ID do VPC endpoint especificado.

Antes de executar esse comando, substitua o ID de endpoint demonstrativo por um válido da sua conta.

```
$ aws ec2 describe-vpc-endpoints \
--query 'VpcEndpoints[?VpcEndpointId==`vpc-endpoint-id`].[PolicyDocument]'
--output text
```

Usando um VPC endpoint em uma declaração de política

Você pode controlar o acesso ao Timestream para recursos e operações do InfluxDB quando a solicitação vem VPC ou usa um endpoint. VPC Para fazer isso, use uma das seguintes chaves de condição global em uma IAMpolítica.

- Use a chave de `aws:sourceVpce` condição para conceder ou restringir o acesso com base no VPC endpoint.
- Use a chave de `aws:sourceVpc` condição para conceder ou restringir o acesso com base no VPC que hospeda o endpoint privado.

> **Note**
>
> Tenha cuidado ao criar políticas e IAM políticas importantes com base em seu VPC endpoint. Se uma declaração de política exigir que as solicitações venham de um determinado VPC VPC ponto final, as solicitações de AWS serviços integrados que usam um recurso Timestream for InfluxDB em seu nome podem falhar.
> Além disso, a chave de `aws:sourceIP` condição não é efetiva quando a solicitação vem de um VPCendpoint da Amazon. Para restringir as solicitações a um VPC endpoint, use as chaves de `aws:sourceVpc` condição `aws:sourceVpce` ou. Para obter mais informações, consulte Gerenciamento de identidade e acesso para VPC endpoints e serviços de VPC endpoint no AWS PrivateLink Guia.

Você pode usar essas chaves de condição globais para controlar o acesso a operações como essas CreateDbInstanceque não dependem de nenhum recurso específico.

Registrando seu VPC endpoint

AWS CloudTrail registra todas as operações que usam o VPC endpoint. Quando uma solicitação ao Timestream for InfluxDB usa um VPC endpoint, o VPC ID do endpoint aparece na entrada de AWS CloudTrail log que registra a solicitação. Você pode usar o ID do endpoint para auditar o uso do seu Timestream for InfluxDB endpoint. VPC

No entanto, seus CloudTrail registros não incluem operações solicitadas por diretores em outras contas ou solicitações de Timestream para operações do InfluxDB no Timestream para recursos e aliases do InfluxDB em outras contas. Além disso, para proteger suasVPC, as solicitações que são negadas por uma política de VPC endpoint, mas que de outra forma seriam permitidas, não são registradas AWS CloudTrail.

Registro e monitoramento no Timestream para InfluxDB

O monitoramento é uma parte importante da manutenção da confiabilidade, disponibilidade e desempenho do Timestream for InfluxDB e suas soluções. AWS Você deve coletar dados de monitoramento de todas as partes da sua AWS solução para poder depurar com mais facilidade uma falha multiponto, caso ocorra. No entanto, antes de começar a monitorar o Timestream para o InfluxDB, você deve criar um plano de monitoramento que inclua respostas às seguintes perguntas:

- Quais são seus objetivos de monitoramento?
- Quais recursos você vai monitorar?
- Com que frequência você vai monitorar esses recursos?
- Quais ferramentas de monitoramento você usará?
- Quem realizará o monitoramento das tarefas?
- Quem deve ser notificado quando algo der errado?

A próxima etapa é estabelecer uma linha de base para o desempenho normal do Timestream for InfluxDB em seu ambiente, medindo o desempenho em vários momentos e sob diferentes condições de carga. Ao monitorar o Timestream do InfluxDB, armazene dados históricos de monitoramento para que você possa compará-los com os dados de desempenho atuais, identificar padrões normais de desempenho e anomalias de desempenho e criar métodos para resolver problemas.

Para estabelecer uma linha de base, você deve, no mínimo, monitorar os seguintes itens:

- Erros do sistema, para que você possa determinar se alguma solicitação resultou em erro.

Tópicos

- Ferramentas de monitoramento
- Registrando o Timestream para chamadas do InfluxDB API com AWS CloudTrail

Ferramentas de monitoramento

AWS fornece várias ferramentas que você pode usar para monitorar o Timestream do InfluxDB. É possível configurar algumas dessas ferramentas para fazer o monitoramento em seu lugar, e, ao mesmo tempo, algumas das ferramentas exigem intervenção manual. Recomendamos que as tarefas de monitoramento sejam automatizadas ao máximo possível.

Tópicos

- Ferramentas de monitoramento automatizadas
- Ferramentas de monitoramento manual

Ferramentas de monitoramento automatizadas

Você pode usar as seguintes ferramentas de monitoramento automatizado para assistir ao Timestream for InfluxDB e relatar quando algo está errado:

- Amazon CloudWatch Alarms — Observe uma única métrica durante um período de tempo especificado por você e execute uma ou mais ações com base no valor da métrica em relação a um determinado limite em vários períodos. A ação é uma notificação enviada para um tópico do Amazon Simple Notification Service (AmazonSNS) ou para uma política do Amazon EC2 Auto Scaling. CloudWatch os alarmes não invocam ações simplesmente porque estão em um determinado estado; o estado deve ter sido alterado e mantido por um determinado número de períodos. Para obter mais informações, consulte Monitoramento com a Amazon CloudWatch.

Ferramentas de monitoramento manual

Outra parte importante do monitoramento do Timestream para o InfluxDB envolve o monitoramento manual dos itens que os CloudWatch alarmes não cobrem. O Timestream for InfluxDB,, CloudWatch Trusted Advisor, e outros AWS Management Console painéis fornecem uma at-a-glance visão do estado do seu ambiente. AWS

- A página CloudWatch inicial mostra o seguinte:
  - Alertas e status atual

- Gráficos de alertas e recursos
- Estado de integridade do serviço

Além disso, você pode usar CloudWatch para fazer o seguinte:

- Crie painéis personalizados para monitorar os serviços com os quais você se preocupa.
- Colocar em gráfico dados de métrica para solucionar problemas e descobrir tendências
- Pesquise e navegue em todas as suas métricas AWS de recursos
- Criar e editar alertas para ser notificado sobre problemas

Registrando o Timestream para chamadas do InfluxDB API com AWS CloudTrail

O Timestream for InfluxDB é integrado com AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, função ou AWS serviço no Timestream for InfluxDB. CloudTrail captura API chamadas de Data Definition Language (DDL) para Timestream for InfluxDB como eventos. As chamadas capturadas incluem chamadas do console Timestream para InfluxDB e chamadas de código para o Timestream para operações do InfluxDB. API Se você criar uma trilha, poderá habilitar a entrega contínua de CloudTrail eventos para um bucket do Amazon Simple Storage Service (Amazon S3), incluindo eventos para Timestream for InfluxDB. Se você não configurar uma trilha, ainda poderá ver os eventos mais recentes no CloudTrail console no Histórico de eventos. Usando as informações coletadas por CloudTrail, você pode determinar a solicitação que foi feita ao Timestream for InfluxDB, o endereço IP do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e detalhes adicionais.

Para saber mais sobre isso CloudTrail, consulte o Guia AWS CloudTrail do usuário.

Timestream para informações do InfluxDB em CloudTrail

CloudTrail é ativado em sua AWS conta quando você cria a conta. Quando a atividade ocorre no Timestream for InfluxDB, essa atividade é registrada em um CloudTrail evento junto com outros eventos de AWS serviço no histórico de eventos. Você pode visualizar, pesquisar e baixar eventos recentes em sua AWS conta. Para obter mais informações, consulte Visualização de eventos com histórico de CloudTrail eventos.

Para um registro contínuo de eventos em sua AWS conta, incluindo eventos do Timestream for InfluxDB, crie uma trilha. Uma trilha permite CloudTrail entregar arquivos de log para um bucket do Amazon S3. Por padrão, quando você cria uma trilha no console, a trilha se aplica a todas as AWS regiões. A trilha registra eventos de todas as regiões na AWS partição e entrega os arquivos de

log ao bucket do Amazon S3 que você especificar. Além disso, você pode configurar outros AWS serviços para analisar e agir com base nos dados de eventos coletados nos CloudTrail registros.

Para obter mais informações, consulte os seguintes tópicos no Guia do usuário do AWS CloudTrail :

- Visão geral da criação de uma trilha
- CloudTrail Serviços e integrações compatíveis
- Configurando as SNS notificações da Amazon para CloudTrail
- Recebendo arquivos de CloudTrail log de várias regiões
- Recebendo arquivos de CloudTrail log de várias contas
- Registrando eventos de dados

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar:

- Se a solicitação foi feita com credenciais de usuário root ou AWS Identity and Access Management (IAM)
- Se a solicitação foi feita com credenciais de segurança temporárias de um perfil ou de um usuário federado
- Se a solicitação foi feita por outro AWS serviço

Para obter mais informações, consulte o CloudTrail userIdentityElemento.

Validação de conformidade do Amazon Timestream para InfluxDB

Auditores terceirizados avaliam a segurança e a conformidade do Amazon Timestream for InfluxDB como parte de vários programas de conformidade. AWS Incluindo o seguinte:

- GDPR
- HIPAA
- PCI
- SOC

Para saber se um AWS service (Serviço da AWS) está dentro do escopo de programas de conformidade específicos, consulte Serviços da AWS Escopo por Programa de Conformidade

Serviços da AWS e escolha o programa de conformidade em que você está interessado. Para obter informações gerais, consulte Programas de AWS conformidade Programas AWS de .

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte Baixar relatórios em AWS Artifact .

Sua responsabilidade de conformidade ao usar Serviços da AWS é determinada pela confidencialidade de seus dados, pelos objetivos de conformidade de sua empresa e pelas leis e regulamentações aplicáveis. AWS fornece os seguintes recursos para ajudar na conformidade:

- Guias de início rápido sobre segurança e conformidade — Esses guias de implantação discutem considerações arquitetônicas e fornecem etapas para a implantação de ambientes básicos AWS focados em segurança e conformidade.
- Arquitetura para HIPAA segurança e conformidade na Amazon Web Services — Este whitepaper descreve como as empresas podem usar AWS para criar HIPAA aplicativos qualificados.

> **Note**
>
> Nem todos Serviços da AWS são HIPAA elegíveis. Para obter mais informações, consulte a Referência de serviços HIPAA elegíveis.

- AWS Recursos de https://aws.amazon.com/compliance/resources/ de conformidade — Essa coleção de pastas de trabalho e guias pode ser aplicada ao seu setor e local.
- AWS Guias de conformidade do cliente — Entenda o modelo de responsabilidade compartilhada sob a ótica da conformidade. Os guias resumem as melhores práticas de proteção Serviços da AWS e mapeiam as diretrizes para controles de segurança em várias estruturas (incluindo o Instituto Nacional de Padrões e Tecnologia (NIST), o Conselho de Padrões de Segurança do Setor de Cartões de Pagamento (PCI) e a Organização Internacional de Padronização ()). ISO
- Avaliação de recursos com regras no Guia do AWS Config desenvolvedor — O AWS Config serviço avalia o quão bem suas configurações de recursos estão em conformidade com as práticas internas, as diretrizes e os regulamentos do setor.
- AWS Security Hub— Isso AWS service (Serviço da AWS) fornece uma visão abrangente do seu estado de segurança interno AWS. O Security Hub usa controles de segurança para avaliar os recursos da AWS e verificar a conformidade com os padrões e as práticas recomendadas do setor de segurança. Para obter uma lista dos serviços e controles aceitos, consulte a Referência de controles do Security Hub.

- Amazon GuardDuty — Isso AWS service (Serviço da AWS) detecta possíveis ameaças às suas cargas de trabalho Contas da AWS, contêineres e dados monitorando seu ambiente em busca de atividades suspeitas e maliciosas. GuardDuty pode ajudá-lo a atender a vários requisitos de conformidade, por exemplo PCIDSS, atendendo aos requisitos de detecção de intrusões exigidos por determinadas estruturas de conformidade.
- AWS Audit Manager— Isso AWS service (Serviço da AWS) ajuda você a auditar continuamente seu AWS uso para simplificar a forma como você gerencia o risco e a conformidade com as regulamentações e os padrões do setor.

Resiliência no Amazon Timestream para InfluxDB

A infraestrutura AWS global é construída em torno de AWS regiões e zonas de disponibilidade. AWS As regiões fornecem várias zonas de disponibilidade fisicamente separadas e isoladas, conectadas a redes de baixa latência, alta taxa de transferência e alta redundância. Com as zonas de disponibilidade, é possível projetar e operar aplicativos e bancos de dados que automaticamente executam o failover entre as zonas sem interrupção. As zonas de disponibilidade são altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data centers tradicionais.

Para obter mais informações sobre AWS regiões e zonas de disponibilidade, consulte Infraestrutura AWS global.

O Amazon Timestream for InfluxDB faz backups internos periodicamente e os retém por 24 horas para oferecer suporte à disponibilidade e durabilidade. Os instantâneos são tirados durante as exclusões e mantidos por 30 dias para apoiar as restaurações. Para acessá-los ou usá-los, registre um ticket no AWS suporte.

Você pode criar sua instância com recursos de recuperação Multi-AZ. Para obter mais informações, consulte Implantações de instâncias de banco de dados multi-AZ.

Segurança de infraestrutura no Amazon Timestream para InfluxDB

Como um serviço gerenciado, o Amazon Timestream for InfluxDB é protegido pelos procedimentos globais de segurança de rede descritos AWS no whitepaper Amazon Web Services: Visão geral dos processos de segurança.

Você usa API chamadas de plano de controle AWS publicadas para acessar o Timestream for InfluxDB por meio da rede. Para obter mais informações, consulte Planos de controle e planos de

dados. Os clientes devem oferecer suporte ao Transport Layer Security (TLS) 1.2 ou posterior. Recomendamos TLS 1,2 ou 1,3. Os clientes também devem oferecer suporte a pacotes de criptografia com sigilo direto perfeito (), como Ephemeral Diffie-Hellman (PFS) ou Elliptic Curve Ephemeral Diffie-Hellman (). DHE ECDHE A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Além disso, as solicitações devem ser assinadas usando uma ID de chave de acesso e uma chave de acesso secreta associada a um IAM principal. Ou você pode usar o AWS Security Token Service (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

O Timestream for InfluxDB é arquitetado para que seu tráfego seja isolado AWS na região específica em que sua instância Timestream for InfluxDB reside.

Grupos de segurança

Os grupos de segurança controlam o acesso que o tráfego tem dentro e fora de uma instância de banco de dados. Por padrão, o acesso à rede é desativado para uma instância de banco de dados. É possível especificar regras em um grupo de segurança que permitem o acesso de um intervalo de endereço IP, de uma porta ou de grupo de segurança. Depois que as regras de entrada são configuradas, as mesmas regras se aplicam a todas as instâncias de banco de dados associadas a esse grupo de segurança.

Para obter mais informações, consulte Controlando o acesso a uma instância de banco de dados em um VPC.

Configuração e análise de vulnerabilidade no Timestream for InfluxDB

A configuração e os controles de TI são uma responsabilidade compartilhada entre você AWS e você, nosso cliente. Para obter mais informações, consulte o modelo de responsabilidade AWS compartilhada. Além do modelo de responsabilidade compartilhada, os usuários do Timestream for InfluxDB devem estar cientes do seguinte:

- É da responsabilidade do cliente corrigir as aplicações de clientes com as dependências relevantes do lado do cliente.
- Os clientes devem considerar o teste de penetração, se apropriado (consulte testes de https://aws.amazon.com/security/penetração/.)

Resposta a incidentes no Timestream para InfluxDB

Os incidentes do serviço Amazon Timestream para InfluxDB são relatados no Personal Health Dashboard. Você pode aprender mais sobre o painel e AWS Health aqui.

O Timestream for InfluxDB suporta relatórios usando. AWS CloudTrail Para obter mais informações, consulte Registrando o Timestream para chamadas do InfluxDB API com AWS CloudTrail.

Amazon Timestream para API InfluxDB e endpoints de interface () VPC AWS PrivateLink

Você pode estabelecer uma conexão privada entre você VPC e o Amazon Amazon Timestream para endpoints do API plano de controle do InfluxDB criando um endpoint de interface. VPC Os endpoints de interface são alimentados por AWS PrivateLink. AWS PrivateLink permite que você acesse de forma privada o Amazon Timestream API para operações do InfluxDB sem um NAT gateway de internet, dispositivoVPN, conexão ou conexão Direct Connect. AWS

Suas instâncias VPC não precisam de endereços IP públicos para se comunicar com o Amazon Timestream para endpoints do InfluxDB. API Suas instâncias também não precisam de endereços IP públicos para usar nenhum Timestream disponível para operações do API InfluxDB. O tráfego entre você VPC e o Amazon Timestream for InfluxDB não sai da rede Amazon. Cada endpoint de interface é representado por uma ou mais interfaces de rede elástica nas sub-redes. Para obter mais informações sobre interfaces de rede elástica, consulte Interfaces de rede elástica no Guia EC2 do usuário da Amazon.

- Para obter mais informações sobre VPC endpoints, consulte Interface VPC endpoints (AWS PrivateLink) no Amazon VPC User Guide.
- Para obter mais informações sobre o Timestream para operações do InfluxDB, consulte Timestream para API operações do InfluxDB. API

Depois de criar um VPC endpoint de interface, se você habilitar DNS nomes de host privados para o endpoint, o Timestream padrão para o endpoint InfluxDB (https://timestream-influxb.*Region*.amazonaws.com) resolve para seu endpoint. VPC Se você não habilitar DNS nomes de host privados, a Amazon VPC fornecerá um nome de DNS endpoint que você pode usar no seguinte formato:

```
VPC_Endpoint_ID.timestream-influxb.Region.vpce.amazonaws.com
```

Para obter mais informações, consulte Interface VPC Endpoints (AWS PrivateLink) no Guia do VPC usuário da Amazon. O Timestream for InfluxDB suporta fazer chamadas para todas as suas ações dentro do seuAPI. VPC

> **Note**
>
> DNSNomes de host privados só podem ser habilitados para um VPC endpoint no. VPC Se você quiser criar um VPC endpoint adicional, o DNS nome do host privado deve ser desabilitado para ele.

Considerações sobre endpoints VPC

Antes de configurar um VPC endpoint de interface para o Amazon Timestream API para endpoints do InfluxDB, certifique-se de revisar as propriedades e limitações do endpoint da interface no Guia do usuário da Amazon. VPC Todas as API operações do Timestream for InfluxDB que são relevantes para o gerenciamento dos recursos do Amazon Timestream for InfluxDB estão disponíveis para você usar. VPC AWS PrivateLink VPCas políticas de endpoint são suportadas pelo Timestream for InfluxDB endpoints. API Por padrão, o acesso total ao Timestream para API operações do InfluxDB é permitido por meio do endpoint. Para obter mais informações, consulte Controle do acesso a serviços com VPC endpoints no Guia do VPC usuário da Amazon.

Criando um VPC endpoint de interface para o Timestream for InfluxDB API

Você pode criar um VPC endpoint para o Amazon Timestream for InfluxDB usando API o console da Amazon ou o. VPC AWS CLI Para obter mais informações, consulte Criação de um endpoint de interface no Guia do VPC usuário da Amazon.

Depois de criar um VPC endpoint de interface, você pode habilitar nomes de DNS host privados para o endpoint. Ao fazer isso, o endpoint padrão do Amazon Timestream para InfluxDB (https://timestream-influxb).*Region*.amazonaws.com) resolve para seu endpoint. VPC Para obter mais informações, consulte Acessando um serviço por meio de um endpoint de interface no Guia do VPC usuário da Amazon.

Criação de uma política de VPC endpoint para o Amazon Timestream for InfluxDB API

Você pode anexar uma política de endpoint ao seu VPC endpoint que controla o acesso ao Timestream do InfluxDB. API A política especifica o seguinte:

- A entidade principal que pode executar ações.

- As ações que podem ser executadas.
- Os recursos sobre os quais as ações podem ser realizadas.

Para obter mais informações, consulte Controle do acesso a serviços com VPC endpoints no Guia do VPC usuário da Amazon.

Example VPCpolítica de endpoint para ações do Timestream for InfluxDB API

A seguir está um exemplo de uma política de endpoint para o Timestream for InfluxDB. API Quando anexada a um endpoint, essa política concede acesso às API ações listadas do Timestream for InfluxDB para todos os diretores em todos os recursos.

```
{
 "Statement": [{
  "Principal": "*",
  "Effect": "Allow",
  "Action": [
   "timestream-influxb:CreateDbInstance",
   "timestream-influxb:UpdateDbInstance"
  ],
  "Resource": "*"
 }]
}
```

Example VPCpolítica de endpoint que nega todo o acesso de uma conta especificada AWS

A seguinte política de VPC endpoint nega AWS a conta *123456789012* todo o acesso aos recursos usando o endpoint. A política permite todas as ações de outras contas.

```
{
 "Statement": [{
   "Action": "*",
   "Effect": "Allow",
   "Resource": "*",
   "Principal": "*"
  },
  {
   "Action": "*",
   "Effect": "Deny",
   "Resource": "*",
   "Principal": {
    "AWS": [
```

```
      "123456789012"
     ]
    }
   }
  ]
}
```

Melhores práticas de segurança para Timestream for InfluxDB

O Amazon Timestream para InfluxDB fornece vários recursos de segurança a serem considerados ao desenvolver e implementar suas próprias políticas de segurança. As melhores práticas a seguir são diretrizes gerais e não representam uma solução completa de segurança. Como essas práticas recomendadas podem não ser adequadas ou suficientes no seu ambiente, trate-as como considerações úteis em vez de requisitos.

Implemente o acesso de privilégio mínimo

Ao conceder permissões, você decide quem está recebendo quais permissões para quais recursos do Timestream for InfluxDB. Você habilita ações específicas que quer permitir nesses recursos. Portanto, você deve conceder somente as permissões necessárias para executar uma tarefa. A implementação do privilégio de acesso mínimo é fundamental para reduzir o risco de segurança e o impacto que pode resultar de erros ou usuários mal-intencionados.

Use IAM funções

Os aplicativos produtores e clientes devem ter credenciais válidas para acessar o Timestream para instâncias de banco de dados do InfluxDB. Você não deve armazenar AWS credenciais diretamente em um aplicativo cliente ou em um bucket do Amazon S3. Essas são credenciais de longo prazo que não são automaticamente alternadas e podem ter um impacto comercial significativo se forem comprometidas.

Em vez disso, você deve usar uma IAM função para gerenciar credenciais temporárias para seus aplicativos produtores e clientes para acessar o Timestream para instâncias de banco de dados InfluxDB. Quando você usa uma função, não precisa usar credenciais de longo prazo (como um nome de usuário e uma senha ou chaves de acesso) para acessar outros recursos.

Para obter mais informações, consulte os seguintes tópicos no Guia IAM do usuário:

- IAMFunções
- Cenários comuns para funções: usuários, aplicativos e serviços

Use contas AWS Identity and Access Management (IAM) para controlar o acesso ao Amazon Timestream para operações do API InfluxDB, especialmente operações que criam, modificam ou excluem recursos do Amazon Timestream for InfluxDB. Esses recursos incluem instâncias de banco de dados, grupos de segurança e grupos de parâmetros.

- Crie um usuário individual para cada pessoa que gerencia os recursos do Amazon Timestream for InfluxDB, incluindo você. Não use credenciais AWS raiz para gerenciar o Amazon Timestream para recursos do InfluxDB.
- Conceda a cada usuário o conjunto mínimo de permissões necessárias para realizar suas funções.
- Use IAM grupos para gerenciar com eficiência as permissões de vários usuários.
- Mude suas credenciais do IAM regularmente.
- Configure o AWS Secrets Manager para alternar automaticamente os segredos do Amazon Timestream para o InfluxDB. Para obter mais informações, consulte Como alternar seus AWS segredos do Secrets Manager no Guia do usuário do AWS Secrets Manager. Você também pode recuperar a credencial do AWS Secrets Manager programaticamente. Para obter mais informações, consulte Recuperando o valor secreto no Guia do Usuário do AWS Secrets Manager.
- Proteja seu Timestream para tokens de influxo do InfluxDB usando o. API APIfichas

Implemente a criptografia do lado do servidor em recursos dependentes

Dados em repouso e dados em trânsito podem ser criptografados no Timestream for InfluxDB. Para obter mais informações, consulte Criptografia em trânsito.

Use CloudTrail para monitorar API chamadas

O Timestream for InfluxDB é integrado com AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, função ou AWS serviço no Timestream for InfluxDB.

Usando as informações coletadas por CloudTrail, você pode determinar a solicitação que foi feita ao Timestream for InfluxDB, o endereço IP do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e detalhes adicionais.

Para obter mais informações, consulte the section called “Registrando o Timestream para LiveAnalytics API chamadas com AWS CloudTrail”.

O Amazon Timestream para InfluxDB oferece suporte a eventos do plano de controle, mas não ao CloudTrail plano de dados. Para obter mais informações, consulte Planos de controle e planos de dados.

Public accessibility

Quando você executa uma instância de banco de dados dentro de uma nuvem privada virtual (VPC) com base no VPC serviço da Amazon, você pode ativar ou desativar a acessibilidade pública dessa instância de banco de dados. Para designar se a instância de banco de dados que você cria tem um DNS nome que se resolve para um endereço IP público, você usa o parâmetro de acessibilidade pública. Ao usar esse parâmetro, você pode designar se há acesso público à instância de banco de dados

Se sua instância de banco de dados estiver em um, VPC mas não estiver acessível publicamente, você também poderá usar uma AWS Site-to-Site VPN conexão ou uma conexão AWS Direct Connect para acessá-la de uma rede privada.

Se sua instância de banco de dados estiver acessível ao público, tome medidas para evitar ou ajudar a mitigar ameaças relacionadas à negação de serviço. Para obter mais informações, consulte Introdução aos ataques de negação de serviço e Proteção de redes.

APIreferência

Para obter uma lista completa e detalhes do Amazon Timestream para InfluxDBAPIs, consulte Amazon Timestream para InfluxDB. APIs

Para códigos de erro comuns a todos os AWS serviços, consulte a seção AWS Support.

Histórico do documento

Alteração	Descrição	Data
Atualização somente com documentação	O tópico Cotas foi atualizado para separar as cotas padrão e os limites do sistema.	22 de outubro de 2024
O Amazon Timestream agora oferece suporte a insights de consulta	O Timestream agora inclui suporte para o recurso de insights de consulta que ajuda você a otimizar suas consultas , melhorar seu desempenho e reduzir custos.	22 de outubro de 2024

Atualização do Amazon Timestream para InfluxDB para uma política existente.	O Amazon Timestream for InfluxDB adicionou a ação à política `ec2:DescribeRouteTables` gerenciada `AmazonTimestreamInfluxDBFullAccess` existente para descrever suas tabelas de rotas	8 de outubro de 2024
`AmazonTimestreamReadOnlyAccess` — Atualização de uma política existente	O Timestream for LiveAnalytics adicionou a `DescribeAccountSettings` permissão à política `AmazonTimestreamReadOnlyAccess` gerenciada para descrever Conta da AWS as configurações.	3 de junho de 2024
LiveAnalytics Por enquanto, o Amazon Timestream oferece suporte às unidades de computação do Timestream () TCUs	LiveAnalytics Por enquanto, o Amazon Timestream inclui suporte para Timestream Compute Units TCUs () para medir a capacidade computacional alocada para suas necessidades de consulta.	29 de abril de 2024

Novas políticas adicionadas	O Amazon Timestream for InfluxDB adicionou duas novas políticas: uma que permite que o serviço gerencie interfaces de rede e grupos de segurança em sua conta. Para obter mais informaçõ es, consulte AmazonTim estreamInfluxDBServiceRoleP olicy. Outro que fornece acesso administrativo total para criar, atualizar, excluir e listar instâncias do Amazon Timestream InfluxDB e criar e listar grupos de parâmetro s. Para obter mais informaçõ es, consulte AmazonTim estreamInfluxDBFullAccess.	14 de março de 2024
O Amazon Timestream para InfluxDB agora está disponível ao público em geral.	Esta documentação abrange a versão inicial do Amazon Timestream para InfluxDB.	14 de março de 2024

Os eventos do Amazon Timestream LiveAnalytics for Query estão disponíveis em AWS CloudTrail	LiveAnalytics Por enquanto, o Amazon Timestream publica API eventos de dados de consulta no. AWS CloudTrai l Os clientes podem auditar todas as API solicitações de consulta feitas em suas AWS contas e ver informações como qual IAM usuário/função fez a solicitação, quando a solicitação foi feita, quais bancos de dados e tabelas foram consultados e o ID da consulta da solicitação.	12 de setembro de 2023
Amazon Timestream para LiveAnalytics UNLOAD	LiveAnalytics Por enquanto, o Amazon Timestream UNLOAD suporta a exportação dos resultados da consulta para o S3.	12 de maio de 2023
Amazon Timestream LiveAnalytics para atualização de uma política existente.	Permissões de carregamento em lote adicionadas a uma política gerenciada.	24 de fevereiro de 2023
Amazon Timestream LiveAnalytics para carregamento em lote.	LiveAnalytics Por enquanto, o Amazon Timestream oferece suporte à funcionalidade de carregamento em lote.	24 de fevereiro de 2023
Por enquanto, o Amazon Timestream oferece suporte LiveAnalytics . AWS Backup	Por enquanto, o Amazon Timestream oferece suporte LiveAnalytics . AWS Backup	14 de dezembro de 2022

Amazon Timestream LiveAnalytics para atualizações de políticas gerenciadas AWS	Novas informações sobre políticas AWS gerenciadas e Amazon Timestream LiveAnalytics for, incluindo atualizações de políticas gerenciadas existentes.	29 de novembro de 2021
O Amazon Timestream LiveAnalytics for oferece suporte a consultas programadas	LiveAnalytics Por enquanto, o Amazon Timestream suporta a execução de uma consulta em seu nome, com base em uma programação.	29 de novembro de 2021
Amazon Timestream LiveAnalytics para suporte à loja magnética.	LiveAnalytics Por enquanto, o Amazon Timestream suporta o uso de armazenamento magnético para suas gravações em tabelas.	29 de novembro de 2021
Amazon Timestream LiveAnalytics para registros de várias medidas.	LiveAnalytics Por enquanto, o Amazon Timestream oferece suporte a um formato mais compacto para armazenar seus dados de séries temporais.	29 de novembro de 2021
Amazon Timestream LiveAnalytics para atualizações de políticas gerenciadas AWS	Novas informações sobre políticas AWS gerenciadas e Amazon Timestream LiveAnalytics for, incluindo atualizações de políticas gerenciadas existentes.	24 de maio de 2021

O Amazon Timestream LiveAnalytics for já está disponível na região da Europa (Frankfurt).	O Amazon Timestream LiveAnalytics for agora está disponível ao público em geral na região da Europa (Frankfurt) (). `eu-central-1`	23 de abril de 2021
O Amazon Timestream LiveAnalytics for is agora VPC oferece suporte a endpoints ().AWS PrivateLink	LiveAnalytics Por enquanto, o Amazon Timestream suporta o uso VPC de endpoints ().AWS PrivateLink	23 de março de 2021
O Amazon Timestream agora oferece suporte a consultas entre tabelas.	Você pode usar o Amazon Timestream LiveAnalytics para executar consultas entre tabelas.	10 de fevereiro de 2021
LiveAnalytics Por enquanto, o Amazon Timestream oferece suporte a estatísticas aprimoradas de execução de consultas.	LiveAnalytics Por enquanto, o Amazon Timestream oferece suporte a estatísticas aprimoradas de execução de consultas, como a quantidade de dados escaneados.	10 de fevereiro de 2021
LiveAnalytics Por enquanto, o Amazon Timestream oferece suporte a funções avançadas de séries temporais.	Você pode usar o Amazon Timestream LiveAnalytics para SQL executar consultas com funções avançadas de séries temporais, como derivadas, integrais e correlações.	10 de fevereiro de 2021
O Amazon Timestream LiveAnalytics for HIPAA já ISO está em conformidade com o Amazon Timestream. PCI	Agora você pode usar o Amazon Timestream LiveAnalytics para cargas de trabalho que HIPAA exigem infraestrutura compatível com ISO e. PCI	27 de janeiro de 2021

LiveAnalytics Por enquanto, o Amazon Timestream oferece suporte a telegraf e grafana de código aberto.	Agora você pode usar o Telegraf, o agente de servidor de código aberto e orientado por plug-ins para coletar e reportar métricas, e o Grafana, a plataforma de análise e monitoramento de código aberto para bancos de dados, com o Amazon Timestream for. LiveAnalytics	25 de novembro de 2020
O Amazon Timestream LiveAnalytics for já está disponível ao público em geral.	Esta documentação abrange a versão inicial do Amazon LiveAnalytics Timestream for.	30 de setembro de 2020

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.